1100/70 Systems

Processor and Storage

Hardware Programmer Reference

A

SPERRY

UP-9652 Rev. 1

Page Status Summary

Section	Pages	Update
Cover/Disclaimer		
PSS	1	
Contents	1 - 10	
Section 1	1	
Section 2	1 - 12	
Section 3	1 - 19	
Section 4	1 - 33	
Section 5	1 - 8	
Section 6	1 - 30	
Section 7	1 - 11	
Section 8	1 - 29	
Section 9	1 - 85	
Appendix A	1 - 10	
Appendix B	1 - 12	
Appendix C	1 - 18	
Appendix D	1 - 7	
Index	1 - 11	
User Comment Sheet		

Section	Pages	Update

A vertical bar (|) in the outer margin of the updated pages indicates technical changes.

Contents

1. Introduction

1.1. General

The SPERRY 1100/70 Systems are general purpose, medium to high performance, unit processor, dual processor, or multiprocessor systems incorporating the latest advances in computer design, system organization, and programming technology. Although the 1100/70 Systems differ from other Series 1100 Systems in some respects, software compatibility with the other systems is maintained. The various components of the system are designed as separate logical units providing maximum functional modularity. The multiprocessing capabilities are an integral part of the system. The processing unit can perform numerous tasks simultaneously under the control of a single SPERRY Series 1100 Executive System. The flexible modular structure enables a system to be tailored to fit a user's individual needs.

Three basic processing systems for the 1100/70 are:

- unit processor system (one Central Processing Unit (CPU) and one Input/Output Unit (IOU), 1x1);
- dual processing system (two CPUs and one IOU, 2x1); and
- multiprocessing system (two to four CPUs and two to four IOUs, 2x2, 3x2, 3x3, 4x2, 4x4).

Each processing system has a variety of models available: the unit processor system has Models B1, C1, C2, E1, E2, H1, and H2; the dual processor and multiprocessor systems have Models E1, E2, H1, and H2.

1.2. Scope

This manual provides reference information to be used by the programmer of an 1100/70 System. Detailed information is provided on the CPU, IOU, Storage Interface Unit (SIU), Main Storage Unit (MSU), and System Support Processor (SSP). Section 2 includes a brief description of all the components in the 1100/70 Systems.

2. Systems Description

2.1. General

The SPERRY 1100/70 Systems incorporate the latest advances in design technology and system organization. Like the other members of the Series 1100 family, the 1100/70 Systems provide an efficient multiprocessor capability that allows Central Processing Units (CPUs) to perform multiple tasks simultaneously under the control of a single common Executive System. The 1100/70 Systems are available in three basic modules: the unit processor (1x1), the dual system processor (2x1), and the multiprocessor (2x2, 3x2, 3x3, 4x2, 4x4) configuration. Principal features of the 1100/70 Systems are:

- single or multiple CPUs, Input/Output Units (IOUs), and Main Storage Units (MSUs);
- common access to most system components in multiple CPU/IOU/MSU configurations;
- Input/Output (I/O) operations controlled by the IOU;
- byte- and word-oriented I/O channels;
- large, modular semiconductor main storage (8,388,608 words maximum);
- an optional CPU-associated high-speed buffer storage (2048 or 8192 words per CPU);
- an optional extended instruction set;
- comprehensive error checking and correction facilities;
- an optional hardware performance monitor;
- system component redundancy;
- central complex and optional subsystem partitioning;
- an automated hardware diagnostic/prognostic system;
- extensive storage and component protection facilities for security and servicing;
- partial-, full-, and double-word addressability;

- an optional compatible channel interface feature;
- program address relocatability; and
- a continuously evolving comprehensive Operating System.

2.2. System Components

The system configuration can vary depending on customer needs. The major hardware components of 1100/70 Systems are:

- Central complex cabinet (can contain one CPU, one IOU, one SIU, one MSU and one SC)
- Central Processing Unit (CPU)
- Input/Output Unit (IOU)
- Main Storage Unit (MSU)
- Storage Interface Unit (SIU)
- Support Controller (SC)
- System Support Processor (SSP)/Console
- System Console
- Attached Virtual Processor (AVP)
- Auxiliary storage and peripheral subsystems
- Onsite and remote communications subsystems.

A unit processor contains one central complex cabinet. Dual and multiprocessor systems can contain from two to four central complex cabinets and can be divided into two clusters.

A cluster consists of one or two central complex cabinets. Each cluster has access to the external main storage. The IOUs in a cluster can only access the CPUs in the same cluster.

An SIU is required for each CPU in a dual or multiprocessor system. Systems with one or two central cabinets use main storage located within the central complex cabinet. Systems with three or four central complex cabinets require external main storage.

2.2.1. Central Processing Unit (CPU)

The CPU executes all control and arithmetic functions in the system by a combination of hard-wired logic and firmware. An 1100/70 System may contain one to four CPUs. The CPU contains interfaces to an MSU or SIU, two IOUs, two SSPs, and to other CPUs.

Capabilities of the CPU include:

- 116 nanoseconds basic cycle time;
- 128-word General Register Stack (GRS);
- an instruction/operand prefetch;
- an addressing range of 16 million 36-bit words;
- relative addressing providing program segmentation and storage protection;
- four bank addressing;
- extensive fault detection and instruction retry capability;
- an optional SIU, providing performance improvement and interface to two MSUs;
- an optional extended instruction set feature; and
- an optional performance monitor feature.

2.2.2. Input/Output Unit (IOU)

An 1100/70 System contains one to four IOUs. Under CPU direction, the IOU controls all transfers of data between main storage and peripheral devices. The IOU consists of a Central Control Module (CCM) and up to six channel modules. The CCM provides independent control paths to one or two CPUs (within the same cluster in a dual or multiprocessor system) and one or two SSPs, and data paths to one or two MSUs. Both block multiplexer and Series 1100 word channel modules are included for (I/O) transmissions.

Capabilities of the IOU include:

- a capacity of up to three block multiplexer channel modules and up to three word channel modules, or five block multiplexer channel modules and one word channel module;
- Internally Specified Index (ISI) or Externally Specified Index (ESI) transfer modes on word channels;
- parity generation/checking capability on all word channels operated in ISI mode;
- optional subsystem partitioning features;
- channel transfer rates of
  - 1.6 million bytes (371k words) per second on a block multiplexer channel,
  - 718k words per second (maximum) on a single word channel interface,
  - 1.4 million words per second (maximum input) aggregate for four word channel interfaces (one word channel module),
  - 86k words per second (maximum output) aggregate for four word channel interfaces (one word channel module),
- direct interface to storage;
- channel operation controlled by channel command words; and
- an optional compatible channel interface feature.

2.2.3. Main Storage Unit (MSU)

The 1100/70 System offers two types of main storage.

One type is located in the central complex cabinet, which has a minimum of 524,288 (524K) words, expandable in 524K-word increments to a maximum of 4,194,304 (4194K) words per cabinet. Main storage located in the central complex cabinet can only be used with 1100/71 and 1100/72 Systems (see Table 2-1).

A second type of main storage is located in a free-standing MSU cabinet, which has a minimum of 1,048,576 (1048K) words, expandable in 1048K-word increments to a maximum of 4194K words per cabinet. One or two free-standing MSU cabinets can be used on a system for a maximum of 8,388,604 (8388K) words. The free-standing MSU cabinet is required with the 1100/73 and 1100/74 Systems.

The two types of storage cannot be used on the same system.

The control section of the MSU provides requester (CPU or SIU, IOU, and SSP) priority circuitry.

Capabilities of the MSU include:

- 43-bit word (36 data bits + 7 error correction code bits) internal format;
- 38-bit word (36 data bits + 2 parity bits) format at the requester interface;
- capacity from 524K up to 4194K words per MSU;
- single- or 4-word read, single- and partial-word write;
- single- and double-bit error detection and single-bit error correction;
- application partitioning by requester interface and by 262K-word granularity;
- scan/set interface for diagnostic/prognostic testing;
- an internal refresh;
- internal priority circuitry; and
- internal exerciser for offline operation.

2.2.4. Storage Interface Unit (SIU)

An SIU contains either 2048-word (2K) or 8195-word (8K) of high speed buffer storage and is interposed between the CPU and MSU. The Models B and C Systems do not contain an SIU. The Model E Systems contain a 2K-word SIU and the Model H Systems contain an 8K-word SIU in each central complex cabinet. An SIU connects to both MSUs in a multiprocessor configuration. The SIU provides dedicated service to one CPU and interfaces with one or two MSUs, and one or two SSPs.

Capabilities of the SIU include:

- single-word wide interface to main storage (one storage cycle)
  - four words contained in the data block are transmitted in word-serial fashion from main storage to the SIU (four storage cycles)
  - the requested word is transferred first and only the requested word is transferred to the requester;
- set associative addressing;
- single-port requester interface (i.e., CPU dedicated);
- paired least-recently-used age algorithm;
- error detection and reporting;
- invalidate interface
  - ensures that data changed in one SIU is invalidated in the other SIU so that the new data will be fetched if needed;
  - activity on this interface does not interfere with CPU requests. The MSU will store data for, and communicates with the SIUs and IOUs.

2.2.5. Support Controller

Each central complex cabinet contains a support controller. The support controller provides an interface between the SSP and the individual units in the central complex. Each support controller may interface with one or two SSPs. System control, maintenance, and diagnostic operations are performed on the central complex cabinet by the SSP via the support controller. The support controller provides a means for the SSP to access main storage with block reads and writes. The support controller allows interrupts to be transferred to and from the SSP and the CPU.

Capabilities of the support controller include:

- loading control storage;
- selecting and controlling clock modes;
- scanning and setting flip-flops, register, and storage in the central complex units;
- loading maintenance microdiagnostic programs;
- testing and verifying control storage;
- partitioning central complex units;
- clearing and resetting system components;
- selecting initial load paths;
- verifying initial load operations;
- enabling system functions (quantum timer, dayclock, real-time clock, and breakpoint stop);
- controlling auto recovery; and
- loading parameters (program address, breakpoint address, stop selects, and jump keys).

2.2.6. System Support Processor (SSP)/Console

Each system includes one or two SSPs. The SSP is a desk-sized separate minicomputer unit that interfaces with the central complex units (CPU, IOU, MSU, SIU), which is capable of controlling one to four Cathode-Ray Tube (CRT)/keyboard consoles and a communications interface for remote diagnostic operation.

Capabilities of the SSP include:

- partitioning of central complex units and, optionally, word and byte peripheral subsystems;
- an initiation of system initial load and automatic recovery;
- microcontrol storage loading;
- error analysis;
- maintenance operations, both onsite and remote;
- control of up to four system consoles;
- retrieval of performance data via the optional performance monitor feature; and
- control of logic analyzer.

2.2.7. System Console

The console provides the means for communications between the operator and the Executive System. The console connects to an 1100/70 System via the SSP and consists of a display terminal, a printer, and a desk-like stand.

2.2.8. Attached Virtual Processor

The Attached Virtual Processor (AVP) is a general purpose processor capable of running the Series 90 VS/9 Operating System in an 1100/70 System environment. The AVP is configured on an 1100/70 System as one of the CPUs in a system. The AVP interfaces with CPU, MSU, and SSP units within the system. The I/O operations are handled on the 1100/70 I/O channels.

2.2.9. Peripheral Subsystems

The 1100/70 Systems offer a full range of auxiliary storage, paper peripheral, and communications subsystems. The standard SPERRY subsystems include:

- Semiconductor Storage Family
- 8480/8470/8450/8434/8433/8430 Disk Subsystems
- 8407 Diskette Subsystem
- FH-432/FH-1782 Drum Subsystems
- UNISERVO 22/24 Magnetic Tape Subsystems
- UNISERVO 26/28 Magnetic Tape Subsystem
- UNISERVO 30 Group Magnetic Tape Subsystems
- 0777 Printer Subsystem
- 0776 Printer Subsystem
- 0770 Printer Subsystem
- 0604 Card Punch Subsystem
- 0716 Card Reader Subsystem
- Distributed Communications Processors (DCP)/Telcon System
- General Communications Subsystem (GCS)
- Universal Terminal System (UTS) 4000 Series
- UTS 400 Display Terminal
- UTS 400 Text Editor
- Business Computer/7 (BC/7) Remote Job Entry Terminal
- System 80 (remote)
- V77 Series Systems

Many subsystems used on earlier model SPERRY Series 1100 Systems can be configured, but Sperry will not enhance any existing Series 1100 software for these destandardized subsystems.

2.3. Additional Features

2.3.1. Extended Instruction Set Feature

This optional feature consists of twenty byte-oriented instructions and is used for high level language processing and the operating system. It provides additional data manipulation capabilities, resulting in enhanced system performance (see 9.14).

2.3.2. Performance Monitoring Feature

This optional feature allows an 1100/70 System to collect system profile hardware and software performance data. The sampling of performance data is initiated by software or operator request. The signals are sampled each 475 microseconds and collected by the SSP every 25 seconds. The resulting data is stored in the system log for later report generation. Output reports assist the site manager in doing such tasks as balancing work loads and analyzing long-term trends. This feature operates with negligible overhead. Where more data is desired, the performance data may be combined with data collected by the Software Instrumentation Package (SIP). If this feature is selected a performance monitoring feature is required for each CPU in the system.

2.3.3. Logic Analyzer

The logic analyzer is a standard maintenance tool that provides a means of sampling and recording logic signals at discrete intervals of time. The SSP has control over the starting, stopping, and the sampling rate of logic signal recording. The SSP also has access to the recorded signal data via normal support controller read operations.

2.3.4. Subsystem Partitioning Features

The optional features allow the IOU to partition peripheral subsystems. The word channel feature, which controls Shared Peripheral Interface (SPI) units on word channels, can control SPI interfaces for up to six subsystems. The block multiplexer feature, which controls the Byte Channel Transfer Switch for subsystems connected to a block multiplexer channel, can control up to the maximum Byte Channel Transfer Switch capability of eight subsystems. An IOU can have two such features installed in any combination. Partitioning is accomplished via the SSP under software control.

2.4. System Configuration

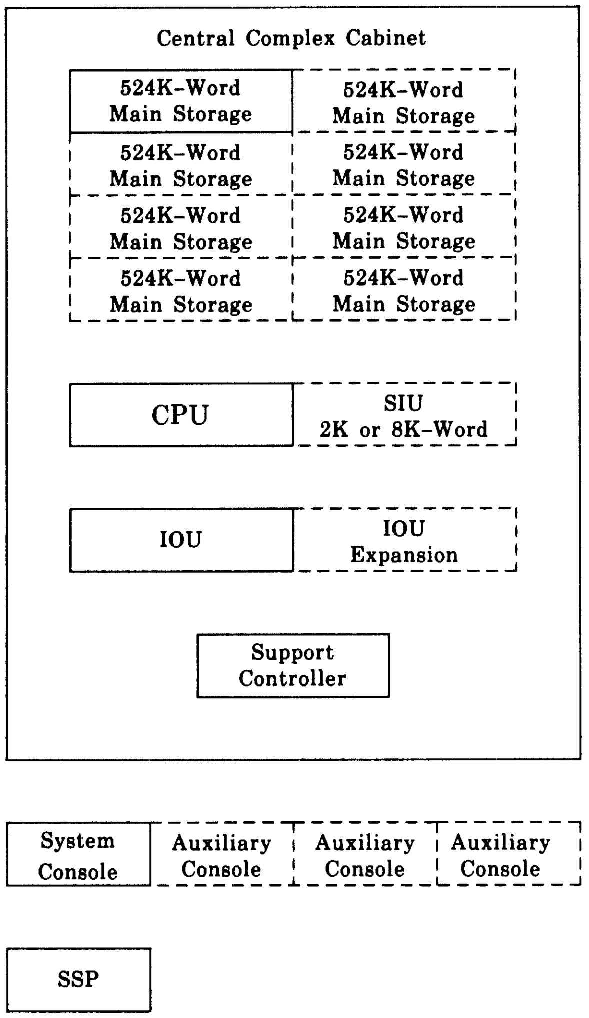
The 1100/70 System has three basic processing systems: the unit processing system, the dual processing system, or the multiprocessing system. There are 23 different models available in the 1100/70 System. The unit processing system has seven models: B1, C1, C2, E1, E2, H1, and H2. The dual processing system has eight models: 2x1 Models E1, E2, H1, and H2; 3x2 Models H1 and H2; 4x2 Models H1 and H2. The multiprocessing system has eight models: 2x2 Models E1, E2, H1, and H2; 3x3 Models H1 and H2; 4x4 Models H1 and H2.

The Model B1, C1, E1, and H1 unit processing systems and the Model E1 and H1 on dual processing and multiprocessing systems use the standard Series 1100 instruction set.

The Model C2, E2, and H2 unit processing systems and Model E2 and H2 on dual processing and multiprocessing systems use the standard Series 1100 instruction set, plus the byte oriented extended instruction set as a standard feature.

2.4.1. Unit Processor Configuration

The unit processing system contains one central complex cabinet. This cabinet houses one CPU, one IOU, one MSU, one SIU (optional), and their applicable expansion features. A unit processor configuration is shown in Figure 2-1. Data paths exist between the MSU and the CPU directly, or via the optional SIU, and between the MSU and IOU directly. Control paths exist between the CPU and IOU, and between the SSP and all central complex units.

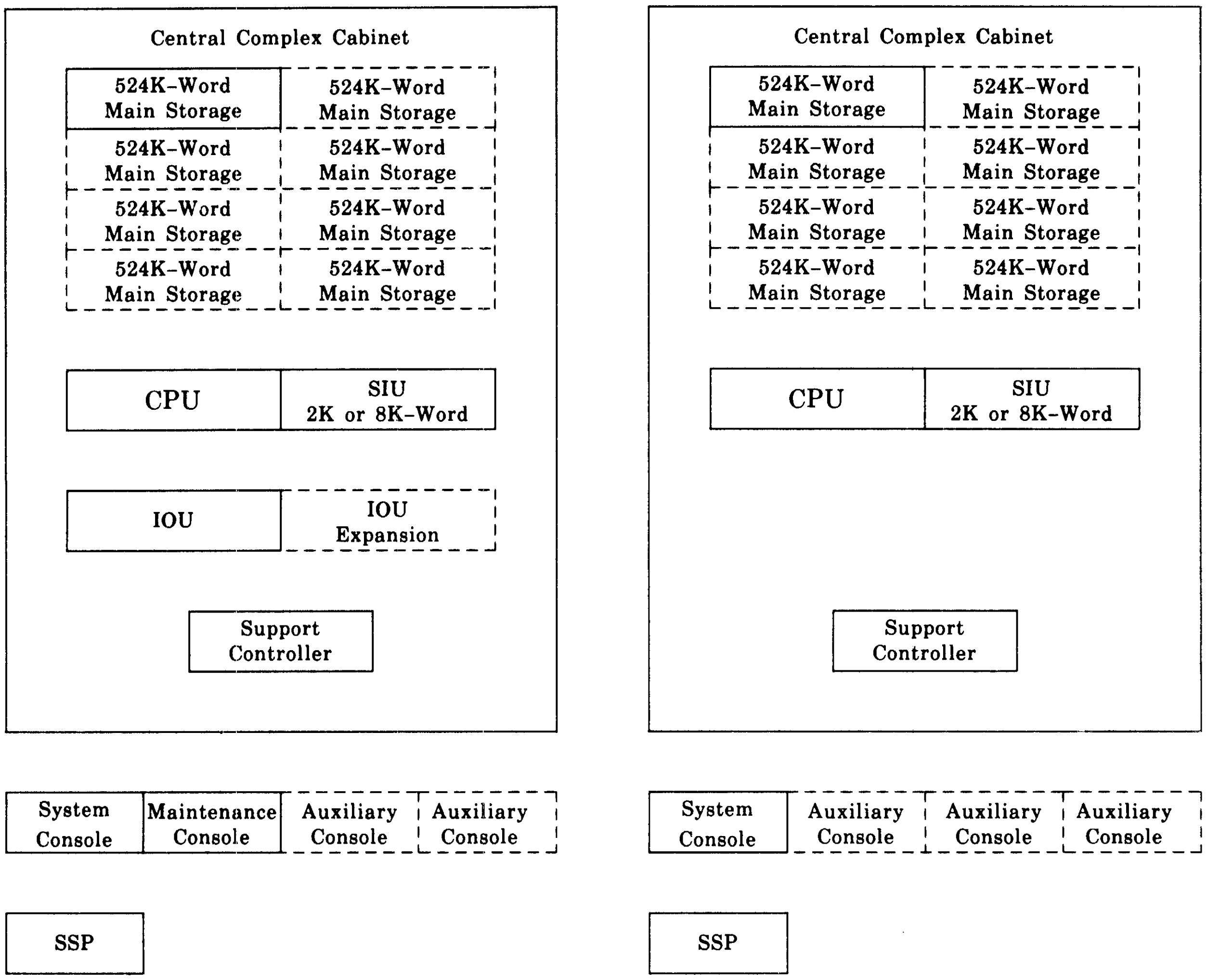


NOTE: Dashed lines indicate optional features.

Figure 2-1. Unit Processing System Configuration 1x1

2.4.2. Dual Processor Configuration

The dual processing system contains two to four central complex cabinets. Each cabinet houses one CPU, one MSU, one SIU, and their applicable expansion features. Only one or two IOUs are used. The dual processing system requires two SSP/consoles plus one console expansion consisting of a maintenance CRT with keyboard. A dual processor (2x1) configuration is shown in Figure 2-2. Data paths exist between an SIU and both MSUs. Control path exist between the CPUs, and between the CPUs and the IOU. Each of the SSPs has control paths to the units of the central complex cabinets. A CPU interfaces only with the SIU in its common cabinet.



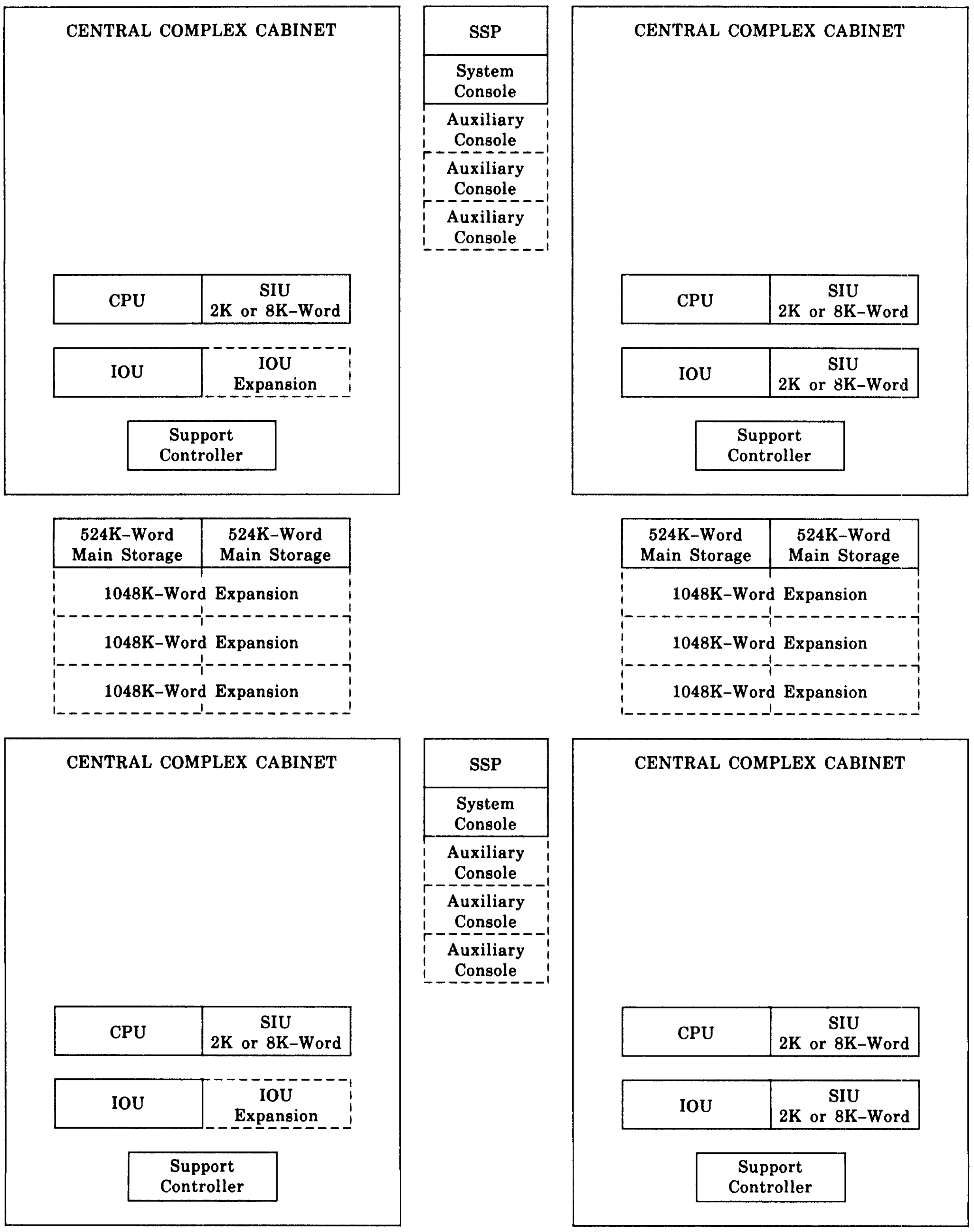
NOTES: 1. *Dashed lines indicate optional features.*

2. *A dual processor system requires two consoles plus one maintenance CRT/keyboard. The maintenance CRT/keyboard connects to one of the auxiliary console interfaces on each SSP.*

Figure 2-2. Dual Processor Configuration 2x1

2.4.3. Multiprocessor Configuration

Multiprocessor systems are configured with two central complex cabinets in a cluster. Each cabinet houses one CPU, one IOU, one MSU, one SIU (required in this configuration), and their applicable expansion features. One or two central complex cabinets may be added to form a second cluster. A CPU in one cluster may communicate only with IOUs in the same cluster. All the CPUs are able to communicate with each other via the interprocessor interrupt mechanism. External main storage is required in 3x3 and 4x4 configurations and is common to both clusters.



NOTES: *1.* *Dashed lines indicate optional features.*

2. *A multiprocessor system requires two consoles plus one maintenance CRT/keyboard. The maintenance CRT/keyboard connects to one of the auxiliary console interfaces on each SSP.*

Figure 2-3. Multiprocessor Configuration (4x4)

Table 2-1. 1100/70 Systems Configuration Component

Configurations	CPU	IOU	SIU	SIU Words	MSU Words	SSP	Auxiliary Consoles*	Extended Instruction Set
Unit Processor Systems								
1x1 Model B1	1	1	0	0	524K to 4194K	1	0–3	no
1x1 Model C1	1	1	0	0	524K to 4194K	1	0–3	no
1x1 Model C2	1	1	0	0	524K to 4194K	1	0–3	yes
1x1 Model E1	1	1	1	2K	524K to 8388K	1	1–3	no
1x1 Model E2	1	1	1	2K	524K to 8388K	1	1–3	yes
1x1 Model H1	1	1	1	8K	524K to 8388K	1	1–3	no
1x1 Model H2	1	1	1	8K	524K to 8388K	1	1–3	yes
Dual Processors Systems								
2x1 Model E1	2	1	2	4K	1048K to 8388K	2	1–6	no
2x1 Model E2	2	1	2	4K	1048K to 8388K	2	1–6	yes
2x1 Model H1	2	1	2	16K	1048K to 8388K	2	1–6	no
2x1 Model H2	2	1	2	16K	1048K to 8388K	2	1–6	yes
Multiprocessor Systems								
2x2 Model E1	2	2	2	4K	1048K to 8388K	2	1–6	no
2x2 Model E2	2	2	2	4K	1048K to 8388K	2	1–6	yes
2x2 Model H1	2	2	2	16K	1048K to 8388K	2	1–6	no
2x2 Model H2	2	2	2	16K	1048K to 8388K	2	1–6	yes
3x2 Model H1	3	2	3	24K	1048K to 8388K	2	1–6	no
3x2 Model H2	3	2	3	24K	1048K to 8388K	2	1–6	yes
3x3 Model H1	3	3	3	24K	1048K to 8388K	2	1–6	no
3x3 Model H2	3	3	3	24K	1048K to 8388K	2	1–6	yes
4x2 Model H1	4	2	4	32K	1048K to 8388K	2	1–6	no
4x2 Model H2	4	2	4	32K	1048K to 8388K	2	1–6	yes
4x4 Model H1	4	4	4	32K	1048K to 8388K	2	1–6	no
4x4 Model H2	4	4	4	32K	1048K to 8388K	2	1–6	yes

* *First auxiliary console is a maintenance console supplied with all Model E and H systems.*

2.4.4. Minimum Peripheral Complement

The following list of peripheral equipment is the minimum for the 1100/70 Systems. This minimum has been established to ensure an adequate complement for Sperry customer engineering and software support.

	Minimum Complement	Alternate
1.	One 0776 Printer Subsystem	One 0770 Printer Subsystem
2.	Disk subsystem with one 5057 control unit and one 8470 Disk Drive	Disk subsystem with one control unit and one 8480, 8450, 8434, 8433 or 8430 Disk Drive

3. Magnetic tape subsystem with one control unit and two UNISERVO 22 or 24 Magnetic Tape Units	Magnetic tape subsystem with one control unit and two UNISERVO 30, 32, 34, or 36 Magnetic Tape Units
4. Telcon System with one DCP/20 or DCP/40.	

3. Central Processing Unit

3.1. General

The Central Processing Unit (CPU) is a microcoded processor that executes the Series 1100 instruction set. The CPU comprises an arithmetic section, control section, general register stack, and interfaces for communicating with other units in the system.

The arithmetic and control sections are discussed here. The General Register Stack (GRS) is discussed in Section 6.

3.2. Arithmetic Section

All arithmetic computation can be performed in either fixed-point or floating-point mode. Fixed-point arithmetic instructions provide for single-precision, double-precision, half-word, and third-word addition and subtraction, and for fraction and integer multiplication and division. Floating-point instructions provide for both single-precision and double-precision operation. The arithmetic section also performs certain logical operations such as shifting and comparisons. Decimal computation of fixed-point numbers is provided. The instruction word may be used to specify the transfer of any chosen portion of a word (half, third, quarter, or sixth) to the arithmetic section. The ability to transfer only the selected portion of a word minimizes the number of masking and shifting operations required.

The shift matrix in the arithmetic section permits the completion of an entire single-word shift operation in one main storage cycle time. By use of the matrix, the shift operation can shift a single- or double-word operand in either direction up to 72-bit positions.

3.2.1. General Operation

During the execution of logical and arithmetic instructions, the following steps are performed:

1. Transfer input data from instruction-word-specified storage locations or general registers to input registers in the arithmetic section. During the transfer, the input data is processed by the main control section to provide absolute values.

2. Perform the arithmetic operations of addition, subtraction (add negative), multiplication, division, skip detection, etc., as specified by the instruction word.

3. Transfer final results from the arithmetic section to temporary holding registers, general register storage, or indicate skip condition.

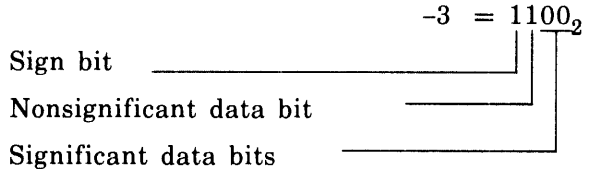
3.2.1.1. Data Word

The highest order binary bit represents the sign of the value contained in the remaining bit positions. If the sign bit contains a 0, the word is positive and 1's in the remaining bit positions represent significant data. If the sign bit contains a 1, the word is negative and 0's in the remaining bit positions represent significant data. A binary data word containing all zeros is referred to as positive zero (+0). A binary data word containing all ones is referred to as negative zero (-0).

Example: (assume a 4-bit word length)

$$+3 = 0011_2$$

$$-3 = 1100_2$$



3.2.1.2. Data Word Complement

The ones complement of any binary arithmetic data word is obtained when all zeros in the word are changed to ones and all the ones are changed to zeros. An arithmetic data word of positive value, when complemented, becomes a negative value; and a negative value, when complemented, becomes a positive value.

3.2.1.3. Absolute Values

The absolute value of an arithmetic number is the magnitude of the number regardless of the sign.

Example:

Binary Value	Absolute Value
001110 (+14)	001110 (14)
110001 (-14)	001110 (14)

3.2.2. Microprogrammed Control

Arithmetic and logic functions are performed in a microprocessor that functions as the main adder. There are two such parallel 36-bit microprocessors in the CPU, fed by a common shifter, and comparison of their results is made each microcycle to provide fault detection. Each Series 1100 macroinstruction is performed by executing a series of microinstructions. Each microinstruction consists of bringing data from GRS or main storage through the shifter into the microprocessors, performing a calculation that combines the data from the shifter with the data from the local stores and the accumulators in each microprocessor, and placing the results in the accumulators. From the accumulator, data may be loaded into main storage or GRS.

When a new instruction is brought into the CPU, it is decoded to select the first microinstruction needed to execute it. Each subsequent microinstruction is chosen by the previous microinstruction until the macroinstruction is completed.

3.2.3. Main Adder Characteristics

The main adder of the CPU arithmetic section performs single- or double-precision adds or subtracts, as well as half-word and third-word addition and subtraction. It also does single- and double-word logical operations.

3.2.4. Fixed-Point Single- or Double-Precision Add or Subtract Overflow and Carry

In fixed-point arithmetic, the execution of certain instructions can result in an overflow or a carry condition. During execution, the Overflow designator (D1) and the Carry designator (D0) bits are cleared to zeros; the overflow and carry conditions set bits D1 and D0, respectively, in the designator register. These bits can be sensed by certain other instructions. Each of these designators, when set to one, remains in the set condition until the next time any one of the instructions in Table 3-1 is executed or until the Load Designator Register instruction is executed.

3.2.4.1. Overflow

An overflow condition is detected when one of the 10 instructions in Table 3-1 is executed, and the numeric value of the result obtained exceeds the maximum numeric value that can be contained in the register holding the final result. This is of significance when the operands for an additive process are of the same sign or when the operands for a subtractive process have different signs. If overflow occurs in these cases, the sign of the result is unnatural. The condition of the D1 can be tested by executing either the Jump Overflow or the Jump No Overflow instruction.

3.2.4.2. Carry

A carry condition is detected when an end-around carry occurs during the execution of an instruction listed in Table 3-1. The detection of a carry condition indicates that a carry was propagated out of the sign bit position and automatically added into the low-order bit position. The detection of the carry condition is significant when programming multiple-precision routines. In ones complement subtractive arithmetic, the carry condition can be equated to the no-borrow condition and the no-carry condition to the borrow condition.

The condition of the carry designator can be tested by executing either the Jump Carry or Jump No Carry instructions. Table 3-2 lists the sign bit combinations for which the designator D0 would be set to 1, indicating that a carry has occurred.

Table 3-1. Instructions that Condition the Carry and Overflow Designators

Function Code (Octal)	Instruction
07,00	Add Decimal
07,01	Double Add Decimal
07,02	Subtract Decimal
07,03	Double Subtract Decimal
14	Add to A
15	Add Negative to A
16	Add Magnitude to A
17	Add Negative Magnitude to A
20	Add Upper
21	Add Negative Upper
24	Add to X
25	Add Negative to X
37,11	Bit Compare
37,13	Bit Compare Long
37,15	Byte to Decimal
37,16	Decimal to Byte
37,17	Edit Decimal
71,10	Double-Precision Fixed-Point Add
71,11	Double-Precision Fixed-Point Add Negative
72,12	Bit Normalize
72,14	Byte to Bit Normalize

Table 3-2. Sign Bit Combinations That Set Carry Designator

Operation	Input Operation Sign		Resultant Sign
	Augend	Addend	
Addition	+	-	+
	-	+	+
	-	-	+
	-	-	-
	Minuend	Subtrahend	
Subtraction (Add Negative)	+	+	+
	-	-	+
	-	+	+
	-	+	-

3.2.4.3. Arithmetic Interrupt

The arithmetic section cannot cause a system interrupt. But when an arithmetic fault occurs, it generates a fault condition signal that allows the control section to set the appropriate designator bit. Other processor conditions in conjunction with those arithmetic fault conditions determine whether or not control generates an interrupt.

3.2.5. Fixed-Point Division

The process of dividing one fixed-point number by another consists of transferring the numbers to the arithmetic section, calculating a quotient and a remainder, transferring the properly signed quotient to a register and, if the remainder is to be saved, transferring the properly signed remainder to another register. All divide operations use the main adder and shifter.

3.2.6. Fixed-Point Multiplication

The arithmetic section contains a high speed multiplier unit to handle multiplications.

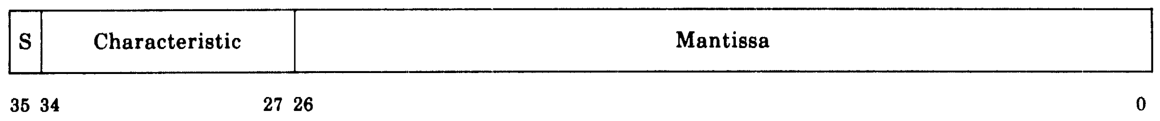
3.2.7. Floating-Point Arithmetic

Floating-point arithmetic handles the scaling problems that arise in computations involving numbers that vary widely in range. In floating-point arithmetic, the numbers are represented in a special format so that the computer can automatically handle the scaling.

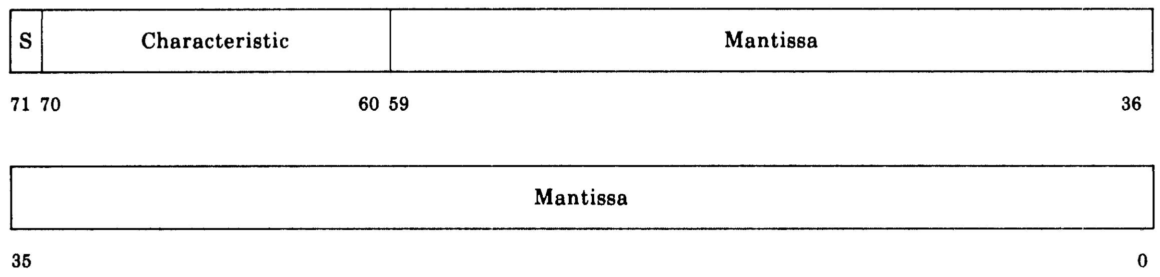
3.2.8. Floating-Point Numbers and Word Formats

Floating-point numbers in the instructions are represented in single-precision format as a 27-bit fractional quantity multiplied by the appropriate power of two, or in the double-precision format as a 60-bit fractional quantity multiplied by the appropriate power of two. The power of two is called the exponent. In machine representation, the exponents are biased to make them lie in the range of positive numbers or zero. These biased exponents are called characteristics. The fractional part is referred to as the mantissa. The two format types, single-precision and double-precision, are as follows:

Single-Precision Floating-Point Format



Double-Precision Floating-Point Format



An explanation of the sign bit, characteristic, and mantissa follows:

- Sign - The sign bit expresses the sign (S) of the numerical quantity represented by the floating-point number.

  If S = 0, the numerical quantity is positive (+).

  If S = 1, the numerical quantity is negative (-).

- Characteristic - The characteristic represents both the numerical value and the sign of the exponent.

  1. Single-Precision Characteristic - The 8-bit characteristic of a single-precision floating-point number represents an exponent value in the range +127 through -128. The characteristic is formed by adding a bias of +128 (200_8) to the exponent. Table 3-3 shows the range of characteristic values and corresponding exponent values.

Table 3-3. Single-Precision Floating-Point Characteristic Values and Exponent Values

Decimal Values		Octal Values	
Characteristic	Unbiased Exponent	Characteristic	Unbiased Exponent
255	+127	377	+177
128	000	200	000
000	-128	000	-200

2. Double-Precision Characteristic – The 11-bit characteristic of a double-precision floating-point number represents an exponent value in the range +1023 through -1024. The characteristic is formed by adding a bias of +1024 (2000_8) to the exponent. Table 3-4 shows the range of characteristic values and the corresponding exponent values.

Table 3-4. Double-Precision Floating-Point Characteristic Values and Exponent Values

Decimal Values		Octal Values	
Characteristic	Unbiased Exponent	Characteristic	Unbiased Exponent
2047	+1023	3777	+1777
1024	0000	2000	0000
0000	-1024	0000	-2000

- Mantissa – The mantissa portion of a floating-point number represents the fractional part of the number. In the instructions, the fractional part is normalized so that the absolute values represented are greater than or equal to 1/2 but less than one. Zero cannot be represented in this range, and it is considered to be normalized as it stands. The binary point of a floating-point number is assumed to lie between the last bit of the characteristic and the first bit of the mantissa. The mantissa of a single-precision floating-point number contains 27 bits; for a double-precision floating-point number, the mantissa contains 60 bits. The mantissa need not be normalized for all instructions.

3.2.8.1. Single-Precision Floating-Point Numbers

A single-precision floating-point number can be derived from a positive decimal number.

Example:

Given number = $+12_{10}$

$+12_{10} = 1100_2 = .1100_2 \times 10_2{}^4$

Sign = + = 0

Characteristic = exponent + bias

$= 00\ 000\ 100_2 + 10\ 000\ 000_2$

$= 10\ 000\ 100_2$

Mantissa $= .110\ 000\ ...\ 000_2$

The format for the floating-point number is as shown (sign included):

Sign	Characteristic	Mantissa
0	10 000 100	1100 0
35	34 27	26 0

$= 204600000000_8$

3.2.8.2. Double-Precision Floating-Point Numbers

A double-precision floating-point number can be derived from a positive decimal number following the same steps that were used for single-precision, with these two exceptions:

- A bias value of 2000_8 is added to the exponent to form the characteristic. For single-precision the value is 200_8.
- The mantissa is 60 bits instead of 27 bits.

3.2.8.3. Negative Floating-Point Numbers

A floating-point number can be derived to represent a given negative number as follows:

- Represent the given number as a positive floating-point number.
- Form the ones complement of the entire positive floating-point number.

Example:

Given number $= -12_{10}$

The single-precision floating-point number for $+12_{10}$ (including sign) is $204\ 600\ 000\ 000_8$.

The single-precision floating-point number for -12_{10} (including sign) is $573\ 177\ 777\ 777_8$.

3.2.8.4. Residue

During single-precision floating-point Add or Add Negative instructions, the bits shifted off the right end of the register during alignment of the mantissas are not included in the addition but are saved and become the residue. After the addition is performed, the sum and the residue are each packed into floating-point format, the sum is stored, and the residue is stored if enable residue store for Single-Precision Floating-Point designator (D17) is 1.

When the two 36-bit input operands for an Add or Add Negative instruction are transferred to the arithmetic section, their characteristics are examined, and the mantissa of the input operand with the smaller characteristic is right-shifted a number of bit positions equal to the difference between the characteristics. The bits shifted out of the 36-bit arithmetic register are saved in an auxiliary register. The portion of the mantissa saved in the auxiliary register is used to form the residue, and it is not included in the algebraic addition. After completion of the addition and any shifting necessary to normalize the sum, the sum and the residue are packed into single-precision floating-point format and transferred to two consecutive A-registers.

3.2.9. Normalized/Unnormalized Floating-Point Numbers

A floating-point number is normalized when the leftmost bit of the mantissa is not identical to the sign bit or when all bits of the mantissa are identical to the sign bit. A floating-point number is not normalized when all bits of the mantissa are not sign bits, and the leftmost bit of the mantissa is identical to the sign bit.

All floating-point operations produce a normalized result when the input operands are normalized. The sums produced by Floating Add and Floating Add Negative instructions and the result produced by the Load and Convert To Floating instruction are always normalized, regardless of whether or not the input operands are normalized. When either or both input operands are not normalized, the result of Add and Add Negative instructions may be less accurate than if normalized input operands had been used.

Normalized input operands must be used for the Floating Multiply, Divide, Compress and Load, and Expand and Load instructions. If normalized input operands are not used for these instructions, the results are undefined.

3.2.10. Floating-Point Characteristic Overflow/Underflow

Floating-point characteristic overflow/underflow occurs when the characteristic does not lie in the range represented in the number of bits allowed for the characteristic.

When any of the Floating-Point Add, Add Negative, Multiply, Divide, or Load and Convert instructions, or the Compress and Load instruction are performed, overflow or underflow may occur.

3.2.10.1. Floating-Point Characteristic Overflow

Single-precision floating-point characteristic overflow occurs when the 8-bit characteristic of the resultant most significant single-precision floating-point word represents a number greater than 377_8, and the associated mantissa is not 0.

Double-precision floating-point characteristic overflow occurs when the 11-bit characteristic of the resultant double-precision floating-point number represents a number greater than 3777_8, and the associated mantissa is not 0.

When overflow is detected, the action taken depends on the Arithmetic Exception Interrupt designator (D20). The Characteristic Overflow designator (D22) is always set.

3.2.10.2. Floating-Point Characteristic Underflow

Single-precision floating-point characteristic underflow occurs when the resultant floating-point word represents a negative number, and the associated mantissa of the result is not 0. This means that the exponent of the result is less than -0200; thus, the attached sign (positive - because absolute value is used) changes due to the borrow. If the characteristic of the residue (Floating Add, Floating Add Negative), remainder (Floating Divide), or the least significant single-precision word of the product (Floating Multiply) represents a negative number, this fact by itself does not result in underflow. Instead, the residue, remainder, or least significant word of the product is cleared to all zero bits or set to all one bits (to reflect the appropriate sign).

Double-precision floating-point characteristic underflow occurs when the 11-bit characteristic of the result represents a negative number, i.e., the exponent of the result is less than 2000_8, the mantissa of the result is not 0, and the Double-Precision Underflow designator (D5) is cleared.

When underflow is detected, the Characteristic Underflow designator (D21) is always set, and the action taken by the CPU depends on the state of designator D20.

3.2.10.3. Floating-Point Divide Fault

For single- or double-precision floating-point division, a divide fault condition will be detected when the mantissa of the divisor is zero. The action taken depends on the Floating-Point Zero Format Selection designator (D8) (for single-precision floating-point division only) and designator D20. The Double Check designator (D23) is always set.

3.2.11. Fixed-Point to Floating-Point Conversion

Conversion of a fixed-point number to floating-point number is performed in the arithmetic section. The first input operand contains a characteristic (biased exponent) that defines the location of the binary point for the fixed-point number with respect to the standard position of the binary point for a floating-point number. The second input operand is the signed fixed-point number to be converted.

The conversion process consists of:

- transferring the two operands to the arithmetic section,
- shifting the fixed-point number, if necessary, to position its bits as the mantissa for a normalized floating-point number,
- modifying the characteristic to reflect the magnitude and direction of the normalizing shift,
- packing the shifted fixed-point number (mantissa) and the modified characteristic in floating-point format, and
- loading the packed results in a register (conversion to single-precision floating-point format) or into two consecutive registers (conversion to double-precision floating-point format).

3.2.12. Floating-Point Addition

The process of adding two floating-point numbers consists of:

- loading the numbers into the arithmetic section,
- determining the difference between the characteristics of the two numbers,
- shifting (right) the mantissa of the number having the smaller characteristic,
- adding the mantissas,
- normalizing the result and correcting the characteristic for the shift,
- combining the results in floating-point format, and
- transferring the resulting floating-point numbers to GRS.

The input operands for floating-point addition need not be normalized numbers. For single-precision addition, the sum (most significant word produced) is always a normalized number. The residue word may or may not be a normalized number. For double-precision addition, the sum is always a normalized number.

3.2.13. Double-Precision Floating-Point Addition

The steps performed for double-precision floating-point addition are similar to those for the single-precision addition with these six differences:

1. Each of the two operands occupy two 36-bit registers in the arithmetic section. In single-precision addition, both operands are contained in two 36-bit registers.
2. The mantissa sum can contain a maximum of 60 bits in double-precision addition, instead of 27 bits as in single-precision addition.
3. The bits that are shifted out of the right end of the 36-bit register when the operands are lined up prior to addition are lost. There is no residue.
4. Double-precision characteristic overflow occurs when the characteristic is greater than 3777_8, and the mantissa is not 0.
5. Double-precision underflow occurs when the exponent is less than -2000_8, and the mantissa is not 0. In single-precision the value is -200_8.
6. The sum is stored in two consecutive registers, A_a and A_{a+1}. No residue is stored.

3.2.14. Floating-Point Subtraction (Add Negative)

Floating-point subtraction (both single-precision and double-precision) uses the same steps as for the Floating-Point Add operation.

3.2.15. Floating-Point Multiplication

The process of multiplying two floating-point numbers consists of loading normalized input operands into the arithmetic section, unpacking, multiplying the mantissas, adding the characteristics, packing the results into floating-point format, and transferring the result to GRS. The results obtained for all cases in which either or both input operands are not normalized numbers are undefined.

3.2.16. Floating-Point Division

The process of dividing one floating-point number by another consists of loading the normalized input operands into the arithmetic section, unpacking, dividing one mantissa by the other, subtracting the characteristics, packing the results into floating-point format, and transferring the result to GRS. The results obtained for all cases in which either or both input operands are not normalized numbers are undefined.

3.2.17. Floating-Point Zero

Floating-point zero can be defined as a floating-point number having all mantissa bits identical to the sign bit. The characteristic sign and all mantissa bits are forced to zeros when this is the result of a floating-point instruction.

3.3. Control Section

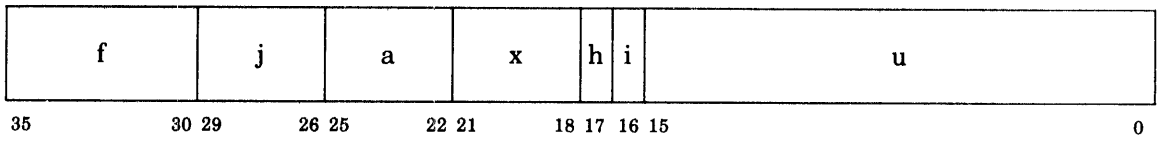
The control section of the CPU interprets instructions and directs all processor operations except certain Input/Output (I/O) operations. The program instruction words are sequentially loaded into the control section. Each instruction word is interpreted by the control section which generates the signals necessary to perform the instruction. The instruction words are located in main storage, and the data words (operands) are located either in main storage or in the addressable control registers, which are part of the control section. The control section includes an address formation segment that generates the absolute main storage addresses to obtain the instruction words.

There are 128 addressable locations in GRS. These control registers are addressed either explicitly or implicitly by the instructions. They fall into five categories: index registers, arithmetic registers, special registers, CPU state control registers, and unassigned registers. The control registers are discussed in Section 6.

The control section includes circuitry that permits the various store instructions to bypass the arithmetic section. This circuitry includes the shifting capability needed for storing partial words in main storage, the sign testing capability needed for the Store Magnitude A instruction, and the complementing capability needed for the Store Negative A and Store Negative Magnitude A instructions.

3.3.1. Instruction Word Format

During the running of a program in the 1100/70 Systems CPU, instructions are transferred from main storage locations to the control section of the CPU. The instructions are transferred from sequentially addressed main storage locations until the sequence is broken by the program or interrupted by the control sections reaction to some special condition or event. Each instruction is a coded directive to the control section; the control section initiates a sequence of steps necessary to perform the particular operation prescribed by the instruction. The 36-bit instruction word, illustrated below, is subdivided into seven fields. These fields specify to the control section the function to be performed, which portion of the operand is to be used, a control register, indexing, index register modification, indirect addressing, and an operand address.



where:

Bits 35–30	The f-field (f) is used as a function code.
Bits 29–26	The j-field (j) is used as an operand qualifier, character address, partial control register address, or minor function code.
Bits 25–22	The a-field (a) is used as an A-, X-, or R-register; jump key or stop keys number; minor function code; or a partial control register address.
Bits 21–18	the x-field (x) is used as an index register.
Bit 17	The h-field (h) is used as an index register incrementation control field.
Bit 16	The i-field (i) is used for indirect addressing control, base register suppression control, 24-bit indexing control, or as an operand base selector.
Bits 15–0	The u-field (u) is used as an operand address or an operand base.

3.3.2. Instruction Word Fields

The following paragraphs describe the manner in which the CPU's control section reacts to the contents of each of the seven fields of an instruction word.

3.3.2.1. Function Code (f-field)

The 6-bit f-field specifies the operation that the CPU is to perform. For function codes 07_8, 37_8, or above 70_8, the f- and j-fields combine to produce a 10-bit function code. When the function code is 05_8, the f- and a-field are combined to produce a 10-bit function code. Further, for some of these function codes, the a-field is also included with the f- and j-fields to form a 14-bit function code. (See Appendix C.) An invalid function code generates an interrupt.

3.3.2.2. Operand Qualifier or Extended Function Code (j-field)

The j-field is an extension to the function code if the value of the f-field is 07_8, 37_8, or greater than 70_8. In all other cases, the j-field represents an operand qualifier as a partial-word specification. As an operand qualifier, the j-field indicates whether the actual operand is the entire 36-bit word specified by the operand address (j = 0); an 18-, 12-, 9-, or 6-bit subfield of the word (j= 01_8 – 15_8); or the operand address itself (j = 16_8–17_8). Figures 3-1 and 3-2 illustrate all the possible data transfer patterns that can be specified by the j-field.

The j-field values of 4, 5, 6, and 7 each have more than one meaning, depending on the value of the Quarter-Word Mode designator (D10). If D10 is 0, these values indicate certain half- and third-word operations. If D10 is 1, j-values of 4 through 7 indicate quarter-word operations.

For store instructions, if the j-field specifies a full word transfer, the word in the register is transferred to the specified location in main storage. If the j-field specifies a partial-word transfer, the partial word is transferred from the least significant bit positions of the register to the j-field specified portion of the storage location; the remaining portion of the storage location is undisturbed. If the j-field value is 16 or 17 for a store instruction, no transfer occurs.

For other instructions, if the j-field specifies a full word transfer (00), the word at the locations specified by the operand address is the actual operand. If the j-field specifies a partial-word transfer (01_8–15_8), the partial operand and the remaining more significant bit positions are filled with 0's or sign bits (depending on the j-field value) forming a full word (36-bit) operand. When the value in the j-field specifies sign extension, the remaining more significant bit positions of the operand are filled with bits that are identical to the most significant bit of the partial word. When the operand address (U) is less than 0200_8, j-field values of 01_8–15_8 are ignored, and a full 36-bit operand is formed. An operand address less than 0200_8 specifies the GRS location.

j (Octal)	Quarter-Word Designator*	Storage Location	⇨	Arithmetic Register
0	0 or 1	35 0	36	35 0
1	0 or 1	17 0	18	35 zeros 18 | 17 0
2	0 or 1	35 18	18	35 zeros 18 | 17 0
3	0 or 1	17 0	18	35 signs 18 | 17 0
4	0	35 18	18	35 signs 18 | 17 0
5	0	11 0	12	35 signs 12 | 11 0
6	0	23 12	12	35 signs 12 | 11 0
7	0	35 24	12	35 signs 12 | 11 0
4	1	26 18	9	35 zeros 9 | 8 0
5	1	8 0	9	35 zeros 9 | 8 0
6	1	17 9	9	35 zeros 9 | 8 0
7	1	35 27	9	35 zeros 9 | 8 0
10	0 or 1	5 0	6	35 zeros 6 | 5 0
11	0 or 1	11 6	6	35 zeros 6 | 5 0
12	0 or 1	17 12	6	35 zeros 6 | 5 0
13	0 or 1	23 18	6	35 zeros 6 | 5 0
14	0 or 1	29 24	6	35 zeros 6 | 5 0
15	0 or 1	35 30	6	35 zeros 6 | 5 0
16**	0 or 1	17 0	18	35 zeros 18 | 17 0
17**	0 or 1	17 0	18	35 signs 18 | 17 0

** The Quarter-Word designator (D10) is held in the designator register.*

*** If $x = 0$, then h, i, and u are transferred. If $x \neq 0$, then $u + (X_m)$ is transferred.*

Figure 3-1. Data Transfers from Storage

j (Octal)	Quarter-Word Designator*	Arithmetic Register	⇨	Storage Location
0	0 or 1	35 0	36	35 0
1	0 or 1	17 0	18	17 0
2	0 or 1	17 0	18	35 18
3	0 or 1	17 0	18	17 0
4	0	17 0	18	35 18
5	0	11 0	12	11 0
6	0	11 0	12	23 12
7	0	11 0	12	35 24
4	1	8 0	9	26 18
5	1	8 0	9	8 0
6	1	8 0	9	17 9
7	1	8 0	9	35 27
10	0 or 1	5 0	6	5 0
11	0 or 1	5 0	6	11 6
12	0 or 1	5 0	6	17 12
13	0 or 1	5 0	6	23 18
14	0 or 1	5 0	6	29 24
15	0 or 1	5 0	6	35 30
16	0 or 1	No Data Transfer		
17	0 or 1	No Data Transfer		

* *The Quarter-Word designator (D10) is held in the designator register.*

Figure 3-2. Data Transfers to Storage

3.3.2.3. Register Operand Address (a-field)

For the majority of instructions, the a-field of the instruction contains an A-, X-, or R-register address for use as an operand. The GRS Selection designator (D6) selects whether the register is selected from the user or executive register set. In some instructions the value in the a-field references two or three A_a-registers. When two A_a-registers are referenced, the value in the a-field explicitly references the register A, and implicitly references the register A_{a+1}. When three A-registers are referenced, the value in the a-field explicitly references the A-register and implicitly references A_{a+1} and A_{a+2}.

In the Jump on Keys instruction (74,04), each value in the range 01_8, through 17_8 in the a-field references one of the 15 selective jump keys, respectively.

In the Halt on Keys and Jump instruction (74,05), each of the 4 bit-positions in the a-field references one of four selective stop keys, respectively.

In the Jump on Greater and Decrement instruction ($f = 70_8$), the value in the j-field and a-field combine to form a single numeric value. This value specifies which one of the 128 addressable control registers is used as the counter in the execution of the instruction.

If the value of the f- and j-fields of the instruction are 05_8; 07_8, 14-15; or 74_8, 14-15; the a-field of the instruction is an extension to the function code.

3.3.2.4. Index Register Address (x-field)

The x-field of the instruction contains an index register address. If the value of the x-field is not 0, the contents of the lower field of the index register (X_m) is added to the value in the u-field to form the operand address or operand ($j = 16_8$-17_8). If the value of the x-field is 0, no indexing occurs.

3.3.2.5. Index Incrementation Designator (h-field)

If the x-field of the instruction contains a non-zero value, the contents of the h-field controls the modification of the x-field addressed X-register. (The a-field addressed X-register can be modified by the subsequent definitions of a BT instruction.)

During the execution of an instruction in which the value in the x-field is not 0 and the value in the h-field is 1 and when the indexing operation is complete, the upper field of the address X-register (X_i) is added to the lower field of the same X-register (X_m), and the sum is stored back in the lower field (X_m). If X_i is positive (bit position 35 = 0), X_m is increased. If X_i is negative (bit position 35 = 1), X_m is decreased. This modification of X_m is performed without affecting the value of the first register operand (A_a, X_a, or R_a), but it can affect the value of a second register operand (A_{a+1}) if one is required by a two-pass instruction that processes both U and U + 1 (for example: DS, DFA, DFAN, DFM, DFD, DA, DAN, and DTE).

If the value in the h-field is 1 and the value of the contents of the x-field is 0, no indexing and no index register modification occurs.

3.3.2.6. Indirect Addressing Designator (i-field)

The i-field of the instruction is used to indicate that one of several modifications to the addressing process is to occur, depending on the value of the Relocation and Storage Suppression designator (D7).

If D7 is 0, the i-field of the instruction indicates indirect addressing. In this case, the location specified by the operand address (indexed u-field) contains a new operand address specification (x-field, h-field, i-field, and u-field) that effectively replaces the current one; if the new i-field value is 1, the indirect addressing operation is repeated. Each time indirect addressing occurs, indexing and index incrementation are performed on the respective operand address and index register if the current value of the h-field indicates index incrementation. Indirect addressing continues (cascades) as long as the new i-field values are one. During cascaded indirect operations, only Guard Mode and Immediate Storage Check interrupts are allowed. All other interrupts and real-time clock updates are locked out for the duration of the cascaded indirect operation. An indirect operand address value of less than 0200_8 specifies a storage location, not a GRS location. If the j-field of an instruction specifies an immediate operand (16_8 or 17_8), and the x-field value is 0, indirect addressing does not occur, but the h- and i-fields are combined with the u-field to form an 18-bit operand; if the x-field value is not 0, indirect addressing proceeds as described above.

If D7 is 1, the i-field of the instruction indicates absolute addressing. In this case the addition of the base value is suppressed, and the relative operand address is also the absolute operand address. D7 is used in conjunction with the i-field to specify the size of the index used to form the operand address.

- If D7 is 1, the i-field is 1, and an immediate operand is not specified ($j \neq 16_8$ or 17_8), a 24-bit index value is used.
- If D7 is 1, the i-field is 1 and an immediate operand is specified ($j = 16_8$ or 17_8), the lower 18 bits of the index register are used to form the operand and, if index incrementation is specified, the upper 12 bits of the index register are added to the lower 24 bits.
- In all other cases, the conventional 18-bit index value is used.

3.3.2.7. Operand Address Displacement (u-field)

The u-field of the instruction contains a displacement value that is used to form an operand address, immediate operand, shift count, or constant value.

When the u-field is used to form an operand address, and the value of the x-field is 0, the value in the u-field is the relative address of the operand. If the value in the x-field is not 0, the value in the u-field is indexed by the contents of the lower field (X_m) of the index register. The sum that results from the indexing operation is identified as U and is the relative address of the operand ($u + X_m = U$). When the content of the u-field is not indexed, U is equal to the u-field value. When U is 200_8 or greater, the address location is in storage. When U is less than 200_8, whether relative or absolute addressing is in effect, the addressed location is in GRS, except when indirect addressing and when the operand address is the address of an instruction (jump or execute). In these cases, the addressed location is always in storage even if U is less than 200_8.

For most instructions, U explicitly specifies the location of a single-word (36-bit) operand. For certain other instructions, U explicitly specifies the location of the most significant 36 bits of a double-word (72-bit); the location of the least significant 36 bits of the double-word operand (U + 1) is implied by U. Note that if the j-field specifies an immediate operand, an 18-bit index register value is always used for the formation of the final immediate operand value (which excludes intermediate indirect addressing), regardless of the addressing mode selected by D7 and the i-field.

If the j-field specifies an immediate operand (16_8 or 17_8), and the x-field value is 0, the h- and i-fields are combined with the 16-bit u-field to extend it to an 18-bit value. In all other cases, the u-field is extended to 18 bits with leading 0's before being used to form an operand address or operand. Because the operand address formation is a ones complement process, an operand address or immediate operand of all ones (0777777) cannot be produced; a potential result of all ones is changed to zero (000000).

If the j-field specifies an immediate operand (16_8 or 17_8), an 18-bit value using the x-field, h-field, i-field, and u-field is extended to 36 bits with leading 0's (if $j = 16_8$), or leading sign bits (if $j = 17_8$, to produce the operand. The 18-bit value is formed in one of two basic ways: if the x-field value is 0, the h-field, i-field, and u-field (the lower 18-bits of the instruction), is the 18-bit value; indexing (x-field), automatic index incrementation (h-field), and indirect addressing (i-field), do not occur in this case. If the x-field is not 0, the final operand address produced by indexing the u-field value and performing indirect addressing (if specified) is the 18-bit value used to form the immediate operand.

4. Input/Output Unit

4.1. General

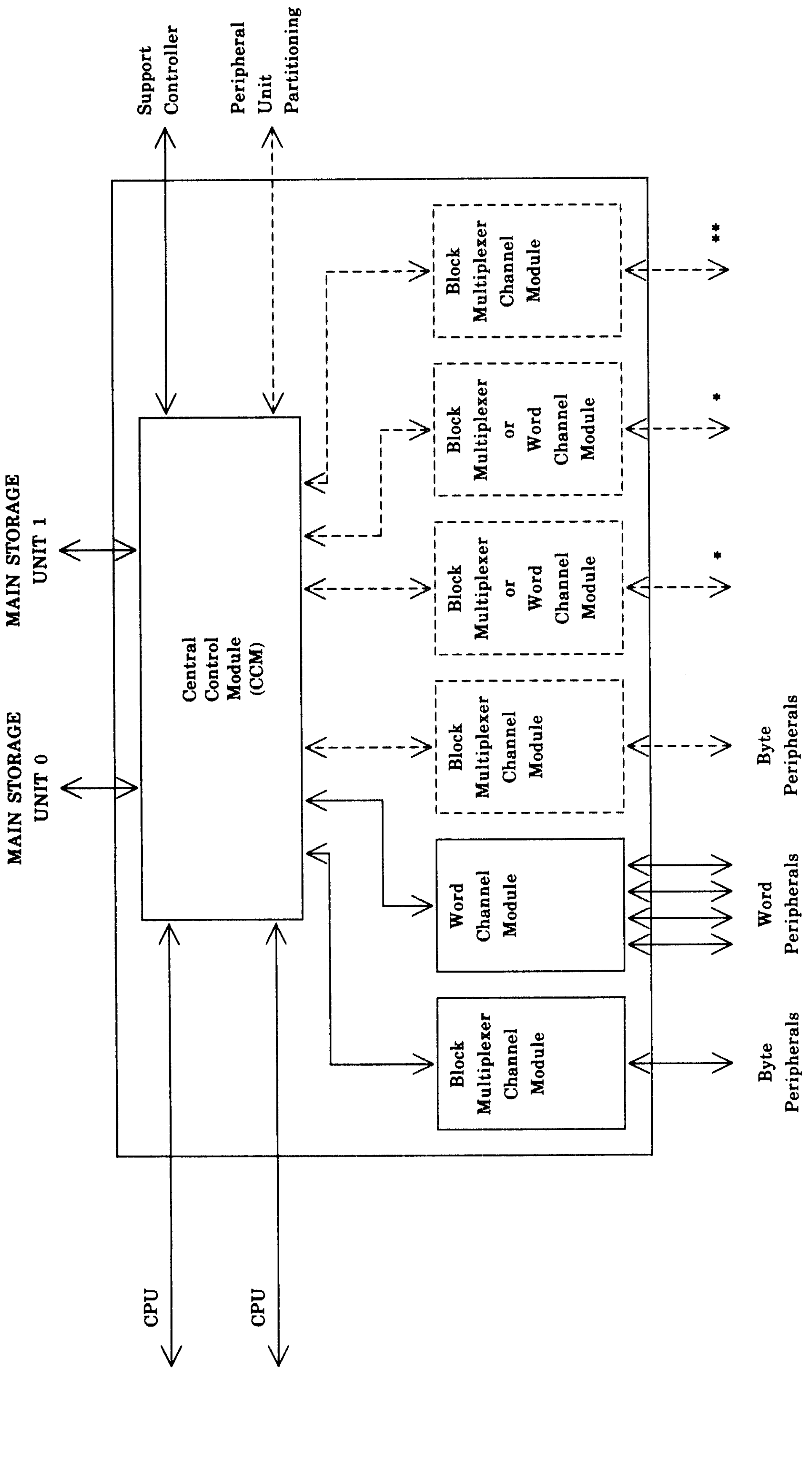
The Input/Output Unit (IOU) provides a means of communications between the Central Processing Unit (CPU) and the external media of the system. Under CPU control, the IOU handles all transfer of data and status between peripherals and main storage. It minimizes CPU involvement in Input/Output (I/O) operations, yet provides a flexible method of controlling and interrogating I/O activity. The IOU section describes the system philosophy, functional characteristics, various hardware and software options, and overall operation.

4.2. Functional Characteristics

The IOU has four interfaces: a Main Storage Unit (MSU) interface, a CPU interface, a support controller interface, and a peripheral interface. An IOU can only communicate with CPUs in the same cluster. The basic IOU consists of one Central Control Module (CCM) and two channel modules. (See Figure 4-1.) The CCM handles the interface with one or two CPUs and one or two MSUs. The CCM to CPU interface initiates instructions and handles interrupts. The CCM to MSU interface transfers data, I/O control words, and status between the channel modules and the MSU interface. The CCM establishes a data handling and interrupt priority among the channel modules.

Available IOU features include subsystem partitioning features, I/O expansion features and the compatible channel interface feature. Subsystem partitioning features include word subsystem partitioning and byte subsystem partitioning, as described in 2.3.4. I/O expansion features provide the expansion capabilities for one additional block multiplexer channel module and two word channel modules, or two block multiplexer channel modules and one word channel module, or four block multiplexer channels. (A word channel module provides four word channel interfaces.)

On block multiplexer channel module operations, a Command Retry is not supported unless the compatible channel interface feature is installed. See *SPERRY Series 1100 8450/5040 Disk Subsystem, Programmer Reference,* UP 9383 for more information on Command Retry.


MAIN STORAGE UNIT 0
MAIN STORAGE UNIT 1
Support Controller
Peripheral Unit Partitioning
Central Control Module (CCM)
Block Multiplexer Channel Module
Word Channel Module
Block Multiplexer Channel Module
Block Multiplexer or Word Channel Module
Block Multiplexer or Word Channel Module
Block Multiplexer Channel Module
Byte Peripherals
Word Peripherals
Byte Peripherals
*
*
**
CPU
CPU

* Either one block multiplexer channel interface or four word channel interfaces.

** Not available when IOU expansion does not consist of four block multiplexer channels.

- - - Dashed lines indicate optional units.

Figure 4-1. 1100/70 Input/Output Unit

4.2.1. Channels

The channel handles all interfacing with the control units. The channel formats and transfers data, generates interrupts and status, and establishes priority among data transfers and interrupts. Each channel is designated to be either a block multiplexer channel or a Series 1100 word channel.

A channel provides a standard interface for communicating with control units. A control unit provides the logical capability necessary to adapt the standard form of control provided by the channel to the characteristics of an I/O device. A control unit may be housed separately and connected to one or many devices, or it may be physically and logically integrated with an I/O device. I/O devices provide external storage and a means of communications for a processing system. Magnetic tape units, printers, storage devices such as disks and drums, consoles, and card readers are examples of I/O devices.

Priority among devices is established by the control units. Priority among control units is determined by the logical connection to the channel. Each word channel has a maximum of four parallel I/O interfaces, each interface is the equivalent of a single Series 1100 word channel on earlier model systems, each with an assigned priority. (See Figure 4-2.) Each block multiplexer channel has one I/O interface, and up to eight control units can be connected to the interface, but only one control unit at a time is logically connected to the channel. The channel polls the control units serially, and the highest priority control unit requiring service logically connects to the channel. A block multiplexer channel connects to a control unit for the length of time to transfer a block of data. No other device can communicate over the interface during the time a block is being transferred.

4.2.2. Subchannels

A subchannel is a set of control words that manages I/O operations. Each set of control words contains a data address, a data count, the mode of the subchannel, the storage address of the next control word, and special flags. Subchannels may be either shared or nonshared. A subchannel is referred to as shared if two or more devices use the same subchannel for I/O operations. On a shared subchannel only one device at a time can transfer data. A subchannel is referred to as nonshared if it is associated with and can only be used with a single I/O device. On a word channel, Internally Specified Index (ISI) subchannels are shared and Externally Specified Index (ESI) subchannels are nonshared.

4.3. Control of Input/Output Devices

The CPU controls I/O operations by means of five I/O instructions: Start I/O Fast release (SIOF), Test SubChannel (TSC), Halt DeVice (HDV), Halt CHannel (HCH), and Load Channel Register (LCR). The Load Channel Register instruction addresses the control module. The Halt Channel instruction addresses only a channel. All other instructions address a channel and subchannel.

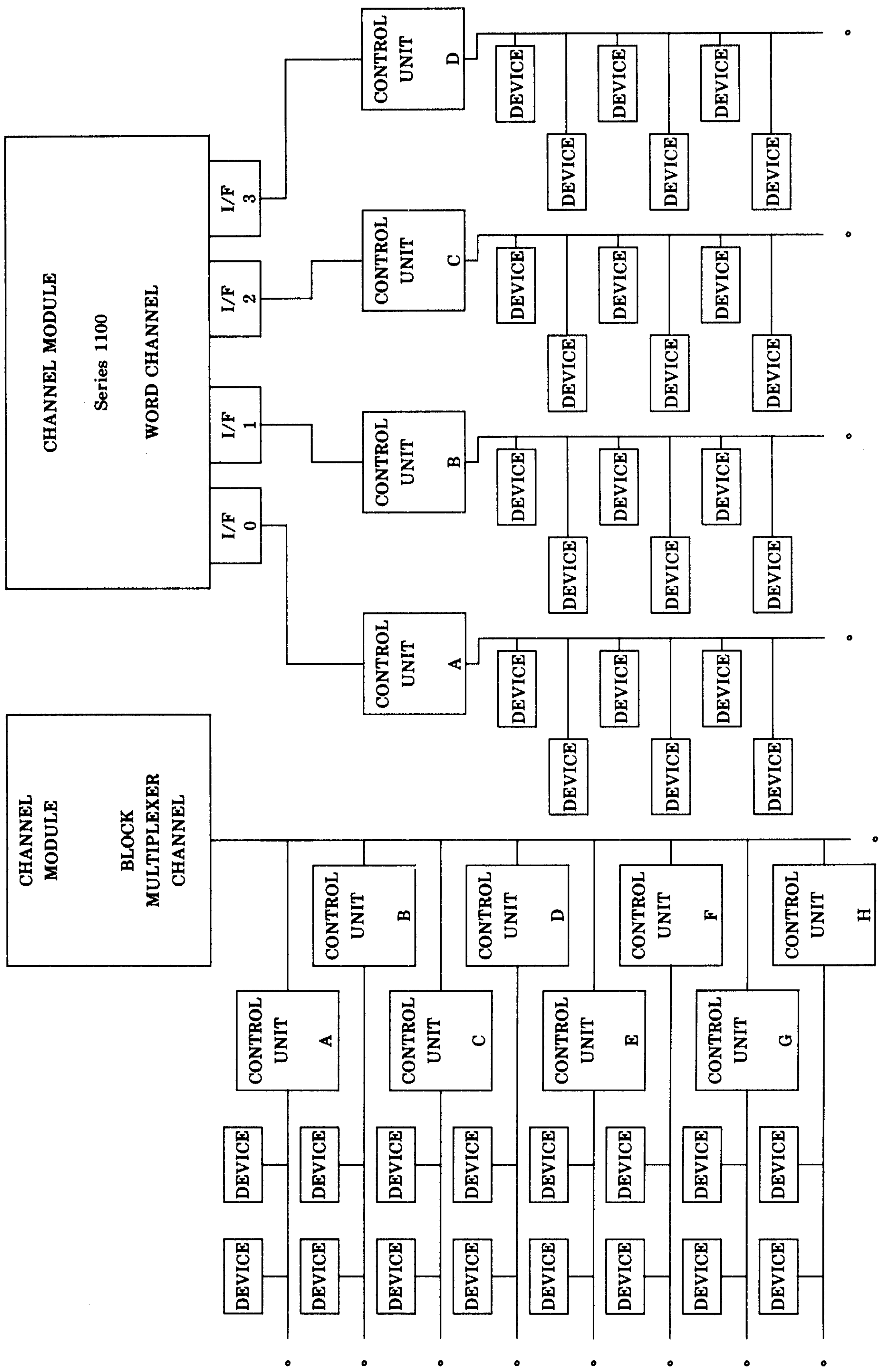


Figure 4-2. Block Multiplexer Channel Compared to Word Channel

4.3.1. Input/Output Device Addressing

Device addressing within the IOU is done on a virtual-to-real basis. The IOU reads the virtual device address from bits 9-0 (subchannel address) of Channel Address Word (CAW1). This 10-bit address references the channel descriptor stack that contains the channel module type and number, the interface number (word only), and the subchannel mode. Once the channel module number has been decoded, the 10-bit address is transferred to that channel module for further decoding and device initiation.

The following table depicts subchannel address assignments:

Bit										
9	8	7	6	5	4	3	2	1	0	Subchannel Type
X	X	X	D	D	D	D	D	D	D	Block Multiplexer
X	X	X	X	E	E	E	E	E	E	Word ESI
1	1	1	1	1	X	X	X	IF	IF	Word ISI
1	1	1	1	1	1	1	1	1	1	Status Table Subchannel

where:

X	No restriction. Specified by System Support Processor (SSP) during system initial load as a function of the system configuration.
IF	Interface. Specifies one of four interfaces on a word channel.
E	ESI identifier bits presented by peripheral.
D	Device address bits presented by control unit or device.

On a word channel module up to four ISI word interfaces are provided. Bits 04-02 of the subchannel address are specified on a word channel module basis by the system support processor during system initial load. Bits 01-00 of the subchannel address specify one of four ISI interfaces on a word channel module. For ESI interfaces the low order 6 bits of the ESI identifier field presented by the peripheral are used directly as a subchannel address. The four remaining bits are specified by the SSP during system initial load. The status table subchannel address is all ones, 1777_8.

On a block multiplexer channel module bits 09-07 of the subchannel address are specified by the system support processor during initial load. The 7-bit device address presented by control unit or device is used directly as bits 06-00 of the subchannel address. Extreme care must be taken when assigning subchannel addresses to guarantee that no two devices on an IOU are being driven by the same subchannel.

The channel descriptor stack is loaded by the SSP during system initialization. Each of the 1024 subchannel addresses reference an address in the channel descriptor stack. The channel descriptor stack has the following format:

P 2	P 1	E S I	W I N	C M T	C M N	M
12	11	10	9 8	7 6	5 3	2 0

where:

Bit 12	P2 is used to maintain odd parity on bits 10–03.
Bit 11	P1 is used to maintain odd parity on bits 02–00.
Bit 10	ESI specifies that the ESI subchannel is operating in 18-bit mode. The absence of bit 10 specifies 9-bit mode.
Bits 09–08	Word Interface Number (WIN) specifies which of the 4-word interfaces is being referenced.
Bits 07–06	Channel Module Type (CMT) specifies the type of channel module.

00	ISI Word Module
01	ESI Word Module
10	Block Multiplexer
11	Not available

Bits 05–03	Channel Module Number (CMN) specifies the actual number (04–00) of the channel module.
Bits 02–00	Mode (M) specifies the mode of the addressed subchannel.

4.3.2. States of the Input/Output System

The result of an I/O instruction is determined by the collective states of the channel and subchannel selected by the I/O address. Depending on the type of channel and the I/O instruction being executed, different combinations of the states of the channel and subchannel will be interrogated to determine the response to an I/O instruction. When the response to an I/O instruction is determined by the state of the channel, the subchannel is not interrogated to determine an I/O instruction response.

4.3.3. Condition Codes

The result of an I/O instruction is reported by a 3-bit condition code. The condition code is stored by the CPU in bits 33–35 of the X_a-register at the time the execution of the instruction is completed. The condition code is determined by the composite state of the I/O system selected by the I/O instruction. The condition codes applicable to each of the I/O instructions are given in 4.4.

4.3.4. Instruction Format and Channel Address Word

All I/O instructions have the f-field = 75_8. The j-field specifies the particular I/O instruction to be initiated. If:

j is 01_8 – Start I/O Fast release (SIOF)
j is 03_8 – Test SubChannel (TSC)
j is 04_8 – Halt DeVice (HDV)
j is 05_8 – Halt CHannel (HCH)
j is 10_8 – Load Channel Register (LCR)

Bits 11–0 of u + X_m specify the I/O address (IOU number and subchannel address). Bits 23–0 of X_a consist of either the starting address of the CCW list or the register input data for the LCR instruction. Upon detection of an I/O instruction, the CPU builds a CAW in a fixed location of low-order storage. The CAW is for hardware use only and consists of two 36-bit words, CAW 1 and CAW 2. The j-value is stored in bits 29–26 of CAW 1. Bits 11–0 of u + X_m (the I/O address) are stored in bits 11–0 of CAW 1. Bits 23–0 of X_a (the first CCW address or register input data) are stored in bits 23–0 of CAW 2. The IOU refers to the CAW only during the execution of an I/O instruction. The pertinent information thereafter is stored in the channel. Formats of the I/O instruction word and the CAWs follow. CAW 2 has several formats, depending on the I/O instruction being performed.

I/O Instruction Word Format:

f = 75	j	a	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 0

where:

Bits 35–30	The f-field specifies function code 75.
Bits 29–26	The j-field (j) specifies I/O instruction.
Bits 25–22	The a-field (a) specifies the address of register holding the first CCW address.
Bits 21–18	The x-field (x) is u + X_m bits 11–0 equal I/O address.

Channel Address Word 1 Format:

Not Used	j-field	Not Used	IOU No.	Subchannel Address
35 30	29 26	25 12	11 10	9 0

where:

Bits 35–30	Not used.
Bits 29–26	The j-field specifies the I/O instruction.
Bits 25–12	Not used.
Bits 11–10	Specifies the IOU number.
Bits 9–0	Specifies the subchannel address.

Channel Address Word 2 Format for SIOF Instruction:

Not Used	Address of First CCW
35 24	23 0

where:

Bits 35–24	Not used.
Bits 23–0	Specifies the double-word address storage location that contains the first CCW in the channel program if the instruction is an SIOF instruction.

Channel Address Word 2 Format for TSC Instruction:

Not Used	Address for Subchannel Snapshot
35 24	23 0

where:

Bits 35–24	Not used.
Bits 23–0	Specifies the starting absolute address location for the storing of the snapshot of a subchannel's control words.

Channel Address Word 2 Format for LCR Instructions:

Not Used	Mask Register Data
35 6	5 0

where:

Bits 35–6	Not used.
Bits 5–0	Contains the data to be loaded into the mask register.

During the execution of an I/O instruction, the CPU stores a CAW in reserved storage. The associated CCW list is stored at an arbitrary location and is pointed to by the CAW. The CCW list is stored prior to the execution of the I/O instruction by the CPU. The CPU activates a line in the CPU to IOU interface to signal the IOU to fetch and handle the CAW. The IOU accepts the signal and reads the CAW from the reserved location. The IOU decodes the CAW to determine the operation to be executed.

The IOU then executes the instruction specified by bits 29 through 26 of CAW 1. Depending on the I/O instruction, the IOU uses the state of the channel, or the states of the channel and subchannel to determine the condition code. When the IOU has successfully initiated the I/O instruction, the processor receives an acknowledge along with a condition code. The CPU then stores the condition code in bits 35–33 of X_a and clears bits 32–30. Bits 29–0 of X_a are left unchanged. If the condition code equals 0, the CPU skips the next instruction in the program. If the condition code equals 1 through 7, the CPU executes the next instruction.

The IOU's execution of an I/O instruction is determined by the state of the subchannel specified by the I/O instruction. A subchannel has three states:

1. Idle
2. Busy
3. Status Pending

A subchannel is placed in the idle state by:

1. being relieved of status, or
2. a successful Halt Device or Halt Channel instruction when in the busy state.

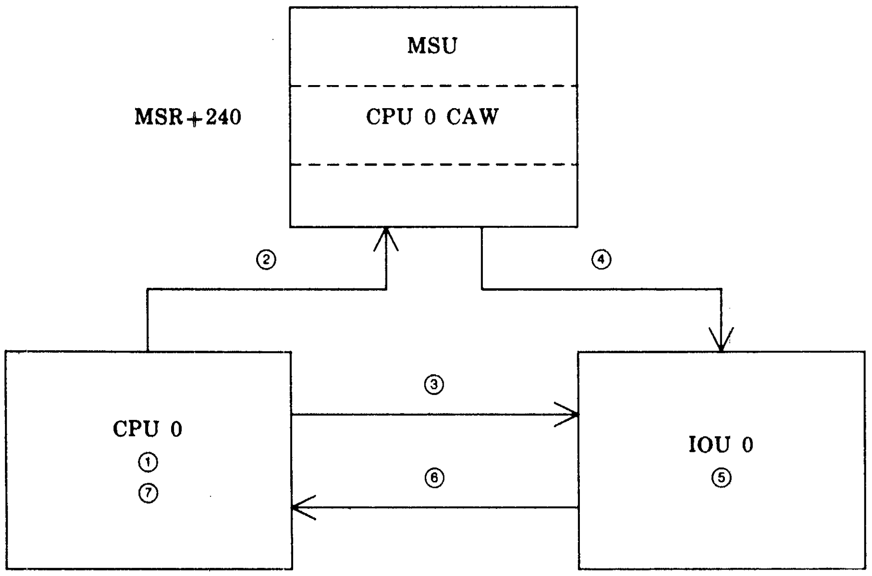
A subchannel is placed in the busy state by either:

1. a Start I/O Fast Release instruction when in the idle state, or
2. an initial load operation initiated by the System Support Processor (SSP).

A subchannel is placed in the status pending state by:

1. detection of a hardware fault,
2. detection of a software fault, or
3. presentation of status by a control unit or device.

The execution of I/O instructions and the associated effects of subchannel states are described in the following subsections. A simplified block diagram of the I/O instruction execution is shown in Figure 4-3.



NOTES: ① *CPU decodes I/O instruction.*
② *CPU stores CAW in MSU.*
③ *CPU interrupts IOU.*
④ *IOU reads CAW.*
⑤ *IOU initiates I/O instruction.*
⑥ *IOU acknowledges CPU request and passes condition code.*
⑦ *If condition code equals 0, CPU skips next instruction; if condition code does not equal 0 CPU takes next instruction.*

Figure 4-3. I/O Instruction Execution Simplified Block Diagram

4.3.4.1. Start I/O Fast Release - SIOF f=75, j=01

This instruction is executed by the CPU that generates the CAW into reserved storage locations. The IOU is signalled to read the CAW by the CPU to IOU interface. The IOU determines which instruction is to be executed by decoding bits 26 through 29 of CAW 1.

The Start I/O Fast Release instruction initiates the execution of a subchannel program on the IOU and subchannel specified by bits 10–0 of the first word of the CAW. Bits 23–0 of the second word of the CAW specifies the starting absolute address of the subchannel program. The results of the SIOF instruction are reported by way of the 3-bit condition code.

If an IOU receives an SIOF instruction and no hardware or software fault is detected, the address of the first CCW is stored in the subchannel, and the subchannel is placed in the busy state. The subchannel address is placed on a first in/first out SIOF stack, and the execution of the subchannel's CCW list is initiated when the associated channel module becomes available. If the SIOF stack is full, the instruction is rejected.

If the SIOF instruction is successfully initiated, the CPU skips the instruction following the SIOF instruction. If a hardware or software fault is detected during the initiation of a SIOF instruction, the instruction is aborted and a 3-bit condition code uniquely identifying the fault is presented to the software.

These condition codes are:

- Condition Code = 0

  The SIOF instruction has been successfully initiated. (Subchannel address has been placed in the SIOF stack.)

- Condition Code = 1

  The selected IOU is not in the application. This is detected by the CPU through a check of the partitioning information for the CPU to the referenced IOU interface.

- Condition Code = 2

  A storage fault has been detected by the IOU during the fetch of the CAW.

- Condition Code = 3

  The addressed subchannel is not in the application. This is detected by the IOU when the channel descriptor table entry for the referenced subchannel is not a block multiplexer or word channel module.

- Condition Code = 4

  The IOU detected an internal hardware fault.

- Condition Code = 5

  The addressed subchannel is presenting status by way of interrupt.

- Condition Code = 6

  The addressed subchannel is in the busy state.

- Condition Code = 7

  The SIOF stack for the addressed channel module is full, or the absolute address specified by the second word of the CAW is not on an even word boundary (bit 0 must be equal to 0).

4.3.4.2. Test Subchannel - TSC 75,03

The Test Subchannel instruction stores a 3-word snapshot of the addressed subchannel's control words at the absolute address specified by bits 23-0 of the second word of the CAW. The format of the 3-word snapshot is shown in Section 8.

If the TSC instruction is successfully executed, the CPU skips the instruction following the TSC instruction. If the TSC instruction encounters a check, the instruction is aborted, the CPU stores a 3-bit condition code and executes the instruction following the TSC instruction.

The checks that are detected and the associated condition codes are:

- Condition Code = 0

  The TSC instruction has been successfully executed.

- Condition Code = 1

  The selected IOU is not in the application. This is detected by the CPU through a check of the partitioning for the CPU to the referenced IOU interface.

- Condition Code = 2

  A storage check has been detected by the IOU during the fetch of the CAW.

- Condition Code = 3

  The addressed subchannel is not in the application. This is detected by the IOU when the channel descriptor table entry for the referenced subchannel is not a block multiplexer or a word channel module.

- Condition Code = 4

  The IOU detected an internal hardware fault.

- Condition Code = 5

  The absolute address specified by the second word of the CAW is not on a 4-word boundary (bits 0 and 1 must be equal to zero).

- Condition Code = 6

  A storage fault has been detected by the IOU during the storage of the 3-word snapshot.

4.3.4.3. Halt Device - HDV 75,04

The Halt Device instruction sets the mode of the addressed subchannel to idle.

- If the addressed device (subchannel) is logically connected to the channel (operationally inactive), an interface disconnect is done.
- If the device is still connected to the interface, the block multiplexer will do a selective reset sequence.
- If the addressed subchannel is in the busy state without a pending interrupt request, the subchannel is set to the idle state.
- If the addressed subchannel is in the active state with a pending interrupt request, the instruction response indicates an unsuccessful completion.
- If the addressed subchannel is in the status-pending state, the HDV instruction is rejected and the instruction response indicates an unsuccessful completion.
- If the addressed subchannel is not active but is doing SIOF, the mode will be changed to idle, and a selective reset will not be issued by the block multiplexer.
- If the HDV instruction is successfully executed, the CPU skips the instruction following the HDV instruction.
- If a hardware or software fault is detected during the execution of an HDV instruction, the instruction is aborted and a 3-bit condition code is presented to software.

The CPU then executes the instruction following the HDV instruction.

These condition codes are:

- Condition Code = 0

  The HDV instruction has been successfully executed and a selective reset is not executed.

- Condition Code = 1

  The selected IOU is not in the application. This is detected by the CPU through a check of the partitioning information for the CPU to the referenced IOU interface.

- Condition Code = 2

  A storage fault has been detected by the IOU during the fetch of the CAW.

- Condition Code = 3

  The addressed subchannel is not in the application. This is detected by the IOU when the channel descriptor table entry for the referenced subchannel is not a block multiplexer or word channel module.

- Condition Code = 4

  The IOU detected an internal hardware fault.

- Condition Code = 5

  The addressed subchannel is presenting or is about to present status by way of interrupt.

- Condition Code = 6

  The HDV was successfully executed and the subchannel was active.

4.3.4.4. Halt Channel - HCH 75,05

The Halt Channel instruction activates the master clear line on the selected interface. If the subchannel address of an HCH instruction specifies a subchannel on a block multiplexer channel module, the associated block multiplexer channel interface is cleared by disabling OPL OUT for six microseconds. If the subchannel address of an HCH instruction specifies a subchannel on a word interface, the associated word interface is cleared by activating the CLEAR switch on that interface for six microseconds. The clear for a word interface does not affect the other three interfaces on a word channel module. The HCH instruction attempts to write the mode of the addressed subchannel to the idle state.

CAUTION

The execution of the HCH instruction halts all of the associated I/O operations for six microseconds and may cause data overruns.

If the HCH instruction is successfully executed, the CPU skips the instruction following the HCH instruction. If a hardware or software fault is detected during the execution of an HCH instruction, the instruction is aborted, a 3-bit condition code uniquely identifying the fault is presented to software, and the instruction following the HCH instruction is executed.

These condition codes are:

- Condition Code = 0

  The HCH instruction has been successfully executed.

- Condition Code = 1

  The selected IOU is not in the application. This is detected by the CPU through a check of the partitioning information for the CPU to the referenced IOU interface.

- Condition Code = 2

  A storage fault has been detected by the IOU during the fetch of the CAW.

- Condition Code = 3

  The addressed subchannel is not in the application. This is detected by the IOU when the channel descriptor table entry for the referenced subchannel is not a block multiplexer or a word channel module.

- Condition Code = 4

  The IOU detected an internal hardware check.

A Halt Channel instruction issued to the status table subchannel will write its mode to the idle state and clear out any subchannel status. An additional interrupt may be received from the status table with the subchannel status field cleared after successful execution of the Halt Channel instruction.

4.3.4.5. Load Channel Register - LCR 75,10

The Load Channel Register instruction loads the interrupt masked register in the selected IOU. The interrupt masked register provides the capability of selecting which CPU or CPUs accept IOU machine check, normal, or table interrupts. The masked register has six bits that are loaded from bits 5–0 of the second word of the CAW. These six bits control the disposition of interrupts as follows:

Bit 5 = 0:	Enables Machine Check interrupt requests to CPU 1.
= 1:	Disables Machine Check interrupt requests to CPU 1.
Bit 4 = 0:	Enables Normal interrupt requests to CPU 1.
= 1:	Disables Normal interrupt requests to CPU 1.
Bit 3 = 0:	Enables Table interrupt requests to CPU 1.
= 1:	Disables Table interrupt requests to CPU 1.
Bit 2 = 0:	Enables Machine Check interrupt requests to CPU 0.
= 1:	Disables Machine Check interrupt requests to CPU 0.
Bit 1 = 0:	Enables Normal interrupt requests to CPU 0.
= 1:	Disables Normal interrupt requests to CPU 0.
Bit 0 = 0:	Enables Table interrupt requests to CPU 0.
= 1:	Disables Table interrupt requests to CPU 0.

If table interrupt requests are disabled to both CPU 0 and CPU 1, table entries are still made, but a table interrupt request is not presented until table interrupts are enabled on at least one CPU. The Interrupt Masked register is set to enable Normal interrupts to all CPUs and disable Table and Machine Check interrupts at power up and master clear time.

The execution of the LCR instruction always loads all six bits of the Interrupt Masked register. Any interrupts that are currently being presented when an LCR instruction is received will remain under control of the previous interrupt mask.

If the LCR instruction is successfully executed, the CPU skips the instruction following the LCR instruction. If a hardware check is detected during the execution of an LCR instruction, the instruction is aborted, a 3-bit condition code uniquely identifying the check is presented to software, and the instruction following the LCR instruction is executed.

The checks that are detected and the associated condition codes are listed below:

- Condition Code = 0

  The LCR instruction has been successfully executed (Interrupt Masked register loaded).

- Condition Code = 1

  The selected IOU is not in the application. This is detected by the CPU through a check of the partitioning information for the CPU to the referenced IOU interface.

- Condition Code = 2

  A storage check has been detected by the IOU during the fetch of the CAW.

- Condition Code = 3

  Not used by LCR instruction.

- Condition Code = 4

  The IOU detected an internal hardware check.

4.3.5. IOU Control Words

IOU data transfer operations are initiated when the CPU executes an I/O instruction and are controlled in the IOU via a Channel Address Word (CAW) and one or more Channel Command Words (CCWs). Updated copies of the CAW and CCW for each subchannel are maintained in the IOU central control hardware. Status detected or received during execution of an I/O operation and ending status associated with ISI, block multiplexer, and the status table subchannels are stored by the IOU in a Channel Status Word (CSW) in a reserved location in storage during the interrupt sequence. Ending status associated with ESI subchannels, which cannot be stacked, is stored in a Table Status Word (TSW).

4.3.5.1. Channel Address Words

The double word CAW is loaded by CPU hardware in a reserved location in low order storage during the execution of an I/O instruction. The first word of the CAW specifies the instruction, the IOU, and the subchannel.

For the SIOF instruction, bits 23–00 of the second word of the CAW contain the absolute address of the first CCW for the subchannel that is being initiated.

For the TSC instruction, bits 23–00 of the second word of the CAW contain the starting absolute address location for the storing of the snap shot of a subchannel's control words.

For the LCR instruction, bits 05–00 of the second word of the CAW contain the data to be loaded in the mask register.

4.3.5.2. Channel Command Words

The Channel Command Word (CCW) is a 72–bit control word used to govern the data transfer to or from a specific peripheral. The CCWs are written into main storage by the CPU and are pointed to by the address field of the SIOF instruction.

A double–word CCW may be located anywhere in storage, but it must be on a double–word boundary. A CCW specifies the operation to be performed by a device or subchannel. When the specified operation is to be executed, the CCW is moved to the appropriate subchannel storage location. The CCW is then modified and updated to control the operation as it progresses.

New CCWs are generally located in the next two contiguous locations following the present CCW. A Transfer In Channel (TIC) command in the present CCW causes a jump to a new CCW list. A normal ending status containing the status modifier bit causes the channel to skip the next contiguous CCW.

4.3.5.2.1. ISI Word Interface CCW

The format of the ISI word interface CCW follows:

Reserved for Software	Command Code	Data Address
35 – 28	27 – 24	23 – 0

Reserved for Software	CCW Flags									Reserved for Software	Word Count
	CD	CC	XXX	SK	XXX	MON	DAD	DAL	XXX		
71 – 68	67	66	65	64	63	62	61	60	59	58 – 52	51 – 36

XXX = Not Used

where:

Bits 27–24 Command Code specifies the operation to be performed by the device and subchannel; however, this command code is not sent to the device. The device responds to the External Function word. In the command codes listed below, the letter X indicates that the bit position is ignored.

Command	Bits 27	26	25	24
Invalid	0	0	0	0
Invalid	0	1	0	0
Transfer in Channel (TIC)	1	0	0	0
Invalid	1	1	0	0
Write	X	X	0	1
Read	X	X	1	0
Forced External Function	X	X	1	1

Bits 23–0 Data Address contains the absolute storage address for the first data word or External Function (EF) word to be transferred unless the command code is TIC. In this case the field contains the absolute storage address of the new CCW.

Bits 67–59 CCW Flags

Bit 67 Chain Data (CD) specifies that upon exhaustion of the word count of the current CCW, a new CCW is to be read from storage and the operation is to be continued under control of the new CCW. The CD flag is ignored on all command CCWs with the word count equal to 0.

Bit 66	Chain Command (CC) specifies that upon exhaustion of the word count of the current CCW, a new CCW is to be read from storage, and the operation specified by the new command code is initiated. If the CD flag is set, the CC flag is ignored.
Bit 64	SKip data (SK) specifies that data is not written in storage for input operations. However, the subchannel and control words are handled in the same manner as during conventional input operation. The SK flag is ignored on output operations.
Bit 62	MONitor (MON) specifies that the subchannel generate an interrupt and present monitor subchannel status when the word count in the last CCW of the channel program has been exhausted. The MON interrupt condition is detected for ISI operations when the device requests one more data transfer than the count specified in the CCW word count field.
Bit 61	Data Address Decrement (DAD) specifies that the data address be decremented by one for each data word transferred. This flag is ignored if the Data Address Lock (DAL) flag is set. If neither the DAL nor DAD flag is set, the data address is increased by 1 for each data word transferred.
Bit 60	Data Address Lock (DAL) specifies that the contents of the data address field remain unchanged for each data word transferred under control of the current CCW.
Bits 51–36	Word Count specifies the number of words to be transferred to or from storage.

4.3.5.2.2. ESI Word Interface CCW

The format of the ESI word interface CCW follows.

Reserved for Software	FCF	Command Code	Data Address
35 – 30	29 28	27 – 24	23 – 0

Reserved for Software	CCW Flags									Reserved for Software	Data Count
	CDC	CC	EIC	SK	XXX	MON	DAD	DAL	EOT		
71 – 68	67	66	65	64	63	62	61	60	59	58 – 52	51 – 36

XXX = Not Used

where:

Bits 29-28 The Format Control Flags (FCF) specify the location within a word of the first data character. In quarter-word mode, the FCF specify the location of the first quarter word within the first word of data for each CCW.

Bits 29-28	Description
00	Specifies that the first quarter word be selected from or placed in bits 35-27 of the first data word.
01	Specifies that the first quarter word be selected from or placed in bits 26-18 of the first data word.
10	Specifies that the first quarter word be selected from or placed in bits 17-9 of the first data word.
11	Specifies that the first quarter word be selected from or placed in bits 8-0 of the first data word.

As each quarter word is used (input or output), the format count is updated in the IOU by +1. The counter counts module 4 with the carry from the fourth count used to update the data address field. The counts are updated after the data transfer, which leaves the IOU internal CCW pointing to the next word to be transferred.

In half-word mode, the FCF specifies the location of the first half word within the first word of data for each CCW.

Bit 28	Description
0	Specifies that the first half word be selected from or placed in bits 17-0 of the first data word.
1	Specifies that the first half word be selected from or placed in bits 35-18 of the first data word.

As each half-word is used (input or output), the format flag is updated in the IOU by +1. The counter counts modulo 2 with the carry from the second count used to update the word count field. The counts are updated after the data transfer, which leaves the IOU internal CCW pointing to the next word to be transferred.

Bits 27-24 Command Code specifies the operation to be performed by the device and subchannel. However, this command code is not sent to the device. The device responds to the external function word. In the command codes listed below, the letter X indicates that the bit position is ignored by the IOU.

Command Code	Bits 27	26	25	24
Invalid	0	0	0	0
Invalid	0	1	0	0
TIC	1	0	0	0
Invalid	1	1	0	0
Write	X	X	0	1
Read	X	X	1	0
Forced External Function	X	X	1	1

Bits 23–0	Data Address contains the absolute storage word address of the first data character to be transferred unless the command code is TIC. In this case, the field contains the absolute storage address of the new CCW.
Bits 51–36	Data Count specifies the number of characters (quarter words or half words) to be transferred to or from storage.
Bits 67–59	CCW Flags
Bit 67	Chain Data (CD) specifies that upon exhaustion of the character count of the current CCW, a new CCW is to be read from storage and the operation is to be continued under control of the new CCW.
Bit 66	Chain Command (CC) specifies that upon exhaustion of the character count (data count) of the current CCW, a new CCW is to be read from storage and the operation specified by the new command code is to be initiated. If the CD flag is set, the CC flag is ignored. ESI command chaining from a Write command (01) is not supported and causes a Program Check Subchannel Status condition.
Bit 65	El Chain (EIC) specifies that upon completion of the operation at the device (El presented), a new CCW is to be read from storage, and the execution of the CCW list is to be continued under control of the new CCW. The external interrupt presented by the device is stored as part of a Table Status Word (TSW) prior to chaining. EI chaining to a forced External Function command is not allowed. Only the TIC, Read, or Write commands are used.
Bit 64	SKip data (SK) specifies that data is not written into storage for input operations. However, the subchannel and control words are handled in the same manner as during conventional input operation. The SK flag is ignored on output operations.
Bit 62	MONitor (MON) specifies that the subchannel shall store a TSW when the data count in the current CCW has been exhausted. Monitor conditions for ESI are detected when the last word buffer is transferred. Monitor interrupts occur on ESI for each CCW in a CCW list.
Bit 61	Data Address Decrement (DAD) specifies that the data address be decreased rather than increased. This flag is ignored if the DAL flag is set. If neither the DAL nor DAD flag is set, the data address is increased when a word boundary is reached.
Bit 60	Data Address Lock (DAL) specifies that the contents of the data address field remain unchanged for each data word transferred under control of the current CCW.

Bit 59 End Of Transmission (EOT) specifies that an EOT bit (a 1-bit in bit position 9 of the output data word) be transmitted with the last character that is transferred under control of the CCW. The EOT bit is not set and not transmitted to the peripheral if the data chain flag in the CCW is set.

4.3.5.2.3. Block Multiplexer Channel CCW

The format of the block multiplexer channel CCW follows:

Reserved for Software	Command Code	Data Address
35 32	31 24	23 0

Reserved for Software	CCW Flags												Reserved for Software	Data Count
	CD	CC	SLI	SK	X	TS	DAD	DAL	X	FORMAT A	FORMAT B	FORMAT C		
71 68	67	66	65	64	63	62	61	60	59	58	57	56	55 52	51 36

X = Not Used

where:

Bits 31–24 Command Code specifies the operation to be performed by the device and subchannel. In the command codes listed below, the letter X indicates that the bit position is ignored, and the letter M identifies a modifier bit. The meaning of the modifier bits depends upon the type of I/O device and is defined in the reference manuals for the individual devices.

	Bits							
Command Code	31	30	29	28	27	26	25	24
Invalid	X	X	X	X	0	0	0	0
Sense	M	M	M	M	0	1	0	0
TIC	X	X	X	X	1	0	0	0
Read Backward	M	M	M	M	1	1	0	0
Write	M	M	M	M	M	M	0	1
Read Forward	M	M	M	M	M	M	1	0
Control	M	M	M	M	M	M	1	1

Bits 23–0 Data Address contains the absolute address for the first data word to be transferred unless the command code is TIC. In this case, the field contains the absolute address of the new CCW.

Bits 67-56	CCW Flags
Bit 67	Chain Data (CD) specifies that upon exhaustion of the word count of the current CCW, a new CCW is to be read from storage and the operation is to be continued under control of the new CCW.
Bit 66	Chain Command (CC) specifies that upon completion of the operation at the device (status code contains Device End), a new CCW is to be read from storage and the operation specified by the new command code is to be initiated. If the CD flag is set or incorrect length conditions are detected, the CC flag is ignored.
Bit 65	Suppress Length Indication (SLI) flag disables the checking of subchannel word count versus the number of bytes presented by a device. If the SLI flag is not set and the number of bytes that a device attempts to transfer to the IOU is not exactly equal to the CCW word count, the subchannel program is immediately terminated and incorrect length subchannel status is reported to the software. If the SLI flag is set and the number of bytes that a device attempts to transfer is not exactly equal to the CCW word count, the operation is unaffected and the execution of the subchannel program proceeds normally.
Bit 64	SKip data (SK) specifies that data is not written in storage for input operations. However, the subchannel and control words are handled in the same manner as during conventional input operation. The SK flag is ignored on output operations.
Bit 62	Truncated Search (TS) specifies that the channel module issues a Search command after the current CCW command and then reissue the current command as long as proper status is received from the device and until the word count is exhausted. (Truncated search is not supported by systems using the compatible channel interface feature.)
Bit 61	Data Address Decrement (DAD) specifies that the data address be decremented by one for each data word transferred. This flag is ignored if the DAL flag is set. If neither the DAL nor DAD flag is set, the data address is incremented for each data word transferred.
Bit 60	Data Address Lock (DAL) specifies that the contents of the data address field remain unchanged for each data word transferred under control of the current CCW.
Bit 58	Format A specifies the quarter word format for packing bytes into words and unpacking bytes from words. (See Tables 4-1 and 4-2.) One byte is right-justified in each 9-bit quarter word. If the ninth bit in any quarter word is set, the data transfer is terminated, and an interrupt request generated. Four bytes are packed in each 36-bit word.
Bit 57	Format B specifies 6-bit packed format for packing bytes into words and unpacking bytes from words. (See Tables 4-1 and 4-2.) The most significant two bits of every byte are ignored on input and set to zero on output. Six characters of six bytes each can be packed in each 36-bit word.
Bit 56	Format C specifies 8-bit packed format for packing bytes into words and unpacking bytes from words. (See Tables 4-1 and 4-2.) Four and one-half bytes are packed in each 36-bit word. On a Format C read, when a device

terminates on the fifth byte of a nine-byte group, the last byte is not discarded. It is properly justified, zero filled, and sent to storage.

Bits 51-36 Data Count specifies the number of words to be transferred to or from storage. All writes or reads are full-word transfers.

4.3.5.2.4. Status Table Subchannel CCW

The status table is only for ESI status. The status table generates a status table subchannel CSW. Status table entries are described in 8.4.2.3. The format of the status table subchannel CCW follows:

Reserved for Software	Command Code	Table Address
35 28	27 24	23 0

Reserved for Software	CD	Reserved for Software	MON	Reserved for Software	Table Entry Count
71 68	67	66 63	62	61 52	51 36

where:

Bits 27-24 Command Code specifies either an input or TIC command. In the command codes listed below, the letter X indicates that the bit position is ignored.

	Bits							
Command Code	31	30	29	28	27	26	25	24
TIC	X	X	X	X	1	0	0	0
Activate Table	X	X	X	X	0	X	X	X
Activate Table	X	X	X	X	1	1	X	X
Activate Table	X	X	X	X	1	X	1	X
Activate Table	X	X	X	X	1	X	X	1

Bits 23-0 Table Address specifies the starting address for the next table entry.

Bit 67 CCW Flag Chain Data (CD) specifies that, upon exhaustion of the table entry count of the current CCW, a new CCW is to be read from storage and the operation is to be continued under control of the new CCW. The CD flag is ignored on all command CCWs with data count equal to zero.

Bit 62 CCW Flag MONitor (MON) interrupt specifies that the status table subchannel generated an interrupt request and presents the MON subchannel status after the entry count is decreased to zero for the current CCW.

If interrupts are locked out too long, the Monitor interrupt may be overlaid with more interrupts.

Bits 51-36 Table Entry Count specifies the number of entries to be made in the table. Note that each 2- or 3-word Table Status Word (TSW) counts as only one entry in the status table.

Table 4-1. MSU Data Format – 36-Bit Format, Forward Operation

① *Format A*

*	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	*	B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0	*	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0	*	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
	34		32		30		28		26		24		22		20		18		16		14		12		10		8		6		4		2		0

Format B

A_5	A_4	A_3	A_2	A_1	A_0	B_5	B_4	B_3	B_2	B_1	B_0	C_5	C_4	C_3	C_2	C_1	C_0	D_5	D_4	D_3	D_2	D_1	D_0	E_5	E_4	E_3	E_2	E_1	E_0	F_5	F_4	F_3	F_2	F_1	F_0
	34		32		30		28		26		24		22		20		18		16		14		12		10		8		6		4		2		0

Format C

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	E_7	E_6	E_5	E_4
	34		32		30		28		26		24		22		20		18		16		14		12		10		8		6		4		2		0

Format C (continued)

E_3	E_2	E_1	E_0	F_7	F_6	F_5	F_4	F_3	F_2	F_1	F_0	G_7	G_6	G_5	G_4	G_3	G_2	G_1	G_0	H_7	H_6	H_5	H_4	H_3	H_2	H_1	H_0	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0
	34		32		30		28		26		24		22		20		18		16		14		12		10		8		6		4		2		0

* *If the ninth bit in any Format A is set, the data transfer is terminated (write only).*

① *The letters indicate the order in which bits are transferred. A indicates the first byte transferred, B indicates the second byte transferred, C indicates the third byte transferred, etc. The subscripts indicate the bit position in the byte. A 7 is the most significant bit in the byte, a 6 is the second most significant bit in the byte, a 5 is the third most significant bit in the byte, etc.*

Table 4-2. MSU Data Format – 36-Bit Format, Backward Operation

Format A ①

*	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	*	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0	*	B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0	*	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
	34		32		30		28		26		24		22		20		18		16		14		12		10		8		6		4		2		0

Format B

F_5	F_4	F_3	F_2	F_1	F_0	E_5	E_4	E_3	E_2	E_1	E_0	D_5	D_4	D_3	D_2	D_1	D_0	C_5	C_4	C_3	C_2	C_1	C_0	B_5	B_4	B_3	B_2	B_1	B_0	A_5	A_4	A_3	A_2	A_1	A_0
	34		32		30		28		26		24		22		20		18		16		14		12		10		8		6		4		2		0

Format C

E_3	E_2	E_1	E_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0	B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
	34		32		30		28		26		24		22		20		18		16		14		12		10		8		6		4		2		0

Format C (continued)

I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	H_7	H_6	H_5	H_4	H_3	H_2	H_1	H_0	G_7	G_6	G_5	G_4	G_3	G_2	G_1	G_0	F_7	F_6	F_5	F_4	F_3	F_2	F_1	F_0	E_7	E_6	E_5	E_4
	34		32		30		28		26		24		22		20		18		16		14		12		10		8		6		4		2		0

* *If the ninth bit in any Format A quarter word is set, the data transfer is terminated.*

① *The letters indicate the order in which bits are transferred. A indicates the first byte transferred, B indicates the second byte transferred, C indicates the third byte transferred, etc. The subscripts indicate the bit position in the byte. A 7 is the most significant bit in the byte, a 6 is the second most significant bit in the byte, a 5 is the third most significant bit in the byte, etc.*

4.4. Execution of I/O Operations

The commands that a channel module can execute are specified by the command code field of the CCW (bits 27 through 24 of CCW 1 for word channels, and bits 31 through 24 of CCW 1 for block multiplexer channel modules).

A channel can execute six commands: Write, Read, Read Backward (only on a block multiplexer channel), Sense (only on a block multiplexer channel), Control, and Transfer In Channel (TIC).

Each command, except TIC, initiates a corresponding I/O operation. The initiation and execution of a command issued to a subchannel and device is termed an I/O operation.

For word channel peripherals, the command code in the CCW is not sent out as the external function word but, rather, the command code is used by the IOU. The external function word is fetched from storage in the same manner as data. For block multiplexer channels, the command code in the CCW is the same command code that is sent out to the peripheral control unit.

A series of I/O operations on the same device (block multiplexer channel) or on the same subchannel (word channel) is executed under control of a set of CCWs. The execution of a set of CCWs (CCW list) is initiated by an SIOF instruction. For an SIOF instruction, the address of the first CCW is obtained from the CAW and is stored in the IOU at the specified subchannel location. Then, a condition code is presented to the CPU. At an idle time in the channel sequencing, the first CCW is fetched and the specified I/O operation is initiated. The CCWs can be located on any double-word boundary in main storage. Fetching of a CCW by the IOU does not affect the contents of the location in main storage. Each additional CCW in the CCW list is obtained when the operation has progressed to the point where an additional CCW is needed. As each CCW is fetched, the address of the next CCW for the subchannel is retained in the IOU.

Data transfers are controlled by the data address and word count fields of the CCW. Initially the data address field contains the storage address of the first data to be transferred. The word count field of a CCW specifies the number of words to be transferred. On a word channel ISI interface or a block multiplexer channel, the word count specifies the number of 36-bit words to be transferred. On a word channel ESI interface, the word count specifies the number of quarter words or half words to be transferred.

A CCW operation can be terminated by the IOU or by the device. A CCW operation may also be terminated by the HDV or HCH instructions. Termination by the HCH and HDV instructions is covered in the instruction descriptions (see 4.4). The IOU terminates the operation when the word count is exhausted. A device terminates the operation by presenting status. When a CCW operation is terminated, either a new CCW is fetched and a new operation is initiated, or an interrupt is generated. A subchannel on a word channel may, instead of fetching a new CCW or generating an interrupt, return to the available state with no other action being taken. This occurs when the word count for the current operation is exhausted; the Chain Data, Chain Command, and Monitor flags are all cleared, and the device does not present an external interrupt. This condition can be prevented by setting the Monitor flag. The Monitor flag set specifies that an interrupt is to be generated when the data count of the present operation is exhausted.

When an operation is terminated, the action taken by the subchannel is determined by the condition that caused the operation to be terminated and by the Chain Data, Chain Command, Truncated Search, Suppress Length Indicator (SLI), EI Chain, and Monitor flags of the CCW flag field.

4.5. Transfer In Channel (TIC) Command

The TIC command provides a branching function in the IOU. The TIC command provides for chaining between CCWs not located in contiguous storage locations. In addition, the TIC command provides the means to set up command chaining or data chaining loops. The loop is terminated when an interrupt is received from the device and transmitted from the channel module to the IOU control section. A new CCW is fetched from the location designated by the data address field of the TIC command CCW. A new CCW and CCW list are immediately initiated. The TIC command can be used with data chaining and command chaining. The word count field, the format flags, and the CCW flags of a TIC command are not interpreted. The data chain or command chain operation is carried through the TIC CCW to the new CCW.

If the CCW address of a TIC command is not on a double-word boundary, the execution of the CCW list is terminated, and an interrupt is presented with the program check bit of the subchannel status field set.

4.6. Chaining Operations

When a channel has performed the transfer of data specified by a CCW, it can continue the activity initiated by the SIOF instruction by fetching a new CCW (chaining). Chaining occurs only between CCWs located in successive double-word locations in storage. A CCW list is executed in an ascending order of addresses. The address of a new CCW is obtained by adding two to the address of the current CCW. Two CCW lists in noncontiguous storage locations can be connected for chaining purposes by a TIC command. On a block multiplexer channel, all CCWs of a CCW list apply to the I/O device specified in the original SIOF instruction. On a word channel, all CCWs of a CCW list apply to the subchannel specified in the original SIOF.

Three types of chaining are provided: data chaining, command chaining, and EI chaining (valid only on an ESI subchannel). The specification of chaining is effectively propagated through a TIC command. When, in the process of chaining, a TIC command is detected, the CCW designated by the TIC is used for the type of chaining specified in the CCW preceding the TIC CCW.

A chaining operation is initiated when the operation on the present CCW is completed. A CCW operation is completed when either the data count is exhausted or the device presents status. The combination of the data count, device status, Chain Data flag, Chain Command flag, and EI Chain flag, determine what type of chaining, if any, occurs.

4.6.1. Data Chaining

Data chaining permits blocks of information to be transferred to or from noncontiguous areas of storage as one continuous data transfer from the peripheral. The address of the data is normally changed after each word is transferred. When using the Skip Data flag, data chaining allows the software to place selected portions of a block of data in main storage.

For data chaining, the new CCW fetched by the channel defines a new storage area for the original I/O operation. Execution of the operation at the I/O device is not affected. The contents of the command code field of the new CCW is ignored unless it specifies a TIC command.

Data chaining is executed immediately after the last word under control of the current CCW has been transferred to storage or to channel module for input or output, respectively. The old CCW is replaced by the new CCW before another data request from the channel module is handled. If the device presents status after exhausting the count of the current CCW, but before transferring any data to or from the storage area designated by the new CCW, the action taken by the IOU is controlled by the new CCW flags. If a hardware or software error is detected when fetching the new CCW, the operation is terminated and an interrupt is generated. The CSW or TSW associated with the interrupt indicates why the operation was terminated.

Data chain CCWs are prefetched by the IOU hardware. Each channel module has a data buffer in the channel module hardware that decreases the probability of data overruns under adverse conditions, such as peak channel module activity and peak IOU control activity. During output operations, a data chain is executed immediately after the last word under control of the current CCW has been transferred to the data buffer. The old CCW is replaced by the new CCW and the IOU continues to accept data under control of the new CCW. The data buffer in the channel module continues to accept data from the peripheral (input) or transmit data to the peripheral (output) during the data chaining operation in the central control of the IOU until the buffer is full or empty respectively. If a hardware or software error is detected when fetching the new CCW, the operation is terminated and an interrupt is generated. The CSW/TSW associated with the interrupt indicates why the operation was terminated.

The Suppress Length Indicator (SLI) bit (see 4.3.5.2.3) of the CCW flags has significance only in the first CCW of data chained channel program. The use of the SLI bit in any subsequent CCW of the channel program is ignored. This also holds true for block multiplexer format bit. This is because no new CCWs are issued to the channel module for any subsequent data channel CCW.

4.6.2. Command Chaining

A subchannel executes a command chain operation by retrieving a new CCW and beginning execution of the command specified by that CCW. The subchannel is activated and begins executing the new command. On a block multiplexer channel module, the new command is passed directly to the device. On a word channel module, the external function word is typically specified by one CCW, and an associated data buffer is specified by a second CCW.

Command chaining makes it possible for software to initiate the transfer of multiple blocks of data by means of a single SIOF instruction. It also allows a subchannel to initiate the execution of auxiliary functions and data transfer operations without software interference at the end of each intermediate operation. On a block multiplexer channel, command chaining, in conjunction with the status modifier condition, allows the IOU to modify the normal sequence of operations in response to signals provided by the I/O device.

During command chaining, the new CCW fetched by the channel specifies a new I/O operation. On a block multiplexer channel, command chaining occurs only when the I/O device presents chaining status, the Chain Command flag is set, and the Chain Data flag is not set, and either the byte count equals zero or the SLI flag is set. On a word channel, command chaining occurs only when the word count of the current operation is exhausted, the Chain Command flag is set, and the Chain Data flag is not set. When command chaining takes place, the completion of the current operation does not cause an interrupt.

Command chaining takes place, and the new operation is initiated, only if no hardware error has been detected during the current operation. If a hardware error has been detected, the operation is immediately terminated and an interrupt is generated. Also, if a hardware or software error is detected when trying to initiate the new CCW of a command chain, the operation is terminated, and an interrupt is generated.

An exception to the sequential chaining of CCWs occurs on a block multiplexer channel when the I/O device presents the status modifier condition and the chaining status. When command chaining is specified, the combination of chaining status and the status modifier condition causes the IOU to fetch and chain to the CCW whose main storage address is four higher than that of the current CCW.

Command chaining is allowed when word counts equal zero. The only exception to this is ESI output. In this case, command chaining is illegal, regardless of the word count.

4.6.3. EI Chaining (ESI Word Interface Only)

The External Interrupt (EI) Chain flag is interpreted only on an ESI subchannel. A subchannel executes an EI chaining operation by placing the External Interrupt in the status table if the status table subchannel is active, retrieving a new CCW from a storage address two higher than the storage address of the current CCW, and executing the command specified by that CCW. An EI chain is executed only if all of the following conditions are met:

1. The EI Chain CCW flag is set.
2. An ESI external interrupt is presented.
3. The status table subchannel is active.
4. No hardware error is detected when making an entry into the status table for the external interrupt.
5. The subchannel is active.

   NOTE: *If the data count on a word subchannel is exhausted and neither data chaining nor command chaining is specified, the subchannel is immediately changed from active mode to either status pending or idle mode.*

6. No hardware or software error is detected when retrieving the new CCW.

If the EI Chain flag is set along with the Chain Data and/or Chain Command flags, the action taken depends upon which event happens first. If the channel detects exhaustion of the word count first, the data chain or command chain will be performed, and any subsequent EI chain is under the control of the EI Chain flag in the new CCW. If the channel detects the external interrupt first, an EI chain is performed if the EI Chain flag in the current CCW is set.

If a software or hardware error is detected when trying to initiate the new CCW during an EI chain, the operation is terminated, a TSW explaining the termination is stored in the status table, and a Table interrupt request is generated. The subchannel is returned to the idle state.

4.6.4. Truncated Search (Block Multiplexer Channel Only)

The truncated search capability provides software with a simple yet effective method of reading or writing multiple records on a disk control unit. A truncated search operation is executed under control of the CCW Truncated Search flag. A block multiplexer channel executes a truncated search operation by reissuing the Search command and the command in the current CCW, when proper status is received from the device before the data count is exhausted. (Truncated search is not supported by systems using the compatible channel interface feature.)

The following is an example of a truncated search operation:

CCW	Command
1	Search command (This CCW has the CC flag set.)
2	TIC command (TIC back to CCW 1.)
3	Read or Write command (This, and only this, CCW must have the Truncated Search flag set.)

The IOU retrieves CCW 1 and issues the Search command and search bytes to the device. If the control unit does not make a compare on the search bytes, device status of Channel End and Device End is presented to the block multiplexer channel module. The IOU executes a command chain, retrieves the TIC command, and then retrieves the Search command and reissues the Search command to the device. This loop continues until the device makes a compare (finds the correct record) and presents device status of Channel End, Device End, and Status Modifier. Because of the special device status, the IOU skips the next CCW (CCW 2) and executes CCW 3. The Read or Write command is issued to the device and the device begins transferring data.

When the device detects the end of a record, device status of Channel End and Device End is presented to the block multiplexer channel module. The IOU checks the device status and detects the CCW Truncated Search flag. The IOU begins the truncated search operation by reissuing the previous command (the Search command from CCW 1) and responding to the first request for data with the COMMAND OUT interface line. The control unit then responds by presenting device status of Channel End, Device End, and Status Modifier. The block multiplexer channel module reissues the Read or Write command and data transfer is initiated, starting with the residual word count and data address from the previous read/write operation. This procedure is repeated at the end of each record until the word count is exhausted. Data chaining after the word count is exhausted is allowed.

NOTE: When the word count has been exhausted from a truncated search input operation, the channel module does not terminate the operation. Instead, another truncated search operation is initiated. For this reason, command chaining should not occur on a truncated search operation.

4.7. Interrupt Generation Flag

The MONitor flag (MON) is an interrupt generation flag that causes the subchannel to generate an interrupt. The MON flag is set in the subchannel status field.

4.7.1. Monitor (Word Channel and Status Table Subchannel)

The MON flag specifies that an interrupt will be generated with the MON subchannel status bit set. The MON interrupt is generated at the end of the operation specified by the CCW on ESI operation, independent of chaining. On ISI operations the MON flag is ignored if chaining is specified. The MON bit of the subchannel status field may accompany other valid subchannel status bits.

4.7.2. ISI Monitor Conditions

The MON flag is interpreted only in the final CCW of a CCW list. If the Chain Command or Chain Data flag is set in the CCW, the MON flag is ignored and no interrupt is presented.

4.7.3. ESI Subchannel Conditions

If no errors are encountered, a table entry with the MON subchannel status bit set will be generated when the word count in the current CCW is exhausted. If the Chain Data flag is also set, the monitor table entry will be generated before the data chain is done. If a device EI is received before the word count is exhausted, the MON flag is ignored.

4.7.4. Status Table Subchannel Monitor Conditions

A MON interrupt is generated when the entry count is exhausted if no errors are encountered. If data chaining is also specified, the data chain will occur before the MON interrupt is generated.

NOTE: Interrupts may be locked out for an extended period of time. When data chaining with the MON flag set in two or more CCWs, only one interrupt may be generated.

4.8. Status

I/O status can be separated into the following four categories:

1. Channel Status – Hardware error that cannot be associated with a particular device or subchannel.

2. Status for Noncommunications Subchannels – Status caused by a device, CCW flags, or a hardware or software error on block multiplexer or ISI subchannels.

3. Status for Communications Subchannels – Status caused by a device, CCW flags, hardware error, or software error on ESI subchannel.

4. Status for Status Table Subchannel – Status caused by the MON CCW flag, hardware error, or software error on the status table subchannel.

I/O status is presented either by Test Subchannel (TSC) instruction, status table, or interrupt. A standard Channel Status Word (CSW) or Table Status Word (TSW) is generated in all four cases. The format is:

CSW or TSW

IOU No.	Subchannel Address	Next CCW Address
35 34	33 24	23 0

*	Device Status	Subchannel Status	Residual Data Count
71 68	67 60	59 52	51 36

External Interrupt Status Word (Word Channel Only)	
107	72

where:

Bits 71–68	Residual Byte Count (block multiplexer channel only)	
Bits 67–60	Device Status	
Bit 67	Attention	
Bit 66	Status Modifier	
Bit 65	Control Unit End	Block Multiplexer Only
Bit 64	Busy	
Bit 63	Channel End	
Bit 62	Device End	
Bit 61	Unit Check	
Bit 60	Unit Exception	
Bits 59–52	Subchannel Status	
Bit 59	Not Used	
Bit 58	Incorrect Length (block multiplexer channel) Monitor Interrupt (word channel)	
Bit 57	Program Check	
Bit 56	Internal Hardware Check	
Bit 55	Storage Check	
Bit 54	Channel Module Bus Check	
Bit 53	Channel Module Control Check	
Bit 52	Device Not Available Check (block multiplexer channels)	

On a block multiplexer channel, the device status field contains the actual status presented by the device. On a word channel, the only valid device status bit is the Attention bit (bit 67). On a word channel, the Attention bit set indicates that bits 107–72 of the CSW contain an external interrupt status word. A cleared Attention bit indicates that bits 107–72 of the CSW are invalid.

Each of the two IOUs has one reserved address for the machine check status word and three addresses for the CSW. Status is presented if interrupts are not masked, and the interrupt acknowledge lines from the CPU are inactive.

4.9. Initial Load

An initial load capability is provided on block multiplexer subchannels and ISI word subchannels. The initial load operation is initiated by the System Support Processor (SSP). For a more detailed initial load operation see *SPERRY 1100/70 Systems, System Support Processor (SSP), Operator Reference,* UP-9123 (applicable version).

The SSP can execute I/O instructions and process I/O interrupts by using the SSP's port in main storage and by using the scan/set interface in the IOU.

The SSP performs the initial load by:

1. Initializing the IOU, which involves master clearing the IOU and setting each subchannel to the idle state.

2. Activating the initial load program, which involves loading the CAW and CCW list in main storage and initiating the IOU via the scan/set interface.

3. Processing initial load status, which involves detecting completion conditions by way of the scan/set interface and retrieving status from main storage.

4. Initiating the CPU, which involves issuing an SSP interrupt to the CPU to activate the CPU.

4.10. Word Channel Back-to-Back Operations

A back-to-back data transfer capability is provided on each word interface. The back-to-back mode of operation is activated and deactivated by the SSP. Output can be activated on a given word interface, and input can be activated on any of the other three word interfaces in the word channel module. Input and output cannot be active at the same time on a given word interface.

When initiating a back-to-back operation, the input channel must be activated before the output channel. Input command chaining in back-to-back mode will result in the loss of one to four words of data. Only the input channel can be run in ESI mode back-to-back.

Correct operation of the word channel cannot be guaranteed when back-to-back interfaces are operated concurrently with other word devices on the same channel.

The following is the recommended back-to-back programming procedure:

1. Activate the input channel with the MON flag set in the final CCW.
2. Activate the output channel.
3. Wait for the input channel monitor interrupt indicating completion of the data transfer.

4.11. Priorities

The Central Control Module (CCM) establishes priority among the six channel modules. There are six priority assignments, up to five may be used at any one time. Channel module 0 has preemptive priority over all other channel modules. Channel modules 1, 2, and 3 have a cyclic priority among themselves and are higher in priority than 4 and 5. Channel module 4 has preemptive priority over channel module 5. The CCM gives highest priority to delayed storage checks, second priority goes to data transfers. Third priority is status requests from a channel module. Fourth comes initiating I/O instructions from the processor. Fifth is initiating a queued-up SIOF operation. Sixth in the priority structure is status table Normal interrupts. Seventh is reporting a Table interrupt request, and last is reporting a Machine Check interrupt.

Priority summary (highest listed first):

1. Delayed storage checks.

2. Data transfers to and from channel modules. (Highest priority listed first.)

0	Channel Module #0	
1	Channel Module #1	
2	Channel Module #2	Cyclic
1	Channel Module #3	
2	Channel Module #4	
3	Channel Module #5	

3. Status requests from channel modules (same priority among channel modules as data transfers).

4. Initiating I/O instructions from a processor.

5. Delayed Initiation of a queued–up SIOF.

6. Status table Normal interrupts.

7. Reporting Table interrupt requests.

8. Machine Check interrupt.

4.12. Basic Programming Procedure

The programmer should use the following basic procedure to execute a series of operations on a block multiplexer device or a word subchannel:

1. Generate the channel program, making sure the correct CCW flags are set in each CCW. Also, generate any necessary external function words or data buffers.

2. Load the address of the first CCW in X_a bits 23–0. (The X–register specified by the a–field of the I/O instruction in the processor.)

3. The u–field of the I/O instruction plus the (X_m) specified by the x–field of the I/O instruction, specifies in bits 9–0 of X_m, the subchannel address for the I/O operation. The contents of $U = u + X_m$ are not used to specify the subchannel address.

4. Execute the SIOF instruction.

5. Test the condition code to determine the result of the SIOF instruction. (Note that the next instruction is skipped if the condition code equalled 0.)

6. If a condition code of zero is received, enable I/O interrupts and continues with the processor program. If any other condition code is received, the appropriate action should be taken.

7. Wait for the I/O interrupt or interrupts. Use the resultant status to determine if the CCW list was successfully executed. If the CCW list was terminated before it was completed, the status will contain enough information to determine how much of the CCW list was executed and why the list was terminated before it was completed.

5. System Support Processor

5.1. General

The System Support Processor (SSP) is a desk-sized separate minicomputer unit that interfaces with the 1100/70 System central complex units: Central Processing Unit (CPU), Input/Output Unit (IOU), Main Storage Unit (MSU), and Storage Interface Unit (SIU). The SSP interfaces with the central complex units and external main storage cabinets through a support controller. The SSP controls up to four system consoles, four diskette drive units, and has a communications interface for remote diagnostic operation.

The SSP includes a desk-style cabinet that contains a processing unit, an interpretive macroprocessor, 131K bytes of addressable storage, a Diskette Direct Memory Access (DDMA) interface, a maintenance interface adapter, and a system interface adapter. The SSP interfaces with the central complex cabinet support controller through the maintenance interface adapter. It interfaces with the diskette drive units through the DDMA and with the system consoles through the system interface adapter. Figure 5-1 shows a block diagram of the SSP.

All 1100/70 System maintenance and system control operations are controlled via the SSP from a display terminal keyboard located on the system console. The SSP software processes display terminal keyboard commands by passing control information to the various individual system components and retrieving the desired results.

The 1100/70 System requires the SSP to perform several important system functions. These include: partitioning, system Initial Program Load (IPL), automatic recovery, CPU fault analysis, performance monitoring, logic analyzer control, and system control. A redundant interface to a second SSP is provided so that a primary SSP and a backup SSP can be configured in a system.

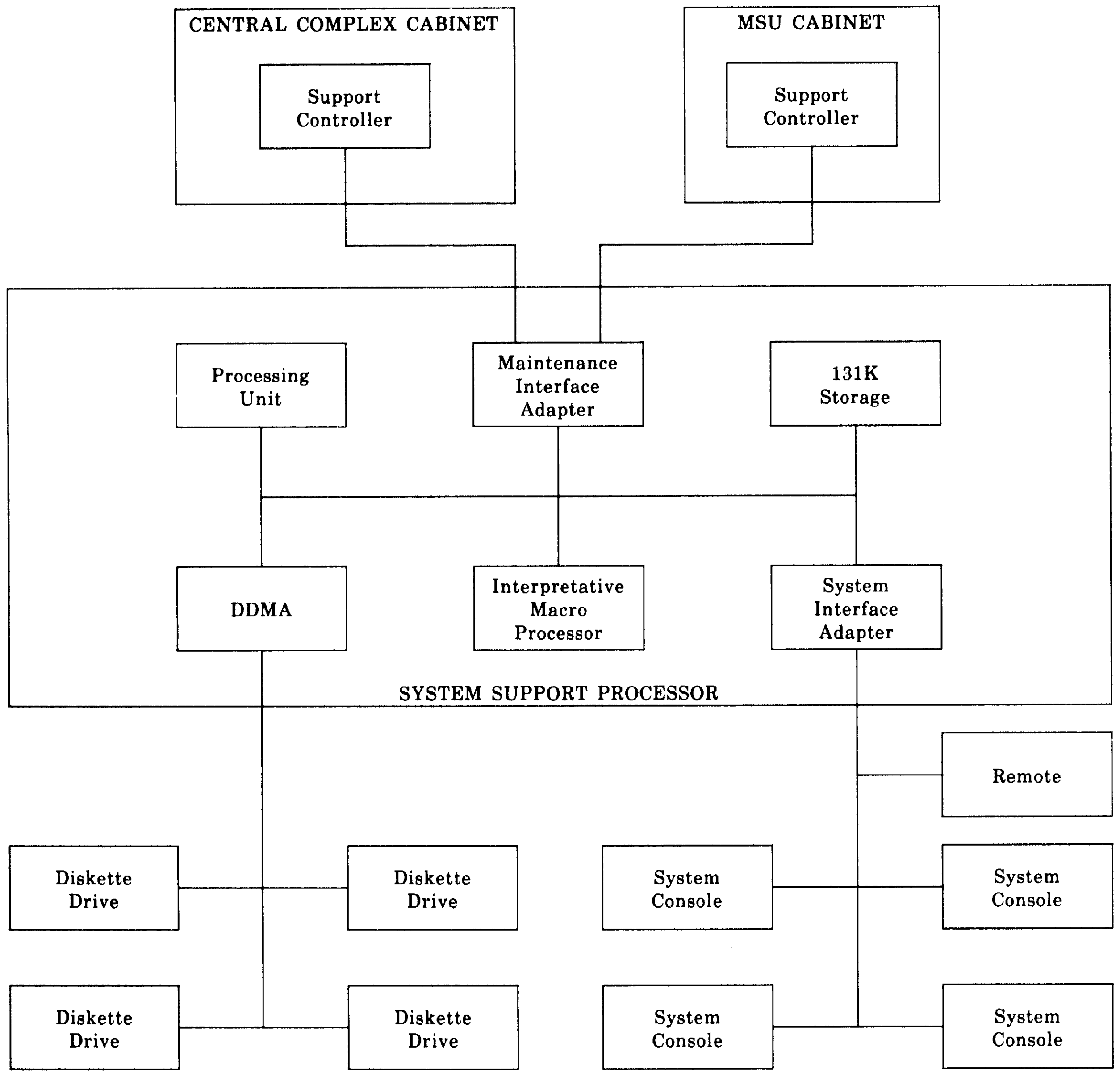


Figure 5–1. SSP Configuration and Interfaces

5.2. System Communications

The Executive software sends information to and receives information from the SSP software. The Executive software interrupts the SSP software by executing the Initiate Maintenance Interrupt (IMI) instruction. The IMI instruction loads an operand into a CPU hardware register reserved for SSP usage and interrupts the SSP. The SSP scans the reserved register and executes the operation specified by the operand. The SSP software interrupts the Executive software via the soft SSP interrupt. The soft SSP interrupt actually interrupts the Executive software and jumps to the fixed storage location, MSR+233_8. A status word associated with the interrupt is stored in the General Stack Register (GRS). The soft SSP interrupt is called a soft interrupt to distinguish it from a hard SSP interrupt. A soft SSP interrupt is executed by the Executive software. A hard SSP interrupt is executed by CPU microcode and is transparent to the Executive software.

The SSP software controls the system hardware by two basic mechanisms: a direct write and a hard interrupt. A direct write transfers data from the SSP directly to the hardware. Data is then either loaded into a register or decoded to initiate a hardware operation. A hard interrupt is executed by the SSP as a direct write to a CPU. The direct write for a hard interrupt loads one reserved CPU register with data, loads another CPU register with the starting microinstruction address of the function to be performed, and initiates a hard interrupt request. When the CPU microcode completes execution of the present Series 1100 instruction, the hard interrupt request is detected and the microcode jumps to the microinstruction routine specified by the SSP. The SSP microinstruction routine is executed and then execution of Series 1100 instructions is continued. The execution of the microinstruction routine is transparent to the user and Executive instruction stream being executed when the hard interrupt is accepted.

5.3. Functions Controlled by the SSP

The following subsections briefly describe several of the functions controlled by the SSP. The 1100/70 Systems SSP Supervisor program controls the execution of these functions and interfaces the application programs to the Series 1100 Systems and SSP peripheral devices.

5.3.1. Partitioning

The partitioning function provides the capability to assign individual central complex units, word-oriented peripheral subsystems, and SSPs (software partitionable only) to any one of three independent smaller systems (called applications), to isolate a unit from an application for maintenance, or to place a unit offline. Partitioning also provides for control of byte channel transfer switches to select alternate paths or an offline condition for byte-oriented subsystems.

The initial program load function provides the capability to set Module Select Register (MSR) values, select initial load paths, and initiate the initial load operation for the partitioning application. The MSR selects the section of main storage in which the fixed interrupt addresses are located, and the location in main storage where the instruction execution sequence is initiated on an initial load.

Partitioning restrictions are:

- partitioning is done by clusters,
- if both CPUs in a cluster are down, the IOUs in that cluster are also down,
- an SSP may be assigned to more than one application, and
- main storage is partitioned in 262K-word increments.

5.3.2. System Initial Program Load

The initial program load consists of initializing the 1100/70 Systems central complex hardware and initiating execution of the Executive software.

5.3.3. Automatic Recovery

The automatic recovery function provides the system with an automatic system recovery capability for each system application. When the automatic recovery function is enabled, and the automatic recovery timer is not reset within the preset time interval by the Executive System, the SSP clears, reloads, and reinitiates the system application. Alternate recovery paths are automatically initiated when the attempted recovery fails. The automatic recovery function also provides for software resetting of the automatic recovery timer and for selection of the automatic recovery path to be used by the next recovery attempt.

5.3.4. CPU Fault Analysis

When a control store parity error occurs, the SSP compares the suspected faulty portion with a known good portion, corrects it (if necessary), and initiates a retry sequence. The SSP software logs the necessary information pertaining to the error, and may use this information to isolate the cause of the error.

5.3.5. Performance Monitoring

Performance monitoring provides an 1100/70 System with the capability to collect system profile hardware data and software performance data. The data generates system-use profiles that enable the site manager to balance daily workloads and analyze long-term performance trends. The hardware related data uses individual hardware signals such as processor-busy, and individual I/O channel active states. The software related data uses Executive elements, system processor, or user program states. The data provides an important complement to the system log data yet requires negligible system overhead. The signals in the cabinet containing the CPU are monitored by this feature. If a cabinet does not contain a CPU (e.g., 1x2 configuration), the IOU signals in that cabinet cannot be monitored. Where more detailed performance data is required, the profile data can be combined with the performance data collected by the Software Instrumentation Package (SIP). Although SIP need not be configured to collect the profile data produced by the performance monitoring feature.

The performance monitor is turned on at bootstrap initialization by the Executive. The parameters that are sampled and counted by the CPU are presented to the SSP via the central complex scan set matrix. The SSP software periodically reads the data in the scan set matrix and presents it to the Executive upon request. If the SSP has more than one CPU in an application, it reads the data from each CPU and the Executive stores the data in the system log. Data collection and report generation are handled by the system as required and requested by the user.

Table 5-1 lists the parameters measured by the performance monitor feature. The Guard Mode or User Time parameter is counted when the Privileged Instruction and GRS Protection designator (D2) is set. Input/output channel active signals are measured for word channels by $\boxed{\text{OR}}$ing input active and output active. For the block multiplexer channels, the channel is considered active whenever it is hooked up to a control unit or device and is unavailable for use. Three additional designator bits (Software Performance Monitor designators, bits D24, D25, and D26) are used for software monitoring. These three designator bits are software controllable via the Load Designator Register (LD) instruction and, when used in conjunction with D2, provide 16 software state indicators in the User and Executive modes.

Eighteen signals are required to construct the performance parameters. These signals are sampled periodically (every 475 microseconds) by the CPU and their state captured in a holding register. The contents of the holding register is then evaluated and the appropriate counters are incremented. Thirty-two 16-bit counters are individually updated. The Elapsed Time counter counts the total number of samples taken and forms a base for all other counters. A read of the 32 counters (64 bytes) must be made within 30 second intervals to avoid counter overflow. The Executive requests the data at 25 second intervals. The 64-byte buffers are time stamped by the SSP software and returned to the Executive where an entry is created for the system log. The performance data is directed to the system log file continuously after bootstrap initialization of the Executive System. In addition, if SIP is configured and collecting data, the hardware monitor data is included with the other SIP data.

A data reduction and report generation software processor (HMLOG) also builds a performance monitor data file from the log entries and generates reports for the user on demand.

Table 5-1. System Profile Parameters

Designator Bits	IOU Without Expansion	Two Word Channel and One Block Multiplexer Channel Module IOU Expansion	Two Block Multiplexer Channels and One Word Channel Module IOU Expansion	Four Block Multiplexer Module IOU Expansion
D2 D24 D25 D26				
NA	Elapsed Time			
0 1 1 1	EXEC Software State			
1 X X X	Guard Mode Time			
NA	I/O Word Channel A Interface 0	I/O Word Channel A Interface 0	I/O Word Channel A Interface 0	
NA	I/O Word Channel A Interface 1	I/O Word Channel A Interface 1	I/O Word Channel A Interface 1	
NA	I/O Word Channel A Interface 2	I/O Word Channel A Interface 2	I/O Word Channel A Interface 2	
NA	I/O Word Channel A Interface 3	I/O Word Channel A Interface 3	I/O Word Channel A Interface 3	
NA	Not Used	I/O Word Channel B Interface 0	I/O Word Channel B Interface 0	Not Used
NA	Not Used	I/O Word Channel B Interface 1	I/O Word Channel B Interface 1	Not Used

Table 5-1. System Profile Parameters (continued)

Designator Bits	IOU Without Expansion	Two Word Channel and One Block Multiplexer Channel Module IOU Expansion	Two Block Multiplexer Channels and One Word Channel Module IOU Expansion	Four Block Multiplexer Module IOU Expansion
D2 D24 D25 D26				
NA	Not Used	I/O Word Channel B Interface 2	I/O Word Channel B Interface 2	Not Used
NA	Not Used	I/O Word Channel B Interface 3	I/O Word Channel B Interface 3	Not Used
NA	Not Used	I/O Word Channel C Interface 0	I/O Block Multiplexer C Active	I/O Block Multiplexer C Active
NA	Not Used	I/O Word Channel C Interface 1	Not Used	I/O Block Multiplexer D Active
NA	Not Used	I/O Word Channel C Interface 2	Not Used	I/O Block Multiplexer E Active
NA	Not Used	I/O Word Channel C Interface 3	Not Used	Not Used
NA	I/O Block Multiplexer A Active	I/O Block Multiplexer A Active	I/O Block Multiplexer A Active	I/O Block Multiplexer A Active
NA	Not Used	I/O Block Multiplexer B Active	I/O Block Multiplexer B Active	I/O Block Multiplexer B Active
0 0 0 0	Highest priority EXEC (interrupts logically disabled), I/O control interrupt processing, console, dispatcher, expool, register save and restores.			
0 0 0 1	Program file load, contingency handler, expool monitor, I/O errors, x-keyin.			
0 0 1 0	Swapfile management, console, error handling, task termination, symbionts, TIP compool and file control.			
0 0 1 1	Mass storage allocation, I/O logging. Tip message handler and timer.			

Table 5-1. System Profile Parameters (continued)

Designator Bits				IOU Without Expansion	Two Word Channel and One Block Multiplexer Channel Module IOU Expansion	Two Block Multiplexer Channels and One Word Channel Module IOU Expansion	Four Block Multiplexer Module IOU Expansion
D2	D24	D25	D26				
0	1	0	0	Dynamic allocator, TIP scheduler.			
0	1	0	1	Accounting and logging.			
0	1	1	0	Coarse scheduling, facilities, security, poolback, directory control, periodic adjustment.			
0	1	1	1	CPU Idle - Idle subroutine.			
1	0	0	0	Real-Time - User activity with real-time privilege enabled.			
1	0	0	1	ESI Completion Activities - User activities that complete the interrupt handling for user communications I/O.			
1	0	1	0	Reserved (CMS).			
1	0	1	1	DMS 1100.			
1	1	0	0	TIP - Transaction programs.			
1	1	0	1	Deadline Batch - A batch run that is afforded certain scheduling priorities to assure run completion by a prespecified time.			
1	1	1	0	Demand - A run initiated from an interactive terminal.			
1	1	1	1	Batch - A run that does not have interactive requirements.			

5.3.6. Logic Analyzer Control

The logic analyzer is a maintenance tool which provides the capability for sampling and recording logic signals at discrete intervals of time. The SSP has control over the starting, stopping, and sampling rate, of logic signal recording. The SSP also has access to the recorded signal data via normal support controller read operations. One logic analyzer per CPU is provided. This capability is used by the customer engineer.

5.3.7. System Control

The SSP provides a software system operator panel and display that provides the following capabilities:

- Clear/reset
- Enabling and disabling halt and jump selects
- Tape and disk-drum initial load path definitions
- Automatic recovery enable/disable
- Breakpoint address selection
- CPU start/stop control
- Sets MSR value

6. Storage Systems

6.1. General

The internal storage system has 524,288 (524K) words expandable up to 4,194,304 (4194K) words of main storage per MSU (a total of two MSU per system for a maximum of 8,388,608 (8388K) words. The external main storage unit cabinet has 1,048,576 words expandable up to 8388K words of main storage (a maximum of two cabinets per system for a total of 8,388,608 words of storage). Internal storage cannot be used with external storage. The SIUs can contain 2048 (2K) or 8192 (8K) words of high-speed buffer storage, (a total of four SIUs per system for a maximum of 32,768 words) and 512 words of control registers.

The Main Storage Units (MSUs) provide storage for instructions and data words. The Storage Interface Units (SIUs) provide high-speed buffer storage between the MSUs and the Central Processing Unit (CPU). The 512 addressable control words exist as 128 addressable control registers in the control section of each CPU to provide fast access storage for data and control words.

6.2. Storage Interface Unit

An SIU is required for each CPU in a multiprocessing system and dual processing system, and connects to the CPU and both MSUs to form shared backing storage. The SIU provides 2K and 8K words of buffer storage.

The SIU is a high-speed buffer which reduces overall storage delay time. The SIU is placed between the system CPU and MSU. Each word of data that the CPU acquires from the MSU is stored in the SIU data buffer with three adjacent words from the same four-word block in the MSU. Subsequent read references by the CPU for a data word from that same block, or from any other block that is resident in the SIU (i.e., a read-hit), is satisfied by the SIU instead of by the MSU. Resident blocks of data in the buffer are displaced by new blocks from the MSU by the use of a Paired Least-Recently-Used (PLRU) age algorithm.

SIU references require much less access time than MSU references, which means that significant time savings are realized each time a requested word of data is retrieved from SIU buffer storage instead of from main storage.

The buffer-based system is effective because of two basic properties of executing programs:

1. Instructions and data are likely to be reused.

2. The instructions and data used are likely to be near instructions and data that have been previously used.

Each time a read reference is made to the MSU, the four words contained in the data block are transmitted in word-serial fashion to the SIU buffer, but only the requested word is transferred to the CPU. All four words are transferred during a single main storage cycle. For a write reference, only one word of data is transferred from the CPU to the MSU during a single main storage cycle.

Each data block in the SIU data buffer is accompanied by an associated block address, associated age information, and associated degrade/valid information. The block age information determines which block within an SIU set is the Least Recently Used (LRU). The degrade/valid information for each block indicates whether the block contains corrupted data or non-valid data, respectively. Degrade implies that a portion of the SIU is unavailable.

Three bits of data, plus parity, are used to specify the relative ages of the block within a set. This data is stored in the age buffer as part of the tag information for each set. The age data specifies which of the blocks within a set is the LRU and, therefore, the most likely candidate for replacement in the event of a read-miss operation.

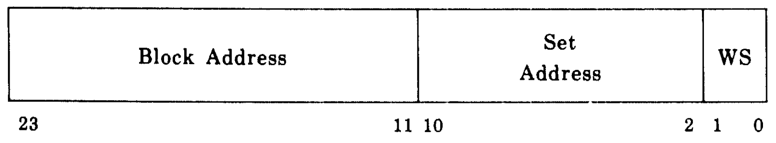
In order to maintain data integrity within the system, the MSUs provide invalidate address information to each connecting SIU; this is used to invalidate data that resides in the SIU buffer storage if data in the same block address is altered in main storage by another accessing module.

The SIU contains a support controller interface that provides scan/set capability for selected elements with the SIU. The SIU also contains an interface for connecting to a system synchronizing clock. The clock is used to synchronize the operations of the various components in the system. The SIU uses the clock to control the requests to main storage, to control requestor interface activity, and to control the invalidate interface, as well as the internal timing of the SIU.

The SIU employs a set-associative addressing structure. The 2K word SIU is divided into 512 sets, each containing one block. Each block contains 4K words for search and hit operation. The 8K-word SIU is divided into 512 sets, each containing four blocks. Each block contains 4K words for both search and replace operations.

The SIU interfaces with up to two MSUs. The MSUs are divided into sets: 512 sets in a single MSU configuration, and 256 sets each in a dual MSU configuration.

Any one of the 4-word blocks in the MSUs may be transferred into the corresponding set in the SIU. Each block remains in the SIU until it becomes the LRU block in its set and is displaced by a new block of data brought in from the MSU. At any point in time, a maximum of 16 words per MSU set may be resident in the SIU. The storage address format is:



where:

Bits 23–11	Block address selects a 4-word block within a set.
Bits 10–2	Set address selects one of the 512 sets.
Bits 1–0	Word Selects (WS) one of four words in a block.

NOTE: Address bit 23 is used as MSU select bit for dual MSU configuration.

Storage addresses presented to the SIU with a storage request are examined to determine if the referenced block is resident in the SIU. If the desired block for a read operation is not resident in the SIU (i.e., a read-miss), the following will take place:

- A request is presented to the MSU for the 4-word block of data.
- The desired word is presented to the CPU when it is received from the MSU.
- The block of data is stored into the SIU buffer in place of the oldest block of data in the affected set.
- The block address for the new data is stored in the SIU tag buffer in place of the address of the displaced block.
- The associated age information for the affected set is altered, with the newly-stored block flagged as the most current.

If the desired block for a read operation is resident within the SIU (a read-hit), the following will take place:

- The desired word is presented to the CPU.
- The age information of the affected SIU set is altered as required to indicate the relative ages of the blocks in the set, with the referenced block flagged as the most current.

A write-miss reference bypasses the SIU data buffer, and the write data is stored directly into the MSU; a new resident block is not established in the SIU.

A write-hit reference affects the SIU as follows:

- For a full- or half-word write, the data word is written over the existing resident word in the data buffer, the age information for the set is altered, and the word is also written into main storage.
- For a partial-word write other than half-word, the word bypasses the data buffer, is written in main storage, and the resident block in the SIU is marked invalid.

6.2.1. Functional Units

The SIU has nine major functional sections: the CPU interface, the resident directory and associated control, the data buffer, the main storage interface, a system support interface, an invalidate interface from each MSU, the SIU main control section, and a system clock interface.

6.2.1.1. CPU Interface Section

The CPU interface is a single-port word interface operating with Emitter Coupled Logic (ECL) signal levels. The interface is compatible with the CPU interface in the MSU. This allows a system to be configured without an SIU, if desired, by connecting the CPU directly to the MSU.

The CPU interface includes:

- registers for request data (i.e., address information, write data, and write control information),
- parity checking of request data, and
- parity generation for block addresses that are to be stored in the tag buffer on read-miss operations.

6.2.1.2. Resident Directory and Directory Control Section

The resident directory section is located functionally between the CPU interface and the data buffer section. The resident directory consists of:

- a tag buffer (for storing the addresses of blocks that are resident in the data buffer),
- an age buffer, and
- a degrade/valid buffer.

The directory control portion contains the timing and control to operate the data buffer, the control for data transfers to/from the main storage interface section, and the control to operate the tag buffer, the age buffer, and the degrade/valid buffer. Errors in the block and aging logic in the SIU will result in the degradation of one or more blocks within the SIU, and generate a successful instruction or at least successful retry. A mechanism in the Executive makes degradation available to the system operator. This mechanism counts individual block degrade reports and calculates the total percentage of buffer storage currently degraded. Messages are issued at specific degradation levels that indicate the percentage of buffer storage degradation. This section also checks the following for correct parity:

- block addresses of resident blocks,
- block degrade/valid information, and
- block age data.

6.2.1.3. Main Storage Interface Section

The main storage interface section controls the transfer of data between the SIU and one or two MSUs, and detects and reports errors on the SIU/MSU interfaces. The interfaces operate with ECL signal levels.

A read data request is accepted by the CPU interface section and is passed to the resident directory section where the address is inspected. If the addressed data word is not resident in the data buffer section, the resident directory section turns control over to the main storage interface section. The main storage interface section initiates a read request to the selected MSU. The interface to each MSU is one word wide. The four words from the requested block in the MSU are transmitted to the SIU in a word-serial fashion.

All write operations result in a request to main storage via the main storage interface section. Full- or half-word writes to resident addresses are write-through (written in the SIU buffer and in main storage); other partial-word writes to resident addresses are write-past (invalidated in the SIU buffer and written in main storage). Writes to nonresident addresses are write-bypass (written in main storage without affecting the SIU data buffer).

6.2.1.4. SIU Main Control Section

The SIU main control section provides over-all control for interaction between the various sections of the SIU. In addition, the main control section performs the following functions:

- formatting and reporting of all errors,
- processing of Select Interrupt Location (SIL) operations,
- handling of storage lock operations, and
- buffer degradation.

6.2.1.5. System Support Interface

The system support interface in the SIU is compatible with the support controller. The SIU responds to scan/set signals received over the system support interface section of the central complex.

Each SIU provides four bytes of scan data containing the serial number, buffer size, and revision level of that unique SIU.

6.2.1.6. Invalidate Interface

Each MSU in a multiprocessor system stores data for, and communicates with, all SIUs and all IOUs. Data is kept current in the SIUs and the MSU by a write-through mechanism and by a special invalidate interface between the SIU and the MSUs. The SIU accepts a write request, passes the request information to the MSU, and presents the acknowledge to the CPU. The MSU will notify the other SIU via the invalidate interface that a data word has been altered and includes the address of the altered block. This causes the SIU in the same application to flag the altered block as invalid if the block is resident. The block remains invalid in that SIU until a new block is brought in as a replacement.

When an IOU performs a write reference to an MSU, an invalidate request is also transmitted to the SIUs in the system; the invalidate request, along with the affected address, invalidates resident blocks.

Simultaneous invalidate requests received from separate MSUs at the invalidate interface are honored on a rotational-priority basis.

6.2.1.7. System Clock Interface

The system clock interface in the SIU is designed to receive ECL signal levels. The clock is used to control the main storage interface, the CPU interface, and the invalidate interface sections, in order to minimize the problems that are involved with asynchronous control among different components within the 1100/70 System.

6.2.1.8. Data Buffer

The data buffer section contains the integrated storage for the SIU buffer. The buffer contains 2,048 or 8,192 words and is controlled by the directory control section.

6.2.2. Performance

The SIU is capable of cycling at a rate of one read-hit reference every 116 nanoseconds measured at the SIU-CPU interface; read-hit access time is also 116 nanoseconds.

6.2.3. Error Detection and Reporting

The SIU provides the means for detecting and reporting faults in the SIU/MSU requester complex. When an error is detected, the CPU is notified via error-reporting mechanisms. A detected fault is reported only once. All requests are completed even though fault conditions may be encountered. Data transfers that may be affected by fault conditions are controlled as described in the sections that follow.

The following are checked for error conditions:

- Requester address parity.
- Requester control parity (write controls, segment lock, SIL, status reference).
- Requester write data parity.
- Read data parity, SIU data buffer.
- Block address parity, SIU tag buffer.
- Block degrade/valid parity, SIU degrade/valid buffer.
- Block age parity, SIU age buffer.
- Read data parity, main storage interface.
- Invalidate interface address parity.
- MSU read data Error Correction Code (ECC) and partial-word write ECC (checked by MSU, errors reported to SIU).
- Address parity, control parity, write data parity, invalid address (checked by MSU, errors reported to SIU).

6.2.3.1. Storage Check Interrupt Status Request Formats

Errors that are detected in the CPU interface section are reported to the CPU with the ACKNOWLEDGE 2 (ACK 2) signal, which is returned to the CPU. Errors that are detected by the MSU when a request is received (address parity, etc), are also returned to the CPU with ACK 2 if a read reference was requested. Errors detected elsewhere and MSU errors for a write reference are reported to the CPU with a Storage Check Interrupt Status Request (SCISR) and a status word whose format is described in Figures 6-1 and 6-2. The CPU and its associated software is responsible for processing the error and taking appropriate action.

Figure 6-1 illustrates the format for reporting internal SIU errors. Figure 6-2 illustrates the format for reporting errors on the SIU/MSU interface or for reporting maintenance read-miss operations. Bits 35-33 of each interrupt status word specify the format for that status word:

Bit 35	Bit 34	Bit 33	Interrupt Status Format
0	0	0	Internal SIU Check
0	0	1	SIU/MSU Interface Check or Maintenance Miss

MSU read of partial-word write ECC check SCISR interrupts are generated and formatted in the MSU and passed to the SIU for transmittal to the requester.

0	0	0	SSO	BDD	BCD	BBD	BAD	DCU	DCL	TBC	AGE	Age Bits	Requested Address
35	34	33	32	31	30	29	28	27	26	25	24	23 20	19 0

Figure 6-1. Internal SIU Storage Check Interrupt Status Format

where:

Bits 35-33	The code of 0_8 identifies the status word as an internal SIU check.
Bit 32	SCISR Status Overflow (SSO)
Bit 31	Block D Degraded (BDD)
Bit 30	Block C Degraded (BCD)
Bit 29	Block B Degraded (BBD)
Bit 28	Block A Degraded (BAD)
Bit 27	Data Check Upper (DCU) half word (data buffer)
Bit 26	Data Check Lower (DCL) half word (data buffer)
Bit 25	Tag Buffer Check (TBC) (block address or degrade/validity parity)
Bit 24	AGE parity check
Bits 23-0	Requested address if bit 24 is not set
Bits 23-20	Age Bits if bit 24 is set (4 bits including parity)
Bits 19-0	Requested Address lower 20 bits

NOTES:

1. *Requested address is used if bit 24 is not set; age bits are used if bit 24 is set.*

2. *For bits 32-24, the one state denotes the condition specified.*

0	0	1	SSO	SCD	IAC	RDR	MDM	ANA	DPC	ADC	WCC	Requested Address
35	34	33	32	31	30	29	28	27	26	25	24	23 0

Figure 6-2. SIU/MSU Interface Storage Check Interrupt Status Format

where:

Bits 35-33	The code of 1_8 identifies the status word as a SIU/MSU interface check.
Bit 32	SCISR Status Overflow (SSO)
Bit 31	Special Code Detected (SCD*)
Bit 30	Invalidate Address Check (IAC)
Bit 29	ReaD Request (RDR)
Bit 28	Maintenance Read Miss (MRM)
Bit 27	Address Not Available (ANA*)
Bit 26	Data Parity Check (DPC)
Bit 25	ADdress Check (ADC)
Bit 24	Write Control Check (WCC)
Bits 23-0	Requested Address or invalidate address

NOTES: *For bits 32-24, the one state denotes the condition specified.*

** SCD and ANA will be set together if a special code is detected on a partial-write request. ANA will be set only if Address Not Available is detected on a write request.*

6.2.3.2. CPU Errors

The following bits are received by the CPU interface in the SIU:

- 24 address bits with one parity bit
- 36 data bits with two parity bits (one parity bit for each 18-bit half word)
- Eleven control bits (eight write control, one SIL control, one Storage Lock control, one Storage Check Status Reference control) with one parity bit

When the SIU detects a control parity error, an address parity error, or a write data parity error, the request is aborted without any altered MSU or SIU data. The fault conditions are reported to requestor along with the ACK 2 signal. (See 8.3.6)

6.2.3.3. Internal SIU Errors

The error detection logic of the SIU detects parity faults in the data buffer, the tag buffer, the age buffer, and the degrade/valid buffer. If a parity error is detected at any of these points, an SCISR interrupt is generated and the failing blocks are disabled by the degrade flag. The degrade flag can be cleared only by a special upgrade sequence initiated by the support controller or by an SIL function initiated by the CPU.

Each 36-bit word in the data buffer has two parity bits (one parity bit for each 18-bit half word). Parity is checked on the requested word from the buffer on a read-hit operation.

Each block address in the tag buffer has one parity bit. Whenever the tag buffer is referenced, parity is checked on all of the non-degraded block addresses associated with the referenced set. If a parity error is detected, an SCISR interrupt is generated to report the fault, and the failing block or blocks are degraded.

Each block in the SIU has an associated degrade bit, an invalidate bit, and a parity bit in the degrade/valid buffer. Whenever the degrade/valid buffer is referenced, parity is checked on all groups of invalidate and degrade bits for the referenced set. If a parity error is detected on any of the blocks, an SCISR interrupt is generated, and the failing block or blocks are degraded.

Parity of the age data is checked whenever one of the 512 sets in the age buffer is referenced. If a parity error is detected, the following takes place:

- An SCISR interrupt is generated to inform the processing system of a probable failure in the replacement algorithm for that set.
- The entire set is degraded - this prevents repeated error reporting whenever that set is referenced in the future, assuming the parity error is the result of a solid hardware failure and not a temporary failure caused by noise, etc.

If a read-hit operation is in progress in the SIU and a tag buffer or data buffer parity error is detected, the following operations occur:

- The read-hit data is presented to the CPU as well as the normal ACK 2 signal (the data may not be correct).
- Following the ACK 2 signal, a READ-HIT ERROR DETECTED signal is presented to the CPU.
- The SIU automatically reinitiates the CPU request for which the error was detected, except that the request goes to main storage as if a read-miss operation were in progress.
- A second ACK 2 signal and the requested word of data is presented to the CPU when the MSU responds with read data.

The data buffer parity errors are invisible to the processing system once the fault has been reported, except as affecting data paths and access times for future references. Failing blocks will be degraded and isolated from the system via hardware control, and future references to those blocks will bypass the SIU.

6.2.3.4. SIU/Main Storage Interface Errors

If the CPU requests an address that is nonexistent or illegal, the SIU passes an ADDRESS NOT AVAILABLE (ANA) signal with ACK 2 to the CPU, if the request was for read data. If the MSU detects ANA on any of the three remaining words in that address block, the SIU invalidates that block. For write requests, the ANA signal causes a Delayed Check interrupt to be generated by the SIU.

The integrity of all information passed between the SIU and the MSU is protected by parity. The SIU sends eleven bits of control information plus one parity bit to the MSU. On each main storage reference, the MSU checks the controls for correct parity. If a parity error is detected, the reference is executed, a special error syndrome is stored in the addressed MSU location, and a SCISR interrupt is generated. The special error syndrome stored in the addressed MSU location flags that location as one containing corrupted data.

The SIU sends 36 bits of write data plus two bits of parity to the MSU (one parity bit per 18-bit half word). Write data parity errors detected by the MSU are reported to the SIU via SCISR generation. The MSU stores the special error syndrome, along with the data as received, into the addressed location.

The SIU sends 24 bits of address plus one parity bit to the MSU. On each main storage reference, the MSU checks the address for correct parity. If a parity error is detected, the reference is immediately aborted and the error is reported to the SIU by way of an address parity error. On a write request, the MSU immediately goes offline to protect against the undetected corruption of data. The MSU returns ADDRESS NOT AVAILABLE for all subsequent storage requests (for all CPUs).

Each 36-bit word of read data from the MSU is accompanied by two bits of parity, one for each 18-bit half word. If the SIU detects a read data parity error on one or more of the words received during a read reference, an SCISR interrupt is generated and the block containing the failing word or words is invalidated. The data word that is presented to the CPU includes parity as received from the MSU.

6.2.3.5. Main Storage Errors

The MSU generates and stores 7-bit ECC information for each word of data which enters the storage array. On a read reference, which involves four words of data, the ECC is checked for each word retrieved from storage. Single-bit errors in the data are corrected by the MSU. Multiple-bit errors detected for any of the data words are not corrected and the data is transmitted in its uncorrected state.

The requested data word is presented to the SIU in both ECC error cases - the corrected word in the first case, or the uncorrected word in the multiple-bit uncorrectable error case.

On a partial-word write request, the MSU retrieves the data word that is to be altered from the storage and checks the ECC. If a single-bit error is detected, the error is corrected, the partial-word write is executed, and the altered data word and new ECC are written into storage. If a multiple-bit error is detected, the partial-word write is not executed.

Detection of partial-word write errors and read data ECC errors are reported to the SIU via the SCISR interrupt.

Detection of special codes in read data causes the MSU to return both Multiple Uncorrectable Error (MUE) and ANA to the SIU. If the special code was detected on the requested word, the SIU passes both the MUE and ANA to the CPU. Special codes detected on any of the four words within a block cause the SIU to invalidate that block. The CPU is notified only if both MUE and ANA occurred on the requested word. Detection of special codes on partial writes causes the MSU to return MUE and ANA to the SIU. In this case, the SIU generates a Delayed Check interrupt to the CPU.

6.2.3.6. Invalidate Interface Errors

The SIU checks the parity on address information received on the invalidate address lines from the two MSU interfaces. In the event of a parity error, the SIU will:

- Generate an SCISR interrupt.
- Invalidate the entire buffer storage in order to protect data integrity in the system.

6.2.3.7. Maintenance Interface Errors

Errors detected at the system support interface (parity errors, invalid commands, etc.) are reported to the System Support Processor (SSP). Support controller interface faults will not generate SCISR interrupts.

By enabling the SIL reference control signal when presenting a request to the SIU, the CPU has the capability of selectively degrading and upgrading blocks within the SIU; forcing incorrect parity on SIU-resident address, data, age, and degrade/validity information; and placing the SIU in the dummy MSU mode.

The SIL data and address format for SIL references is:

Select Interrupt Location Reference Format

Not Used	DMME	DMMD	MME	MMD	Not Used	MRFC	MR Data	NU
35 30	29	28	27	26	25 14	13 11	10 2	1 0

where:

Bits 35–30	Not used
Bit 29	Dummy MSU Mode Enable (DMME)
Bit 28	Dummy MSU Mode Data (DMMD)
Bit 27	Maintenance read Miss Enable (MME)
Bit 26	Maintenance read Miss Data (MMD)
Bits 25–14	Not used
Bits 13–11	Maintenance and Reliability Function Code (MRFC)
Bits 10–2	Maintenance and Reliability Data (MR Data)
Bits 1–0	Not Used (NU)

The SIU processes an SIL reference according to the following:

- If bit 27 is set.
- If bit 27 is set and bit 26 is clear, the maintenance read-miss mode is deactivated.
- If bit 29 is set and bit 28 is clear, the dummy MSU mode is deactivated.
- If write data bit 29 is set and write data bit 28 is set, the dummy MSU mode is activated in the SIU. In the dummy MSU mode, all requests to main storage are inhibited. Read-miss operations in this mode will establish resident SIU blocks, but data for the resident blocks are supplied by subsequent full- or half-word hit references by the CPU as for normal write-hit references in the SIU. Read-hit references function as per normal, to provide data turnaround checking in the SIU.

Maintenance and reliability functions are performed at the SIU by use of the SIL reference, according to SIL bits 13-11 and bits 10-2. The following items describe the maintenance and reliability functions:

- If bits 13-11 equal 0, 1, 6, or 7, no maintainability and reliability function is performed. (No action is taken in the SIU; these codes are used by the MSU.)
- If bits 13-11 equal 2, bit 3 set forces an address parity error on the invalidate interface on the next incoming invalidate request.
- If bits 13-11 equal 2, bit 4 set forces the next block address that is loaded into the tag buffer on a read-miss operation to be loaded with incorrect parity.
- If bits 13-11 equal 2, bit 5 set forces a parity error to be loaded into the degrade/valid buffer for the immigrant block on the next read-miss operation.
- If bits 13-11 equal 2, bit 6 forces a control parity error on the next main storage request.
- If bits 13-11 equal 2, bit 7 set forces an address parity error on the next main storage request.
- If all of the write controls are set and bits 13-11 equal 2, SIL data bit 8 set forces a lower half-word write data parity error on the next main storage request; bit 9 set forces an upper half-word write data parity error.
- If bits 13-11 equal 3, bits 10-3 force the four data words loaded into the data buffer on the next read-miss operation to be loaded with incorrect parity according to the following:

    1. Bit 3 set forces the lower half of word 0 to be loaded with incorrect parity.
    2. Bit 4 set forces the upper half of word 0 to be loaded with incorrect parity.
    3. Bit 5 set forces the lower half of word 1 to be loaded with incorrect parity.
    4. Bit 6 set forces the upper half of word 1 to be loaded with incorrect parity.
    5. Bit 7 set forces the lower half of word 2 to be loaded with incorrect parity.
    6. Bit 8 set forces the upper half of word 2 to be loaded with incorrect parity.
    7. Bit 9 set forces the lower half of word 3 to be loaded with incorrect parity.
    8. Bit 10 set forces the upper half of word 3 to be loaded with incorrect parity.

- If bits 13-11 equal 4, write data bits 10-2 will determine the age data for the set indicated by the address accompanying the SIL reference. The age will be written according to the following (bits 10-3 are mutually exclusive; bit 2 may be in addition to bits 10-3):

  1. Bit 3 set forces the age information to 0_8.
  2. Bit 4 set forces the age information to 1_8.
  3. Bit 5 set forces the age information to 2_8.
  4. Bit 6 set forces the age information to 3_8.
  5. Bit 7 set forces the age information to 4_8.
  6. Bit 8 set forces the age information to 5_8.
  7. Bit 9 set forces the age information to 6_8.
  8. Bit 10 set forces the age information to 7_8.
  9. Write data bit 2 set forces bad age parity.

- If the write controls are set and bits 13-11 equal 5, bits 6-3 will control the upgrading and degrading of the SIL blocks according to the following (items 3, 4, 5, 6, 7, and 8 applicable only for an expanded 8K word SIU).

  1. If bit 3 is set, block A is degraded.
  2. If bit 3 is clear, block A is upgraded.
  3. If bit 4 is set, block B is degraded.
  4. If bit 4 is clear, block B is upgraded.
  5. If bit 5 is set, block C is degraded.
  6. If bit 5 is clear, block C is upgraded.
  7. If bit 6 is set, block D is degraded.
  8. If bit 6 is clear, block D is upgraded.

The set being degraded is selected by bits 10-2 of the address data presented with the SIL reference request.

6.2.4. Storage Lock

When the STORAGE LOCK control signal is presented with a SIU request, the SIU passes the request and SEGMENT LOCK signal to the MSU. Storage lock references will generate a request to the MSU; the requested block is invalidated if it is resident in the SIU. The MSU verifies requests in that application solely from the CPU that presented the STORAGE LOCK signal until that CPU presents a request without the STORAGE LOCK control signal. The MSU continues to honor requests from other CPUs assigned to a different application.

6.3. Main Storage Unit

The internal main storage contains 524K 43-bit words consisting of 36 data bits and 7 Error Correction Code (ECC) bits. The MSU is expandable in 524K word increments, to a maximum of 4194K words. Two internal MSUs can be used for a maximum of 8388K words.

The external main storage contains 1048K 45-bit words consisting of 36 data bits, 7 ECC bits and 2 through-checking bits. The MSU is expandable in 1048K word increments, to a maximum of 4194 words. Two MSU cabinets can be used for a maximum of 8388K words. Configurations with three or four central complex cabinets require external main storage.

read cycle	580 nanoseconds
write cycle	580 nanoseconds
corrected read access	625 nanoseconds

partial write cycle	812 nanoseconds
refresh cycle	580 nanoseconds
refresh period (typically)	26 microseconds

6.3.1. Modes of Operation

The online/maintenance (offline) modes are selectable via the SSP. Online availability is indicated to the CPU by the presence of the storage AVAILABLE signal. The MSU contains an internal exerciser for offline operation. This exerciser is controlled via the SSP.

6.3.1.1. Online Operation

The MSU is capable of performing the following functions in the online mode:

- Read

  When the MSU is connected to the CPU or IOU, the MSU reads a single 43-bit word (36 data + 7 ECC) from storage and transfers the corrected 36 bits of information to the requester along with two half-word parity bits.

  When the MSU is connected to the SIU, the MSU reads a full 4-word block of data from storage. Each 36-bit word with two half-word parity bits are then sent to the requester via four sequential transfers. The four transfers contain the corrected first word followed by the remaining corrected three words.

- Write

  The MSU writes a single 36-bit word of information plus 7 ECC bits (plus 2 bits through check on external storage) into the specified storage location.

- Partial Write

  The MSU writes partial-word information into the specified storage location. A data word is capable of being written within any sector, defined by dividing the word into sixth words (6 bits), quarter words (9 bits), third words (12 bits), half words (18 bits), or full words (36 bits). The control of the sector to be written is provided by eight write-control signals which are presented to the MSU over eight interface lines during each cycle. Each write control signal controls a fixed field within the data word. Activating combinations of these controls allows writing into the various word sectors. If none of the write control signals are active, a read operation is specified. The partial-write information is written in the specified storage location on the following boundaries:

7	6	5	4	3	2	1	0
35 30	29 27	26 24	23 18	17 12	11 9	8 6	5 0

- Refresh

  The MSU is a volatile storage device and requires a refresh cycle to maintain the data. Each MSU generates its own refresh cycle. Refresh signals are generated at 26-microsecond intervals. The refresh cycle is continually sequenced through the storage unit such that 1/128 of the storage unit is refreshed during each refresh cycle.

6.3.1.2. Offline Operations

The MSU is capable of performing the following exerciser functions in the offline mode via the SSP:

- Read

  In this mode, the internal exerciser continuously reads from storage. The exerciser sequentially addresses through all available storage locations.

- Write

  In this mode, the internal exerciser continuously writes the selected data pattern into storage. The exerciser sequentially addresses through all available storage locations.

- Single Step

  When controlled under single step operation, the exerciser advances one address if appropriate and executes one MSU cycle each time it is started.

- Pattern Selection

  The exerciser has various selected patterns available for maintenance of the storage unit. These are based upon the anticipated worst case data patterns, which depend upon switching of data in combination with different fixed address sequences. The patterns include write/read, write/read alternating data, 2 pass, address convergence, march, random address, and partial-write test. These patterns are selected either automatically by the exerciser or manually under operator control. Additionally, control functions such as starting, stopping, stopping on errors, and jamming address bits are also performed.

6.3.2. Operational Characteristics

6.3.2.1. Data Format

The MSU uses one word read transfer when communicating with CPUs or IOUs but four single-word read transfers when communicating with SIUs. In the situation of four single-word read transfers the information is contained in a four-word block and the first word from the block corresponds to the location as selected by the SIU. The remaining three words are transferred in ascending order as through a ring counter sequence in order to complete the four-word block transfer to the SIU. In all situations a single word write transfer is used. The internal word format consists of 36 data bits plus 7 ECC bits. The word interface format is:

P 1	D35 Data Bits D18	P 0	D17 Data Bits D0

P1 Parity bits for data bits 35–18.

P0 Parity bits for data bits 17–0.

6.3.2.2. Address Format

Twenty-four lines are available for addressing. Twenty of these lines specify the address for the four-word data block for internal storage. The external storage uses twenty-three lines to specify the address for the four-word data block. Parity is checked on all 24 lines. The storage units are selected for sequential addressing of units through an 8-million-word boundary or interleaving of units about the 8-million-word boundary. Bit 23 is used by the CPU to direct the main storage request to the proper unit for the non-interleaved operation. Bit 2 is used to direct the request in the internal storage interleaved mode (See Figures 6-3 and 6-4).

Storage Boundaries

MSU0 (Bit 23 = 0) 1048K				MSU1 (Bit 23 = 1) 1048K			
4th 262K	3rd 262K	2nd 262K	1st 262K	1st 262K	2nd 262K	3rd 262K	4th 262K

Address Format

MS	Not Used	Exp. Adrs	16K Adrs	Column Address	Row Address	FWB
23	22 20	19 18	17 16	15 9	8 2	1 0

Figure 6-3. Storage Boundary and Address Format for Internal Storage

where:

Bit 23	MSU Selection (MS)
Bits 22-20	Not Used
Bits 19-18	1 of 4 16K Expansion Address
Bits 17-16	1 of 4 16K Address
Bits 15-9	1 of 4 16K Column Address
Bits 8-2	1 of 4 16K Row Address
Bits 1-0	Four Word Block (FWB)

NOTE: Address bits 17, 18, and 19 are inverted internally when bit 23 = 0.

Storage Boundaries

MSU0 (Bit 23 = 0) 4194K								MSU1 (Bit 23 = 1) 4194K							
8th 524K	7th 524K	6th 524K	5th 524K	4th 524K	3rd 524K	2nd 524K	1st 524K	1st 524K	2nd 524K	3rd 524K	4th 524K	5th 524K	6th 524K	7th 524K	8th 524K
Bank 0				Bank 1				Bank 0				Bank 1			

Address Format

MS	NU	A	1 of 8 Part	Column Address	Row Address	FWB
23	22	21	20 18	17 10	9 2	1 0

Figure 6-4. Storage Boundary and Address Format for External Storage

where:

Bit 23	MSU Selection (MS)
Bit 22	Not Used (NU)
Bit 21	1 of 2 Address Banks (A)
Bits 20-18	1 of 8 Partitions
Bits 17-10	Column Address
Bits 9-2	Row Address
Bits 1-0	Four Word Block (FWB)

6.3.2.3. Initialize

The storage unit has logic to correct storage data via the ECC. This is to assure the requester that the MSU has valid ECC information for read cycles. The initialization process is initiated during powering up or via manual controls.

6.3.2.4. Fault Detection and Correction

The MSU detects all double and even multiples of failing storage bits within any read data transfer. Undefined ECC syndromes are detected to permit identification of a limited number of multiple odd-number errors. Single-bit data failures associated with any read data transfer or partial-write cycle are detected and corrected at the MSU.

6.3.2.5. Correction Logging

Correction logging is accomplished via software interrupts with the status word.

6.3.2.6. Status Word Reporting

Each MSU bank contains a status word register for each SIU/CPU port, which is accessible via the read data interface during a status reference request. All multiple-bit errors, along with the appropriate addresses, are reported, provided that the status word is not locked due to a previous error. Also, single-bit error syndromes and/or special codes, along with their appropriate addresses, may be selected for reporting by each SIU/CPU port. For the status word format, see Figure 6-5.

0	1	0	SSO	MUE	ECC Syndrome	Absolute Address
35	34	33	32	31	30 … 24	23 … 0

Figure 6-5. MSU Status Word Format

where:

Bits 35–33	The 2_8 code identifies the status word as an MSU status word.
Bit 32	Indicates that an SCISR Status Overflow (SSO) occurred. If the MSU detects an error condition following the one that caused the current status word to be generated, bit 32 is set to 1.
Bit 31	Indicates that a Multiple Uncorrected Error (MUE) has occurred at the referenced address. If bit 31 is clear, a single- bit error or a special code was detected by the MSU as indicated by the ECC syndrome bits 30–24.
Bits 30–24	Contains 7-bit ECC Syndrome bits generated by the detection/correction logic. The bits are decoded to indicate the erroring data bit on single-bit errors. A special code syndrome indicates that a write control or a write data parity error occurred at that reference location.
Bits 23–0	Indicates the Absolute Address of the storage location where the error condition was detected.

- If a port has a status word available from both banks, the status reference requests are required to retrieve them.
- The first status word generated for a port is the first output to a status reference request. If two status words for a port are generated simultaneously by two banks, internal circuitry arbitrarily assigns one of them as first.
- The address that is placed in the status word is the address that was presented to the MSU interface when an error was detected.
- Once a status word has been generated; i.e., locked, it is not overwritten by any further errors until it has been read.

- The SCISR status overflow bit is set if the status word is locked and any further errors occur in the same bank.
- The ECC syndrome (bits 30-24), when set to 1001111, indicates a special code. This code shows that the data was stored during a write or partial-write cycle in which a write-data fault occurred.
- The MUE bit is set only when the status word was generated for that failure.

6.3.2.7. Status Word Through Checking (External Storage Only)

Each MSU bank reports through check errors via the status word register.

- The ECC syndrome bits 30-24 when set to zeros (no error syndrome) with the MUE bit 31 set, indicates that a merge through check was detected on the transfer of read data to the requester.
- The ECC syndrome bits 30-24 when set to special code syndrome with the MUE bit 31 set, indicates that a merge through check error was detected during a partial write.
- If through check bits C7 or C8 or C7 and C8 are in error, the SBE status word reported contains all zeros in the syndrome bits 30-24 and both the MUE bit 31 and the SCISR status overflow bit 32 are not set.
- If a single bit error is detected in the lower half word and through check bit C8 is in error, or if a single bit error is detected in the upper half and through check bit C7 is in error, the SBE status word indicates the failing single bit error syndrome in bits 30-24 and the SCISR status overflow bit 32 is set.
- If a single bit error is detected in the lower half and through check bit C7 is in error, or if a single bit error is detected in the upper half word and through check bit C8 is in error, a MUE is generated. The status word reported indicates a MUE in bit 31 and the syndrome bits 30-24 indicates the failing single bit syndrome.

6.3.2.8. MSU Single-Bit Error and Special Code Reporting

Two modes of single-bit ECC error and special code reporting are available and can be set via the Maintenance OPeration (MOP) instruction for each SIU/CPU port.

- Unlimited Error Mode

  This mode generates an MSU status word each time the bank detects a single-bit error or special code on either a read or a partial write over the entire MSU address range assigned to an application.

- Limited Error Mode

  This mode generates an MSU status word any time a single bit error or special code is detected in a selected 8K address range. This 8K address range must be set by the Maintenance Operation instruction.

  - The MSU has one 8K address range register per bank that is shared by all ports.

- The status word can capture data for only the first error and indicates an SCISR status overflow for following errors if those errors occur before the first status word for that has been read.

6.3.2.9. MSU Fault Injection

The MSU provides a method of verifying the ECC mechanism. The MOP instruction provides a means of altering the normally generated ECC. This insertion takes place during the next full–word write cycle that falls within the specified 8K address range for that bank. When that address is subsequently referenced via a read or partial write–cycle, a bad–error syndrome will result. By varying the ECC injected by the MOP instruction, the programmer can verify the error correction and detection mechanism in the MSU. This function is self–clearing after execution to prevent corruption of more than one word.

The MOP function code and MOP data field are used to load a mask register, which is compared with the current address bits 23–13. The MSU MOP reference format is:

Not Used	MR	Not Used	MR	MR FC	Address Info
35 32	31 30	29 26	25 14	13 11	10 0

MSU Maintenance Operation Format (Internal Storage)

where:

Bits 35–32	Not Used
Bits 31–30	Maintenance and Reliability data bits 31 and 30 (on any 0_8, 1_8, 6_8, or 7_8 function code) selects the reporting of single–bit errors and special codes for the SIU/CPU channel as follows:

31	30	Function
0	0	Report single bit errors and special codes
0	1	Report single bit errors only
1	0	Report special codes only
1	1	Report single bit errors and special codes

The selected errors are reported as determined by the function code (0_8, 1_8, 6_8, or 7_8).

Bits 29–26	Not Used
Bits 25–14	Maintenance and Reliability data
Bit 25	MR data bit 25 (on 0_8, 1_8, 6_8, or 7_8 function codes) causes corruption of the INVALID ADDRESS PARITY signal on the next write or partial write cycle within the specified 8K address range to the applicable SIU or SIUs.

Bits 24-21 On any valid function code (0_8, 1_8, 6_8, or 7_8), the setting of MR data bits 22 and 21 corrupts read parity, upper and lower respectively, on the next read cycle from any SIU/CPU channel within the specified 8K address range. MR data bits 24 and 23 selects which of the four read data transfers is corrupted as follows:

24	23	Read data transfer
0	0	0
0	1	1
1	0	2
1	1	3

Bits 13-11 Maintenance and Reliability Function Code

13	12	11	Function Code	Definition
0	0	0	0_8	Disable Single-Bit Error (SBE) and special code error reporting for all bank addresses in the same application as the requesting SIU/CPU channel.
0	0	1	1_8	Report SBE or special code for this SIU/CPU channel only if address is within the special 8K range. Write data bits 10-0 specify the 8K range from which single-bit errors are to be reported. The 11 bits are functionally compared with bits 23-13 in the addresses presented to the MSU. This function code is cleared if function code 0_8 or 7_8 is transmitted for this SIU/CPU channel and write data bits 10 through 0 are used by the CPU in through address formation to request the desired MSU (See 9.15.5).
0	1	0	2_8	No operation*
0	1	1	3_8	No operation*
1	0	0	4_8	No operation*
1	0	1	5_8	No operation*
1	1	0	6_8	Invert ECC bits on next full word write cycle from either SIU/CPU channel as follows if address lies within the specified 8K range.

Write data bit	Inverted ECC bit
14	C0
15	C1
16	C2
17	C3
18	C4
19	C5
20	C6

1	1	1	7_8	Enable SBE special code reporting for all bank addresses in the same application as the requesting SIU/CPU channel. This function is cleared by function code 0_8 or 1_8 for the SIU/CPU channel.

* *Not available for MSU use. Interface errors are checked but no operation is performed.*

Bits 10-0 Address Information for F-codes 0_8 and 7_8, the address points to the MSU bank for which Single Bit Error (SBE) reporting is to be enabled/disabled.

For F-codes 1_8 and 6_8, bits 10-0 specify the 8K address range to be used.

Reporting of either single-bit errors or special codes may be disabled by the appropriate MOP instructions per bank.

Only the most recently loaded 8K address range from either SIU/CPU channel is valid for function codes; i.e., if the 8K range is changed after function code 6_8 is sent, but before the next write cycle, the new 8K address is valid for function code 6_8.

These functions are controlled via the SSP interface. The control affects all SIU/CPU channels or the 8K address range simultaneously.

6.3.2.10. Storage Lock

During a storage lock mode of operation, the priority in that bank is retained by that requester until, in its application, this mode is terminated. The MSU continues to honor requests from other requesters assigned to a different application.

6.3.2.11. System Synchronization

The MSU receives a system synchronization signal every CPU microinstruction cycle, which is used to synchronize the MSU with the requester.

6.3.3. Fault Handling

Faults detected at the MSU banks are handled as follows:

- Read Data Faults (Internal Storage)

  Single-bit failures associated with a read data transfer are detected and corrected by the MSU. A status word is generated from the earliest data transfer containing single bit errors as determined by the MOP function.

  All double and even multiples of failing storage bits along with a limited number of odd multiples of failing storage bits leading to undefined syndrome are detected as multiple data failures. If a multiple-bit error occurs within the corrected data transfer, it is reported along with the corrected data transfer as an MUE. An MUE will also occur if a single-bit error is detected on a word that had a special ECC stored earlier. A status word is generated from the earlier data transfer containing corrupted data.

A special code syndrome associated with a read data transfer is transmitted to the system by simultaneous reporting via both MUE and ADDRESS NOT AVAILABLE signals. A status word is generated as determined by the MOP function from the earlier data transfer containing the special code syndrome.

- Read Data Faults (External Storage)

  The MSU detects and corrects single-bit failures associated with a read data transfer. A status word is generated as determined by the MOP function and the availability of the status word register.

  All double-bit failure along with a limited number of multiple-bit failures are detected as multiple-bit errors and reported as an immediate check via the MUE line and a delayed check via the delay check line.

  A special code error is indicated to the requesting system during a read data transfer by the presence of an MUE signal and an ADDRESS NOT AVAILABLE signal. A status word is generated as determined by the MOP function and the availability of the status word register. If a single-bit error occurs after a special code has been forced into an address, an MUE is detected.

  If an error is detected in the read data being transferred to the requester and no reportable ECC errors are detected, an extended delayed check (one 116 nanosecond clock cycle later than the normal delayed check) is sent to the CPU/SIUs in the same cluster and application. The status word register is forced to indicate a MUE without an error syndrome.

- Write Data Faults (Internal storage)

  During a full-word write cycle - If an error is detected within the write data, the special ECC is forced into the correction bits and the CPU is informed of the error via the WRITE DATA ERROR signal. The write operation is completed even in this error condition.

  During a partial write cycle - The type of error detected in the read data determines how the write data is handled.

  - If an error is detected within the write data but no error is detected in the read data, the read and write data are combined and stored with the special ECC forced into the correction bits. The CPU is informed of the write data error via the WRITE DATA ERROR signal.

  - If there is no error in the write data but a single-bit error is detected within the read data; the read data is corrected, combined with the write data, and stored.

  - If there is an error in the write data and a single-bit error within the read data; the read data is corrected, combined with the write data, and stored with the special ECC forced into the correction bits. The CPU is informed of the write data error via the WRITE DATA ERROR signal.

  - If an MUE or special code is detected in the read data, the read and write data is not combined and the stored data is not changed, thus retaining the MUE or special code. In addition, if an error is detected within the write data, the CPU is informed via the WRITE DATA ERROR signal.

- Write Data Faults (External storage)

  During a full-word write cycle – If an error is detected within the write data, the special ECC is forced into the correction bits and the CPU is informed of the error via the WRITE DATA ERROR signal.

  During a partial write cycle – The type of error detected in the read data determines how the write data is handled.

  - If an error is detected within the write data but no error is detected in the read data, the read and write data are combined and stored with the special ECC forced into the correction bits. The CPU is informed of the write data error via the WRITE DATA ERROR signal.

  - If there is no error in the write data but a single-bit error is detected within the read data; the read data is corrected, combined with the write data, and stored.

  - If there is an error in the write data and a single-bit error within the read data; the read data is corrected, combined with the write data, and stored with the special ECC forced into the correction bits. The CPU is informed of the write data error via the WRITE DATA ERROR signal. A status word is generated as determined by the MOP function and the availability of the status word register.

  - If an MUE or special code is detected in the read data, the read and write data is not combined and the stored data is not changed, thus retaining the MUE or special code. Also, the MUE is reported via immediate and delayed check lines. In addition, if an error is detected within the write data, the CPU is informed via the WRITE DATA ERROR signal.

  - If an error is detected in the merged data, the read and write data are combined and stored with the special code forced into the ECC check bits. An extended delayed check (one clock cycle later is sent to the CPU/SIUs in the same cluster and application provided the read data did not contain a reportable error. The status word register is forced to indicate an MUE and special code syndrome. If the status word register has been loaded the SCISR status overflow bit is set.

- Address Faults

  Address faults are recognized by detection of address parity errors or addresses beyond the range of the MSU. Address faults are handled in different ways, depending upon who makes the request and the type of cycle to be performed:

  - SIU requests

    1. If an address error is detected during a read cycle, the CPU is informed of the error by either the ADDRESS PARITY ERROR signal or the ADDRESS NOT AVAILABLE signal.

    2. If an address error is detected during a write cycle, the write cycle is aborted and the CPU is informed of the error by either the ADDRESS PARITY ERROR signal or the ADDRESS NOT AVAILABLE signal. In addition, the ADDRESS NOT AVAILABLE signal is returned to each subsequent requester in that application in that bank until a PORT CLEAR signal is received from the SSP.

- CPU or IOU request

  If an address error is detected during either a read or a write cycle, the cycle is aborted and the CPU is informed of the error by either the ADDRESS PARITY ERROR signal or the ADDRESS NOT AVAILABLE signal.

■ Control Fault

If a control error is detected, the unit is forced into the partial write mode and the special ECC is forced into the check bits provided an MUE or special code was not detected with the read data. If an MUE or special code is detected with the read data, the write is aborted so that the original data remains unchanged. In either situation, a signal is sent back to the CPU to indicate a control error.

■ Refresh Fault

All CPUs are notified of a refresh fault via the ADDRESS NOT AVAILABLE signal line. This signal is continually sent to all CPUs accessing the bank in which the refresh fault occurred until the SSP initiates a CHECK RESET signal or the MSU bank power is dropped and then brought up again. A scannable flip-flop is set for SSP reference. A MASTER CLEAR signal will not clear a refresh fault.

6.3.4. Fixed Address Assignments

The interrupt subroutine entrances and certain status words are assigned fixed locations in main storage as shown in Table 6-1. The listed addresses are relative to the contents of the 7-bit Module Select Register (MSR).

Table 6-1. 1100/70 Fixed Address Assignments

Octal	Decimal	Assignment
200	128	Reserved for Hardware Default
201	129	Unassigned
202	130	Unassigned
203	131	Unassigned
204	132	I/O Normal Status Interrupt
205	133	I/O Tabled Status Interrupt
206	134	I/O Machine Check Interrupt
207	135	Unassigned
210	136	Quantum Timer Interrupt
211	137	Real-Time Clock Interrupt
212	138	Unassigned
213	139	Unassigned
214	140	Unassigned
215	141	Unassigned
216	142	Dayclock Value
217	143	Dayclock Interrupt
220	144	Immediate Storage Check Interrupt
221	145	Invalid Instruction
222	146	Executive Request Interrupt
223	147	Guard Mode Interrupt
224	148	Test and Set Interrupt

Table 6-1. 1100/70 Fixed Address Assignments (continued)

Octal	Decimal	Assignment
225	149	Characteristic Underflow Interrupt
226	150	Characteristic Overflow Interrupt
227	151	Divide Check Interrupt
230	152	Addressing Exception Interrupt
231	153	Breakpoint Interrupt
232	154	Interprocessor Interrupt
233	155	SSP Interrupt
234	156	Delayed Interrupt
235	157	Jump History Stack Interrupt
236	158	Reserved
237	159	Unassigned
240	160	CPU 0 Channel Address Word 0
241	161	CPU 0 Channel Address Word 1
242	162	Reserved
243	163	Reserved
244	164	CPU 1 Channel Address Word 0
245	165	CPU 1 Channel Address Word 1
246	166	Reserved
247	167	Reserved
250	168	CPU 2 Channel Address Word 0
251	169	CPU 2 Channel Address Word 1
252	170	Reserved
253	171	Reserved
254	172	CPU 3 Channel Address Word 0
255	173	CPU 3 Channel Address Word 1
256	174	Reserved
257	175	Reserved
260	176	IOU 0 Machine Check Status Word
261	177	IOU 0 Channel Status Word 0
262	178	IOU 0 Channel Status Word 1
263	179	IOU 0 Channel Status Word 2
264	180	IOU 1 Machine Check Status Word
265	181	IOU 1 Channel Status Word 0
266	182	IOU 1 Channel Status Word 1
267	183	IOU 1 Channel Status Word 2
270	184	IOU 2 Channel Check Status Word
271	185	IOU 2 Channel Status Word 0
272	186	IOU 2 Channel Status Word 1
273	187	IOU 2 Channel Status Word 2
274	188	IOU 3 Channel Check Status Word
275	189	IOU 3 Channel Status Word 0
276	190	IOU 3 Channel Status Word 1
277	191	IOU 3 Channel Status Word 2

NOTE:

All fixed addresses are relative to the MSR.

6.4. General Register Stack

The General Register Stack (GRS) consists of 128 36-bit high-speed, general-purpose registers that can be independently referenced in parallel with storage. The general registers are addressed either explicitly or implicitly by the instruction words and are categorized into five groups: index registers, arithmetic registers, special registers, CPU state control registers, and unassigned registers. Tables 6-2 and 6-3 summarize the control register address assignments.

Table 6-2. GRS Register Assignments 0 Through 63

Octal	Decimal	Register Assignment
0000	0	Unassigned
0001	1	User X1
|	|	
0011	9	User X9
0012	10	User X10
0013	11	User X11
0014	12	User X12/A0
0015	13	User X13/A1
0016	14	User X14/A2
0017	15	User X15/A3
0020	16	User A4
0021	17	User A5
|	|	
0033	27	User A15
0034	28	Unassigned
|	|	
0037	31	Unassigned
0040	32	EXEC BDT Pointer
0041	33	Immediate Storage Check Program Return Address
0042	34	Immediate Storage Check Designator Register
0043	35	Normal Program Return Address
0044	36	Normal Designator Register
0045	37	User BDT Pointer
0046	38	E|0|0--0| BDI0 |E|2|0--0| BDI2
0047	39	E|1|0--0| BDI1 |E|3|0--0| BDI3
0050	40	Quantum Timer
0051	41	Guard Mode Program Return Address
0052	42	Guard Mode Designator Register
0053	43	Guard Mode Interrupt Status
0054	44	Immediate Storage Check Status
0055	45	Normal Status
0056	46	Guard Mode Current Instruction
0057	47	Guard Mode Current Register Set Pointer
0060	48	Unassigned
|	|	*
0067	55	Unassigned
0070	56	Jump History Stack
|	|	
0077	63	Jump History Stack

**Locations 60_8 through 67_8 are used as temporary working storage locations by the CPU, and their contents are therefore unpredictable.*

NOTE: Delay Storage Checks are classed as normal interrupts.

Table 6-3. GRS Register Assignments 64 Through 127

Octal	Decimal	Register Assignment
0100	64	Real-Time Clock
0101	65	User R1/Repeat Count
0102	66	User R2/Mask Register
0103	67	User R3
0104	68	User R4
0105	69	User R5
0106	70	User R6
0107	71	User R7
0110	72	User R8
0111	73	User R9
0112	74	User R10
|	|	
0117	79	User R15
0120	80	Executive R0
0121	81	Executive R1/Repeat Count
0122	82	Executive R2/Mask Register
0123	83	Executive R3
|	|	
0137	95	Executive R15
0140	96	Unassigned
0141	97	Executive X1
0142	98	Executive X2
0143	99	Executive X3
|	|	
0153	107	Executive X11
0154	108	Executive X12/A0
|	|	
0157	111	Executive X15/A3
0160	112	Executive A4
|	|	
0173	123	Executive A15
0174	124	Unassigned
|	|	
0177	127	Unassigned

6.4.1. Index (X) Registers

The index registers, referred to as X-registers, provide the programmer with address modification capability (indexing).

An index register contains a modifier field (X_m), which is used to modify the operand address (indexing), and an increment field (X_i), which is used to modify the modifier field (automatic incrementation). If the Relocation and Storage Suppression designator (D7) and the i-field of an instruction are one, 24-bit index register mode is specified. In this mode, X_m is the lower 24 bits of the index register (bits 23-0), and X_i is the upper 12 bits of the index register (bits 35-24). In all other cases, 18-bit index register mode is selected. In this mode, X_m is the lower 18 bits of the index register (bits 17-0), and X_i is the upper 18 bits of the index register (bits 35-18).

Index register zero (X0) has indexing capability when used by the Block Transfer (BT) instruction. It does not have indexing capability when addressed by the x-field of an instruction. It may also be used as a general-purpose register.

6.4.2. Arithmetic (A) Registers

The arithmetic registers, referred to as A-registers, provide intermediate storage for arithmetic operands and results. To the programmer, the A-registers effectively function as accumulators. Any two or three consecutively addressed accumulators can hold double- or triple-length operands.

Four of the A-registers are also assigned as index registers. Their dual role provides the CPU with additional flexibility because it permits the result of a given operation to be readily used for address modification in a following instruction.

6.4.3. Special (R) Registers

The special registers are referred to as R-registers. Three of the R-registers serve special purposes and are the Real-Time Clock (RTC) register, repeat count register, and masked register. The remaining R-registers are not specifically assigned and may be used as temporary storage locations.

6.4.3.1. Real-Time Clock (R0)

The contents of the lower half (bit positions 17-0) of the RTC register are decreased by one every 200 microseconds, independent of program control or supervision. A Real-Time Clock interrupt occurs if the RTC value in the lower half of the RTC register is zero when a decrementation cycle is initiated. The upper half (bit positions 35-18) of the RTC register should not be used.

6.4.3.2. Repeat Count Register (R1)

The contents of the repeat count register define the maximum number of times a repeated instruction is executed. During execution of a repeated instruction, the contents of the lower half of the repeat count register are decreased by one each time the repeated instruction is executed. If an interrupt occurs during the sequence of repeated executions of an instruction, the repeat sequence is suspended to process the interrupt, and the current count is left in R1. The repeated sequence may be resumed after the interrupt has been processed. The final value of the count after the repeat sequence terminates is always available in R1. If the contents of the repeat count register is zero, the repeated instruction is not executed and the execution of the next instruction is initiated. Zero is defined as all zeros or all ones in the lower half of the word (bit positions 17-0); the upper half (bit positions 35-18) of the repeat count register should not be used.

6.4.3.3. Masked Register (R2)

The bits in the masked register specify the fields of operands to be operated upon in certain instructions. A logical $\boxed{\text{AND}}$ is performed with the operand and the masked register and/or its complement. The portions of the operand so selected are then used in the instruction operation.

6.4.4. CPU State Control Registers

The information assigned to locations 040 through 077 is normally associated with interrupt sequences. Certain information, such as the CPU state, is stored during the interrupt sequence for all interrupts. Other information, such as a specific interrupt status, is stored during the interrupt sequence only for that type of interrupt. The information in the jump history stack is stored during the course of program execution but is not typically used until an interrupt occurs. Addressing status is stored in GRS locations 046 and 047 during the Load Addressing Environment (LAE), the Load Bank and Jump (LBJ) instructions, and is captured by software during the interrupt routine.

6.4.5. Unassigned Registers

The unassigned registers are not assigned to a specific use and may be used as temporary storage locations. Some of the unassigned registers may be used in the performance of those instructions that operate with a double-word or triple-word operand (A15+1 and A15+2).

6.4.6. Control Register Selection Designator

Among the 128 addressable general registers are a set of registers referred to as A–, X–, and R–registers that are designated for use by the user, and another set of A–, X–, and R–registers that are designated for use by the Executive. The Control Register Selection designator (D6) defines which set of A–, X–, and R–registers is addressed by the a– and x–fields of an instruction. When D6 is 0, the a– and x–fields of an instruction reference the set of A–, X–, and R–registers designated for use by the user. When D6 is 1, the a– and x–fields of an instruction reference the set of A–, X–, and R–registers designated for use by the Executive. (This distinction does not hold when the operand address (U) addresses general registers, since they are directly addressed as storage locations 0000 through 0177 in this case.)

6.4.7. Special GRS Notes

The following GRS locations cannot be directly altered when the CPU is in Guard Mode (D2 is 1): Addresses 040_8 through 0100_8 inclusive, and 0120_8 through 0177_8 inclusive. Note that certain privileged GRS locations (for example, 046 and 047, the Bank Descriptor Indices) can be indirectly altered as the result of the execution of special instructions (for example, LBJ).

7. Executive Control

7.1. General

The Central Processing Unit (CPU) operates under the control of an Executive program which controls and coordinates the activities of the combined hardware and software systems and has exclusive use of certain control capabilities. By the use of bank descriptors, it can relocate any program in main storage. It provides storage protection through the use of storage limits registers. The bank descriptor specifications are contained in the General Register Stack (GRS). The bank descriptors are controlled by the Executive program for itself and for user programs. However, the user programs, through the Load Bank and Jump (LBJ), Load I-Bank Base and Jump (LIJ), and Load D-Bank Base and Jump (LDJ) instructions, are allowed to modify part of the designator register from a table prepared by the Executive program. The user programs are also allowed to modify several of the control bits by using the Load DR Designators (LPD) instruction (See 9.13.1). The operations related to and affected by the contents of the bank descriptors are explained in this section.

7.2. CPU State

The CPU state is defined as information contained in the CPU and required to describe a program activity. This includes the bank descriptor information, the designator register, the relative program address, and the GRS. The processor state is automatically saved in GRS when an interrupt occurs. The designator register and relative program address can also be stored by instruction. Each element of the program state can be loaded by instruction, and certain elements may be loaded in groups to facilitate the orderly sequencing of program control.

7.2.1. Designator Register

The designator register contains information controlling functional characteristics of the CPU. The designator register may be loaded by instruction, although certain bit combinations are not valid or may not be available to the user. In general, no hardware checks are made for these invalid combinations. When an interrupt occurs, the current value of the designator register is stored in GRS, and the register is cleared, unless otherwise specified in the following paragraphs, to establish the proper interrupt handling environment.

The format of the designator register is:

Reserved	D29	R	D27	D26 D19	R	D17 D12	R	D10	R	D8 D0
35 30	29	28	27	26 19	18	17 12	11	10	9	8 0

where:

D35–D30	Reserved When the designator register is stored in GRS for an interrupt or in main storage for a Load Designator Register instruction, these bits are zero filled.
D29	Quantum Timer Enable Designator When this designator is 1, the quantum timer value is decreased by 1 for every 116 nanoseconds period that the CPU is actually executing instructions. When the quantum timer value is 0, a Quantum Timer interrupt is generated. When D29 is 0, the quantum timer value is not altered.
D28	Reserved When the designator register is stored in GRS for an interrupt or in main storage for a Load Designator Register instruction, this bit is zero filled.
D27	Fault Interrupt Pointer Designator When set, this designator indicates that a fault interrupt will be directed to the System Support Processor (SSP), instead of to the operation software, if an unretriable fault condition is detected or a storage interface retry fails.
D26–D24	Software Performance Monitors These bits are used along with D2 to specify eight software states that can be monitored by the performance monitoring feature.
D23	Divide Check Designator This designator is set to 1 when the characteristic of a floating-point result is less than -200_8 in a single-precision instruction, or if D20 or D5 is clear and the characteristic is less than -2000_8 in a double-precision instruction. If the characteristic is less than -2000_8 in a double-precision instruction, and D20 and D5 are both set, D21 is unchanged.
D22	Characteristic Overflow Designator This designator is set to 1 when the characteristic of a floating-point result is greater than 177_8 (single-precision) or 1777_8 (double-precision).
D21	Characteristic Underflow Designator This designator is set to 1 when the characteristic of a floating-point result is less than -200_8 (single-precision) or -2000_8 (double-precision).

D20 Arithmetic Exception Interrupt Designator

When this designator is 0, if an arithmetic exception occurs (D23, D22, or D21 set to 1), the specified A-registers are cleared to 0 and no interrupt occurs. When D20 is 1, if an arithmetic exception occurs, the specified A-registers are left unchanged (except as specified by D5), and an interrupt occurs. When D20 is 1, all instructions that can cause an arithmetic exception are executed without instruction overlap (i.e., completely executed prior to beginning the execution of the next instruction).

D19 EXEC Bank Descriptor Table Pointer Enable Designator

When this designator is 0, only the user Bank Descriptor Table (BDT) pointer may be selected during execution of the LBJ, LIJ, or LDJ instructions. If an attempt is made to reference the EXEC bank descriptor table, an Addressing Exception interrupt is generated. When D19 is 1, either the EXEC or user BDT pointer may be selected during execution of an LBJ, LIJ, or LDJ instruction.

D18 Reserved

D17 Floating-Point Residue Store Enable Designator

When this designator is 1, it enables the residue store for single-precision floating-point instruction.

D16 Bank Descriptor Register (BDR3) Write Protection Designator

When this designator is 1 and either D7 or the i-bit is 0, a Guard Mode interrupt will occur if an attempt is made to write into the storage area specified by BDR3.

D15 BDR2 Write Protection Designator

When this designator is 1 and either D7 or the i-bit is 0, a Guard Mode interrupt will occur if an attempt is made to write into the storage area specified by BDR2.

D14 BDR1 Write Protection Designator

When this designator is 1 and either D7 or the i-bit is 0, a Guard Mode interrupt will occur if an attempt is made to write into the storage area specified by BDR1.

D13 BDR0 Write Protection Designator

When this designator is 1 and either D7 or the i-bit is 0, a Guard Mode interrupt will occur if an attempt is made to write into the storage area specified by BDR0.

D12 BDR Selector Designator

When this designator is 1, BDR1 and BDR3 are selected as the primary pair of bank descriptor registers; when D12 is 0, BDR0 and BDR2 are selected as the primary pair. The primary pair is selected over the secondary pair if the storage limits overlap. D12 is toggled during a jump that is taken (excluding User Return) if the jump to address falls exclusively within the limits of the secondary pair. D12 is not altered when an interrupt occurs.

D11 Reserved

D10 Quarter-Word Mode Designator

When this designator is 0, for instructions with function codes less than 70_8 (not including 07), the j-field values of 4, 5, 6, and 7 are interpreted as follows:

j = 4	Specifies half-word (18-bit) transfers to or from bits 35 through 18 of the operand.
j = 5	Specifies third-word (12-bit) transfers to or from bits 11 through 0 of the operand.
j = 6	Specifies third-word (12-bit) transfers to or from bits 23 through 12 of the operand.
j = 7	Specifies third-word (12-bit) transfers to or from bits 35 through 24 of the operand.

When D10 is 1, for instructions with function codes less than 70_8 (not including 07), the j-field values of 4, 5, 6, and 7 are interpreted as follows:

j = 4	Specifies quarter-word (9-bit) transfers to or from bits 26 through 18 of the operand.
j = 5	Specifies quarter-word (9-bit) transfers to or from bits 8 through 0 of the operand.
j = 6	Specifies quarter-word (9-bit) transfers to or from bits 17 through 9 of the operand.
j = 7	Specifies quarter-word (9-bit) transfers to or from bits 35 through 27 of the operand.

The value of D10 has no effect on an instruction in the following circumstances:

- When the f-field of the instruction contains a value in the range 70_8 through 77_8, or 07_8.
- When the j-field contains a value other than 4, 5, 6, or 7.

D9 Reserved

D8 Floating-Point Zero Format Selection Designator

This designator affects Floating Add, Floating Add Negative, Floating Multiply, Floating Divide, and Load and Convert to Floating (only single-precision) instructions. When D8 is 0, and if the mantissa of the most significant word of a single-precision floating-point result is ± 0, the entire word is stored as all zero. When D8 is 1, and if the mantissa of the most significant word of a single-precision floating-point result is ± 0, the most significant word is packed and stored with the appropriate characteristic.

D7 Relocation and Storage Suppression Designator

D7 controls index register length, relocatability, and limit violation checking. If D7 is 0, index registers are 18 bits long, relocation is performed through basing, and a limits violation will cause a Guard Mode interrupt. If D7 is 1, the same functions are dependent upon the i-bit of the instruction currently executing: If i=0, operation proceeds as if D7=0; if i=1, index registers are 24 bits long, relocation is not performed (a base value is not added to the relative address), and relative addresses (which are not identical to absolute addresses) are not checked for limit violations.

Program addresses following a jump instruction are formed under the same D7 and i-bit conditions that were in effect for the jump instruction. That is, if an absolute jump occurred (D7=i=1), subsequent instruction references will be absolute; if a relative jump occurred (D7 or i=0), subsequent instruction references will be relocated according to the current addressing control designators and BDR values. D7 is set to 1 on master clear or when an interrupt occurs.

D6 General Register Stack (GRS) Selection Designator

When D6 is 0, the GRS addresses below are assigned for use by the user program and can be referenced by the a- and x-fields of the instruction.

Index (X) Registers	0001_8 - 0017_8
Accumulators (A-Registers)	0014_8 - 0033_8
Special (R) Registers	0101_8 - 0117_8

When D6 is 1, the GRS addresses below are assigned for use by the Executive program and can be referenced by the a- and x-fields of the instruction. D6 is set to 1 when an interrupt occurs.

Index (X) Registers	0141_8 - 0157_8
Accumulators (A-Registers)	0154_8 - 0173_8
Special (R) Registers	0120_8 - 0137_8

D5 Double-Precision Underflow Designator

When D5 is 0, a Floating-Point Characteristic Underflow interrupt occurs if characteristic underflow is detected during the execution of a double-precision floating-point instruction. The contents of the specified A-registers remain unchanged. When D5 is 1, the interrupt does not occur; however, the contents of the specified A-registers are cleared to zeros and the normal instruction sequence is continued.

D4 Not Used

D3 Allow Interrupts Designator

When D3 is 1, external interrupts are allowed; when D3 is 0, external interrupts are locked out. D3 may be altered by User Return (UR) and Load Designator (LD) instructions, is set by the Allow All Interrupts and Jump (AAIJ) instruction, and is cleared by the Prevent All Interrupts and Jump (PAIJ) instruction, and by the interrupt sequence.

D2 Privileged Instruction and GRS Protection Designator

When D2 is 1, a Guard Mode interrupt will occur if an attempt is made to execute a privileged (Executive) instruction, or to store into an Executive GRS location.

D2 equal to 1 also enables checking the length of the period during which interrupts are locked out by D3 equal to 0 (whether D3 became 0 by instruction execution or by taking an interrupt) or by a string of Execute or Indirect addressing. A total of 256 storage references (only those made by the CPU are counted) are allowed during this locked-out period. Additional references will cause a Guard Mode interrupt. This checking occurs whether or not an interrupt is actually being locked out. D2 equal to 1 enables a Guard Mode interrupt if 256 storage references are made during the Edit Decimal instruction.

NOTE:

Designator bits D16 through D13 are independent of D2.

D1-D0 Overflow Designator (D1) and Carry Designator (D0)

These designators are similar and defined together.

For the following instructions, D0 and D1 are cleared and then set according to the results of the operation:

A	(14)	Add to A
AN	(15)	Add Negative to A
AM	(16)	Add Magnitude to A
ANM	(17)	Add Negative Magnitude to A
AU	(20)	Add Upper
ANU	(21)	Add Negative Upper
AX	(24)	Add to X
ANX	(25)	Add Negative to X
DA	(71, 10)	Double-Precision Fixed-Point Add
DAN	(71, 11)	Double-Precision Fixed-Point Add Negative

D1 is set to 1 if an overflow condition is detected during execution of any of the above instructions, and D0 is set to 1 if a carry condition is detected. When D0 or D1 are set, they remain so until another one of the above instructions is executed, or until the designator bits are directly altered by the program.

Figure 7-1 shows three basic conditions of the Designator Register.

7.2.2. Dayclock

Each CPU contains an internal dayclock register that is updated every 200 microseconds. Master clear does not clear this register. It is cleared only upon power-up or programmed reset to zero. Master clear prevents any main storage dayclock updates until the CPU is running and has been defined as having control of the dayclock. Once the CPU has been defined to have control of the dayclock, it will continue to update the dayclock until it is master cleared, even though it may be in a stop condition. The dayclock value is a single 36-bit binary number without separately identified digits of time.

The dayclock register value is stored in the dayclock location in fixed storage (Module Select Register MSR + 216) if the dayclock of the CPU is enabled by the System Support Processor (SSP) software. The dayclock location is not altered by a CPU's dayclock if the SSP has not enabled that CPU's dayclock. On a multiprocessor system only one CPU's dayclock is enabled by the SSP.

User Mode

0 ---- 0	1	0	R	D25	D24	D23	D22	D21	D20	D19	0	D17	D16	D13	D12	0	D10	0	D8	0	0	D5	0	1	1	D1	D0
35 30	29	28	27 26	25	24	23	22	21	20	19	18	17	16	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Interrupt Mode

0 ---- 0	0 ---- 0	0 ---- 0	D12	0	0	0	0	1	1	0 ---- 0
35 29	28 24	23 13	12	11	10	9	8	7	6	5 0

Executive Mode

0 ---- 0	D29	0	R	D25	D20	1	0	D17	D16	D15	D14	D13	D12	0	D10	0	D8	D7	D6	D5	0	D3	0	D1	D0
35 30	29	28	27 26	25	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Figure 7-1. Basic Designator Register States

The value in the dayclock location (MSR + 216) is kept current, regardless of Input/Output (I/O) waits or execution of extended instructions; e.g., block transfer, etc.

The internal dayclock register value is replaced with the value in the dayclock location in fixed storage upon the execution of an LDC instruction regardless of whether the dayclock enable is set or not.

A Dayclock interrupt request will be set every 6.5536 seconds if the dayclock is both selected and enabled. The interrupt may be honored by any CPU in the application and will occur only if interrupts are allowed.

7.3. Address Formation

The primary levels of addressing within the CPU are relative and absolute. One basic mode of relative addressing exists within the CPU, allowing a number of variations that depend on the bit settings of the designator.

Relative address (U) contains the u-field of the instruction and the modifier field of the X_x-register. Relative addresses are formed by an 18-bit ones complement addition with end-around carry. A negative zero (all ones) cannot be generated as a relative address. For shift instructions, I/O instructions, or immediate operands (j = 16 or 17), the relative address is generally used directly as an operand. If a relative address is less than 0200, it is generally used to reference GRS. If the relative address is greater than 0200, it is converted to an absolute address and used to reference storage.

An absolute address is normally composed of the sum of a relative address and a base value selected from one of the four available bank descriptor registers. The absolute address is formed by performing a twos complement addition of the base value and the upper 18 bits of the 24-bit relative address.

7.3.1. Bank Descriptor

A Bank Descriptor (BD) contains the information necessary to describe a program segment for the purpose of storage allocation. Each Bank Descriptor is contained in a Bank Descriptor Table (BDT), and is located by adding a Bank Descriptor Index (BDI) to a Bank Descriptor Table Pointer. The 1100/70 Descriptors are two words in length; therefore, the BDT length and BDI are in units of Descriptors, not words. The maximum BDT length is then 8K words or 4K Descriptors. The BDT pointer and the Bank Descriptor are shown in Table 2-1 and Table 2-2. The Bank Descriptor fields are described in the following paragraphs.

Bank Descriptor Table Pointer Format

Table Length	Table Address
35 24	23 0

where:

Bits 35–24	Table Length specifies the bank descriptor table length (number of descriptors).
Bits 23–0	Table Address specifies the bank descriptor table address (absolute).

Bank Descriptor Format

Reserved for Software		Base Value						
Upper Limit	Lower Limit	R	W	P	V	*	C	Use Count
35 24	23 18 17 15	14	13	12	11	10	9	8 0

where:

First word

Bits 35–18	Reserved for software
Bits 17–0	Base value for relocation

Second Word

Bits 35–24	Storage protection upper limit value
Bits 23–15	Storage protection lower limit value
Bit 14	Residency flag
Bit 13	Write protection flag
Bit 12	Privileged protection flag
Bit 11	Validate entry point flag
Bit 10	Reserved for software
Bit 9	Use count interrupt on zero flag
Bits 8–0	Use count value

7.3.2. Limits

The upper and lower limits define the range of relative addresses that may be used to reference a program segment. The lower limit allows program segments to begin on 512-word boundaries within relative space, and the upper limit allows program segments to end on 64-word boundaries within both relative and absolute space; hence, the program segment size may be any multiple of 64 words. The limits check is an inclusive comparison; that is, a relative address is within a set of limits if n is greater than or equal to the lower limit and less than or equal to the upper limit ($U_{17-9} \geq LL$ $\boxed{\text{AND}}$ $U_{17-6} \leq UL$).

7.3.3. Control Information

The flag and use count fields of the BD provide control pertaining to the relative space (segment) defined by the BD and a count of the activity or usage of this BD.

The R-flag indicates that an Addressing Exception interrupt is to occur if a reference is made to the segment through the new bank descriptor of an LIJ, LDJ, or LBJ instruction.

The W-flag indicates that a Guard Mode interrupt, indicating write protection violation, is to occur if a write is attempted into the segment.

The P-flag value is transferred to designator register bit 2 (D2 - privileged instruction and GRS protection designator) during the execution of an LBJ, LIJ, or LDJ instruction.

The V-flag indicates that entry point validation must be performed. This is accomplished by assuring that the relative operand address of the LBJ, LIJ, or LDJ instruction (jump address) that references the bank descriptor is equal to the bank descriptor lower limit value extended with low-order zeros (bits 8 through 0). This relative operand address must also select the BDR being loaded. If these conditions are not met and V is one, an Addressing Exception interrupt will occur. If the relative operand address is not within any limits, a Guard Mode interrupt will occur.

The C-flag indicates that an interrupt will occur if the use count is decreased to zero.

7.3.4. Bank Descriptor Registers

A BDR contains the upper limit, lower limit, and base of a bank descriptor; these allow relative address limits checking for protection, base selection, and absolute address formation. Four bank descriptor registers, BDR0, BDR1, BDR2, and BDR3, are provided in the CPU. The BDRs are loaded by the LAE, LL, or LB instructions. These instructions do not test the flags or change the use count.

7.3.5. Address Generation

If base suppress conditions exist, the relative address is used as the absolute address, otherwise an absolute address is generated. To generate an absolute address, a relative address is added to a base value selected from those available in the four bank descriptor registers. To select which of the base values to use, a limits check is made between the relative address and the upper and lower limits. Designator D12 determines the order of BDR use as follows:

Selector	Use (in order of preference)
D12 = 0	BDR0, BDR2, BDR1, BDR3
D12 = 1	BDR1, BDR3, BDR0, BDR2

The lower limit (9 bits) of a BDR is checked against U_{17-9}, and the upper limit (12 bits) is checked against U_{17-6}. The first BDR passing the limits check is used for absolute address generation. Figure 7-3 shows the base value selection in flow chart form. If a relative address is not within the limits of any BDR, a storage limits violation occurs. If a relative address is within limits, the base corresponding to those limits is used to convert the relative address to an absolute address with which to reference storage. Base values may be assigned in 64-word increments.

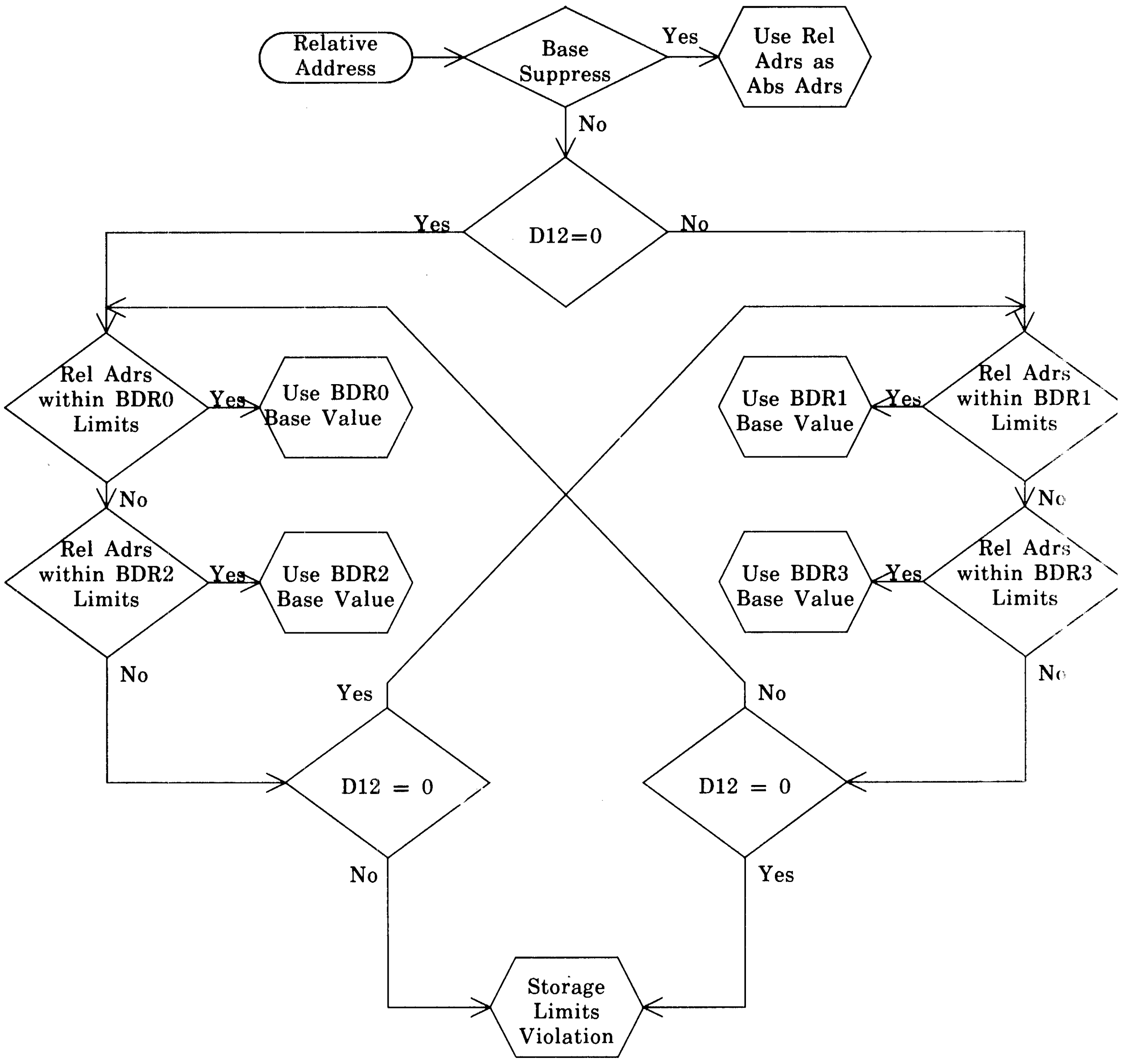
The base addition is done with base and relative address alignments shown below:

```
Relative address     0 ---- 0    17 ---- 6   5 ---- 0

Base value           17 --- 12   11 ---- 0   0 ---- 0

Absolute address     23 -------------------------- 0
```

The base addition is end off; i.e., a carry produced out of bit 23 is not propagated into bit 0.



Define "Base Suppress" as:

D7=1 and i=1 and not P Fetch or A Flag and P Fetch

Figure 7-3. Base Value Selection

8. Interrupts

8.1. General

An interrupt causes the current instruction sequence to be suspended and an instruction sequence starting at a fixed storage location to be initiated; the fixed address replaces the value in the program address register. The fixed storage address is associated with the event or condition that caused the interrupt to be generated, and thereby allows switching to a program to respond to that condition or event. Excepting those instructions that are explicitly named as interruptable, such as repeated instructions like Block Transfer, the Central Processing Unit (CPU) honors interrupts only after the current instruction is completed and only if the interrupt to be honored is allowed. See Table 8-1 for interrupt priorities.

The following interrupts are always allowed unless the CPU is stopped:

- All program exception interrupts, including Guard Mode and Addressing Exception interrupts.
- All arithmetic exception interrupts, including Characteristic Overflow, Characteristic Underflow, and Divide Check interrupts.
- Certain program initiated interrupts, including Executive Request, Test and Set, Quantum Timer, and Breakpoint interrupts.
- Immediate CPU internal or storage check interrupts caused as the result of transfers between the CPU and Storage Interface Unit (SIU) or between the CPU and main storage.

Certain interrupts are disallowed between the execution of a Prevent All Interrupts and Jump (PAIJ) instruction or the occurrence of an interrupt, and the execution of an Allow All Interrupts and Jump (AAIJ) instruction or User Return (UR) or Load Designator Register (LD) instruction that sets Allow Interrupts designator (D3). These include the following:

- All I/O interrupts, including those for Normal status, Table status, and Machine Check interrupts.
- Jump History Stack interrupts.
- Dayclock and Real-Time Clock interrupts.
- System Support Processor (SSP) interrupts.

- InterProcessor Interrupts (IPIs).
- CPU, MSU, and SIU Delayed Check interrupts resulting from interface faults or internal faults in one of the system components.

NOTE: All interrupts are disallowed if the CPU is stopped.

Table 8-1. Interrupt Priority

Priority Level	Interrupt Type
0	Immediate Storage Check
1	Guard Mode
2	Addressing Exception
3	Invalid Instruction
4	Executive Request
5	Test and Set
6	Characteristic Overflow
7	Characteristic Underflow
8	Divide Check
9	Breakpoint
10	Quantum Timer
11	Jump History Stack
12	Real-Time Clock
13	Dayclock
14	Delayed Check Interrupts
	CPU
	MSU
	SIU
15	SSP Interrupt
16	IOU 0 Machine Check
17	IOU 1 Machine Check
18	IOU 0 Normal Status
19	IOU 1 Normal Status
20	IOU 0 Table Status
21	IOU 1 Table Status
22	IPI from CPU 0
23	IPI from CPU 1
24	IPI from CPU 2
25	IPI from CPU 3

NOTES:

1. *Priority levels 0 through 12, 15, 22, through 25 are internal interrupts.*
2. Priority levels 13, 14, and 16 through 21 are external interrupts, which are handled only by the one CPU presently having the External Interrupt Sync Logic enabled.
3. Priority levels 11 through 25 are deferrable interrupts, which can be locked out.
4. Priority levels 0 through 10 cannot be locked out.

8.2. Interrupt Sequence

When the CPU honors an interrupt request, the following events occur:

- The CPU state is stored in the General Register Stack (GRS) in three groupings: program status (041–044, 050–052), addressing status (040, 045–047), and interrupt status (053–055).
- All designator register bits are set to 0 except General Register Selection designator (D6) and Relocation and Storage Suppression designator (D7), which are set to 1, and Bank Descriptor Register Selector designator (D12), which is not altered.
- External interrupts are prevented from occurring until allowed by an AAIJ, UR, or LD instruction.
- Control is transferred to the associated interrupt location. Note that this instruction must be an unconditional jump but need not be a Load Modifier and Jump (LMJ). An LMJ instruction may capture the wrong relative address because of the CPU state change (e.g., LBJ interrupt).

8.2.1. Program Status

Program status is stored for all interrupts and includes the following information:

■ Program Return Address	–	GRS location 043 for Normal interrupts
		GRS location 051 for Guard Mode interrupts
		GRS location 041 for Immediate Storage Check interrupts
■ Quantum Timer Value	–	GRS location 050 for all interrupts
■ Designator Register Value	–	GRS location 044 for Normal interrupts
		GRS location 052 for Guard Mode interrupts
		GRS location 042 for Immediate Storage Check interrupts

Bits 35–30 and 28–24 are always zero filled when the designator register is stored in the GRS for an interrupt.

A program return address is the address of the instruction following the last instruction that was fully executed; program control would normally be returned at this point for recoverable errors. The program return address stored in GRS location 041 or 043 is in the following format:

A	Not Used	Program Return Address
35	34 24	23 0

The program return address value will vary, depending on the operation being performed at the time of interrupt:

- If an incomplete BT, or search instruction is interrupted, the return address will be P.
- If a satisfied skip instruction is interrupted, the return address will be P+2.
- If a satisfied jump instruction is interrupted, the return address will be U.
- If an instruction other than the above is interrupted, the return address will be P+1.

The contents of the program address register is changed only by a jump instruction (including User Return) or interrupt. Instruction references following a jump instruction are made under the same addressing constraints that conditioned the operand address of the jump instruction that began the straight line instruction stream.

Bit position 35 of the first word of the 2-word program status packet contains a flag that identifies the correct program addressing mode. The two modes of program address generation include absolute (A=1), corresponding to D7=i=1, and relative (A=0), corresponding to D7=0 or i=0.

Bit positions 18 through 23 of the relative program address are 0 unless absolute 24-bit indexing mode is selected. For straight line instruction sequencing, the relative program address is increased by 1 for each instruction that is executed or skipped. This increase is accomplished by twos complement addition with wraparound at 18 or 24 bits, depending on the value of the A-flag.

8.2.2. Addressing Status

Addressing status is not actually stored during the interrupt sequence. The information within this group is placed in the GRS by the software, either directly (load, store) or indirectly (Load Bank and Jump, LBJ; Load Addressing Environment, LAE), and is used from the GRS by the CPU for addressing operations. Addressing status includes the following information:

- Executive bank descriptor table pointer.
- User bank descriptor table pointer.
- Bank descriptor specifications, in the following format:

E0	0 0	0 - 0	BDI 0	E2	1 0	0 - 0	BDI 2
E1	0 1	0 - 0	BDI 1	E3	1 1	0 - 0	BDI 3
35	34 33	32 30	29 18	17	16 15	14 12	11 0

8.2.3. Interrupt Status

Interrupt status is information associated with a particular type of interrupt, and is stored only when its type of interrupt occurs. Immediate Storage Check status is stored in GRS location 054, Guard Mode status is stored in GRS location 053, Guard Mode Current instruction is stored in GRS location 056, Guard Mode Current Register Set Pointer is stored in GRS location 057, and all other CPU-generated interrupt status is stored in the Normal status location, GRS 055. Interrupt status is associated with the following types of interrupts:

- Immediate Storage Check interrupts
- Guard Mode interrupts
- Executive Request, Test and Set, and Invalid Instruction interrupts
- Delayed Check interrupts
- Breakpoint interrupts
- All I/O interrupts (Machine Check, Normal, and Table)
- Addressing Exception interrupts
- Interprocessor interrupts
- SSP interrupts

8.3. Interrupt Types

The CPU provides 26 interrupt priorities. The interrupt types are listed in Table 8-1.

8.3.1. Program Exception Interrupts

- Invalid Instruction – This interrupt occurs when the CPU attempts to execute an instruction with an invalid function code. The operand address of the instruction (24 bits of U) is stored in the GRS location 055 as interrupt status.

- Guard Mode – This interrupt occurs in the following cases:

  - When Privileged Instruction and GRS Protection designator (D2) is 1, and the execution of a privileged instruction is attempted or interrupt lockout period is exceeded. The interrupt is also set if 256 storage references are made during the Edit Decimal instruction when D2 is 1.

  - When violating the storage limit if Relocation and Storage Suppression designator (D7) is 0.

  - When attempting to store in GRS locations other than those allowed for the user (40_8 through 100_8 and 120_8 through 177_8) when D2 is 1.

  - When attempting to write into a storage area specified by bank descriptor register (BDR0, BDR1, BDR2, or BDR3) for which the corresponding write protect designator bit (D13 through D16) is 1.

See following for the format of the Guard Mode interrupt status stored in the GRS during the interrupt sequence:

Guard Mode Interrupt Status Format

WP	BDR	SL	IF	SO	IL	CR	PI	Zeros	Effective U
35	34 33	32	31	30	29	28	27	26 24	23 0

where:

Bit 35	Write Protection (WP) violation
Bits 34-33	Bank Descriptor Register (BDR) number associated with write protection violation
Bit 32	Storage Limits (SL) violation
Bit 31	Violation on normal Instruction Fetch (IF)
Bit 30	Violation on Second Operand (SO) fetch
Bit 29	Interrupt Lockout (IL) exceeded
Bit 28	Control Register (CR) violation
Bit 27	Privileged Instruction (PI) violation
Bits 26-24	Are zeros
Bits 23-0	Effective U

The following fields provide Guard Mode status information when a Storage Limits or a Write Protection violation occurs. They are undefined for privileged instruction, control register, or interrupt lockout violations.

- The instruction fetch bit (bit 31 of the Guard Mode Interrupt Status Word) is 1 if a Guard Mode violation occurs on a normal instruction fetch. It is 0 if the violation occurred during an operand fetch, a Jump To instruction fetch, an execute remote instruction fetch, or a fetch of the new x-, h-, i-, and u-fields during indirect addressing.

- The second operand bit (bit 30 of the Guard Mode Interrupt Status Word) is 0 if the violation occurs on the first operand fetch of a non-repeated instruction or repeated instruction pass. It is 1 if the violation occurs on the second operand fetch. It is 0 if the violation is on an instruction or indirect fetch.

- The Effective U-field (bits 23-0 of the Guard Mode Interrupt Status Word) is the relative address of the operand if a violation occurs during an operand fetch. The field is undefined if the violation occurs during a normal instruction fetch. It is the address of the Jump To instruction if the violation occurs during a Jump To instruction fetch. It is the address of the remote instruction if the violation occurs during an Execute instruction. It is the address of the new x-, h-, i-, and u-fields if the violation occurs during an execute instruction.

Current Instruction

When a Guard Mode interrupt occurs, the instruction being executed is stored in GRS location 056. In the case of indirect addressing, the instruction consists of the original f-, j-, a-fields and the x-, h-, i-, and u-fields that generated the invalid address. If the remote instruction fetch fails during an execute remote, the Execute instruction is stored. If the Jump To instruction fetch fails during a jump, the Jump instruction is stored. If a failure occurs on any instruction fetch, other than an execute remote or a jump, the current instruction field is undefined.

Current Register Set Pointer

If a violation occurs on an operand fetch during the execution of a Load Register Set (72,17) instruction, the current values of the area 1 and 2 counts and the area 1 and 2 addresses are stored in GRS location 057. The current address is the GRS address into which the operand would have been stored. The current count is the number of operands that remained to be transferred when the violation occurred; i.e., if the failure occurred on the last operand fetch, the count is 1. If the failure occurs during the transfer of an operand in area 2, the current count in area 1 is 0. The GRS location is undefined if the violation occurs during any other instruction or during any instruction or indirect fetch.

0 0	Area 2 Current Count	0 0	Area 2 Current Address	0 0	Area 1 Current Count	0 0	Area 1 Current Address
35 34	33 27	26 25	24 18	17 16	15 9	8 7	6 0

- Addressing Exception – This interrupt occurs in the following cases:
  - E-bit violation – If the E-bit (bit 35) from X_a of a LBJ, Load I-Bank Base and Jump (LIJ) or Load D-Bank Base and Jump (LDJ) instruction is 1 and EXEC Bank Descriptor Table Pointer Enable designator (D19) is 0.
  - Table length violation – If a Bank Descriptor Index (BDI) value from X_a of an LBJ, LIJ, or LDJ instruction is greater than the selected Bank Descriptor Table Pointer (BDTP) length value.
  - Residency interrupt – If the R-flag of the new bank descriptor is 1.
  - Entry point violation – If the V-flag of the new bank descriptor is 1 and the jump (LIJ, LDJ, LBJ) does not go to the beginning address of the bank being loaded by the instruction.
  - Use count overflow on LBJ, LIJ, or LDJ new bank descriptor.
  - Use count underflow on LBJ, LIJ, or LDJ old bank descriptor.
  - Use count decreased to 0 and the C-flag was 1.

See following format for the Addressing Exception interrupt stored in the GRS during the interrupt sequence. The program return address stored for this interrupt is P+1 for E-bit or table length violations, and the jump to address for all other violations.

Addressing Exception Interrupt Status Format

EN	V	E	R	CO	T	New BDI	EO	BDR	O	CU	CZ	Old BDI
35	34	33	32	31	30	29 18	17	16 15	14	13	12	11 0

where:

Bit 35	The new bank descriptor E-flag specification from X_a.
Bit 34	The V-flag indicates an entry point violation on the new bank descriptor.
Bit 33	The E-flag indicates an E-bit violation on the new bank descriptor.
Bit 32	The R-flag indicates the residency flag of the new bank descriptor was 1.
Bit 31	The CO-flag indicates a use count overflow on the new bank descriptor.
Bit 30	The T-flag indicates a table length violation on the new bank descriptor.
Bits 29–18	The new Bank Descriptor Index specification from X_a.
Bit 17	The old bank descriptor E-flag specification from the GRS.
Bits 16–15	The Old bank descriptor register specification from X_a.
Bit 14	Is zero.
Bit 13	The CU-flag indicates a use count underflow on the old bank descriptor.
Bit 12	The CZ-flag indicates the old bank descriptor use count was decreased from 1 to 0 and the C-flag was 1.
Bits 11–0	The old Bank Descriptor Index specification from the GRS.

NOTE: This interrupt results only from the execution of an LBJ, LIJ, or LDJ instruction. The new bank descriptor specifications are contained in X_a before execution; the old bank descriptor specifications are contained in GRS locations 046 and 047, and are placed in X_a during execution.

8.3.2. Arithmetic Exception Interrupts

An interrupt occurs in the following cases only if the Arithmetic Exception Interrupt designator (D20) is 1.

- Characteristic Overflow – Occurs when the exponent value of a floating-point result is greater than $+127_{10}$ (single precision) or $+1023_{10}$ (double precision). When this condition is detected, Characteristic Overflow designator (D22) is set to 1.

- Characteristic Underflow – Occurs when the exponent value of a floating-point result is less than -128_{10} (single precision) or -1024_{10} (double precision). When this condition is detected, Characteristic Underflow designator (D21) is set to 1.

- Divide Check – Occurs when the magnitude of the quotient exceeds the range of the specified register. When this condition is detected, Divide Check designator (D23) is set to 1.

8.3.3. Program-Initiated Interrupts

- Executive Request – This interrupt occurs as a result of executing an Executive Request (ER) instruction. This instruction allows a worker program to release control of the CPU to the Executive System. The operand address of the instruction (24 bits of U) is stored in GRS as interrupt status.

- Test and Set – This interrupt occurs as a result of executing a Test and Set (TS) instruction if bit 30 of the operand is 1. The operand address of the instruction (24 bits of U) is stored in GRS as interrupt status.

- Jump History Stack – This interrupt occurs when the jump history stack is full, if the S–flag (bit 34) of the breakpoint register is 1.

- Breakpoint – This interrupt occurs when an equality comparison is made between the contents of the breakpoint register and an instruction or operand address. The breakpoint interrupt condition is discarded if a higher priority internal interrupt occurs within the same instruction. See the following format for the Breakpoint interrupt status:

Breakpoint Interrupt Status Format

0 - 0	P	R	W	0 ——— 0	Absolute Breakpoint Address
35 33	32	31	30	29 24	23 0

where:

Bits 35–33	Are zeros.
Bit 32	The P–flag indicates an instruction address breakpoint.
Bit 31	The R–flag indicates an operand address breakpoint during a read operation.
Bit 30	The W–flag indicates an operand address breakpoint during a write operation.
Bits 29–24	Are zeros.
Bits 23–0	The Absolute Breakpoint Address.

NOTE: *For both instruction address breakpoints (P–flag) and operand address breakpoints (R– or W–flags), the instruction is executed and the program return address is captured.*

8.3.4. Interprocessor Interrupt

The Interprocessor interrupt occurs when a CPU in a system executes an Initiate Interprocessor Interrupt (IIIX) instruction. The interrupting CPU number can be determined from the status word stored in GRS address 055_8 during the interrupt sequence. See the following format for the interprocessor interrupt status:

Interprocessor Interrupt Status Format

Zeros	CPU
35 2	1 0

where:

Bits 35-2	Are zeros.
Bits 1-0	The interrupting CPU number.

8.3.5. Clock Interrupts

- Quantum Timer - A Quantum Timer interrupt occurs when the quantum timer value reaches 0.
- Real-Time Clock - This interrupt occurs when the contents of the lower 18 bits of the Real-Time Clock (RTC) register (GRS address 100_8) is decreased to 0. The value contained in the RTC is decreased by 1 every 200 microseconds. The RTC oscillator is accurate to ±0.06 percent.
- Dayclock - This interrupt request is made to all CPUs in the system once every 6.5536 seconds. Only one CPU may honor each request. The dayclock value is increased by 1 every 200 microseconds.

8.3.6. Immediate Storage Check Interrupts

Immediate Storage Check interrupt conditions are related to the current CPU macro-operation. Immediate Storage Check interrupts cannot be locked out by software. An Immediate Storage Check interrupt occurs when either an unretriable fault is detected or a retry operation encounters a fault.

The immediate check status word provides diagnostic information for easier fault isolation. The immediate check status word is stored in GRS location 054. The format for the three immediate check status words is as follows:

Immediate Check Interrupt Status for Storage Interface Checks

0	1	1	MR	OP	RF	MUE	ANA	RDPC	WDPC	CPC	APC	Absolute Address
35	34	33	32	31	30	29	28	27	26	25	24	23 0

where:

Bits 35-33	The code of 3_8 identifies the status word as a storage interface check status word.
Bit 32	Macro-instruction Reference (MR) indicates that the fault was encountered during the execution of an instruction. If this bit is not set, the fault was encountered during the execution of an auxiliary operation (dayclock update, interrupt status handling, etc.).
Bit 31	OPerand (OP) indicates that the fault was encountered during the fetching or storing of an operand. If this bit is not set, the fault was encountered during the fetch of an instruction. This bit is valid only when bit 32 is set.

Bit 30	Retry Failed (RF) indicates that a fault was detected, a retry operation was initiated, and the retry failed. If this bit is not set, the original fault occurred during an unretriable microinstruction sequence (for example, multiple operand instructions).
Bit 29	Multiple Uncorrectable Error (MUE) indicates that a storage reference encountered a multiple bit uncorrectable error on the requested word.
Bit 28	Address Not Available (ANA) indicates that a storage reference encountered one of the following faults: 1. The addressed word is truly not available, or 2. the storage reference encountered an SIU/MSU interface fault.
Bit 27	Read Data Parity Check (RDPC) indicates that a storage reference encountered a read data parity error.
Bit 26	Write Data Parity Check (WDPC) indicates that a storage reference encountered a write data parity error.
Bit 25	Control Parity Check (CPC) indicates that a storage reference encountered a control signal parity check.
Bit 24	Address Parity Check (APC) indicates that a storage reference encountered an address parity.
Bits 23–0	The absolute address that is associated with the fault.

NOTE: When both bit 29 (MUE) and bit 28 (ANA) are set, the MSU detects the special code on the read of the requested word.

Immediate Check Interrupt Status Internal Check 1 Format

1	0	0	MO	RF	Zeros	CS13	CS12	CS11	CS10	CS9	CS8	CS7	CS6	CS5	CS4	CS3	CS2	CS1	CS0	Micro Address
35	34	33	32 30	29	28 25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10 0

where:

Bits 35–33	The code of 4_8 identifies the status word as an internal check 1 status word.
Bits 32–30	Macro-Operation (MO) indicates the three-bit code being executed when the fault was detected.

Bits 32–30	Macro-Operation
0_8	idle loop
1_8	macroinstruction
2_8	software interrupt
3_8	SSP operation
4_8	dayclock or real-time clock update
5_8	retry

Bit 29	Retry Failed (RF) indicates that a fault was detected, a retry operation was initiated, and the retry failed. If this bit is not set, the original fault occurred during an unretriable microinstruction sequence.
Bits 28–25	Are zeros.
Bit 24	Control Store Card 13 (CS13) indicates that a fault was detected on microcontrol store card 13.
Bit 23	Control Store Card 12 (CS12) indicates that a fault was detected on microcontrol store card 12.
Bit 22	Control Store Card 11 (CS11) indicates that a fault was detected on microcontrol store card 11.
Bit 21	Control Store Card 10 (CS10) indicates that a fault was detected on microcontrol store card 10.
Bit 20	Control Store Card 9 (CS9) indicates that a fault was detected on microcontrol store card 9.
Bit 19	Control Store Card 8 (CS8) indicates that a fault was detected on microcontrol store card 8.
Bit 18	Control Store Card 7 (CS7) indicates that a fault was detected on microcontrol store card 7.
Bit 17	Control Store Card 6 (CS6) indicates that a fault was detected on microcontrol store card 6.
Bit 16	Control Store Card 5 (CS5) indicates that a fault was detected on microcontrol store card 5.
Bit 15	Control Store Card 4 (CS4) indicates that a fault was detected on microcontrol store card 4.
Bit 14	Control Store Card 3 (CS3) indicates that a fault was detected on microcontrol store card 3.
Bit 13	Control Store Card 2 (CS2) indicates that a fault was detected on microcontrol store card 2.
Bit 12	Control Store Card 1 (CS1) indicates that a fault was detected on microcontrol store card 1.
Bit 11	Control Store Card 0 (CS0) indicates that a fault was detected on microcontrol store card 0.
Bits 10–0	Micro Address indicates address of the failing microinstruction.

Immediate Check Interrupt Status Internal Check 2 Format

1	0	1	MO	RF	MP	MBM	GPC	LM2U	LM2L	LM1U	LM1L	IDS	SM	DM	LFM	SPP	SIS	Zeros	Micro Address
35	34	33	32 30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15 11	10 0

where:

Bits 35-33	The code of 5_8 identifies the status word as an internal check 2 status word.
Bits 32-30	Macro-Operation (MO) indicates the three-bit code being executed when the fault was detected.

Bits 32-30	Macro-Operation
0_8	idle loop
1_8	macroinstruction
2_8	software interrupt
3_8	SSP operation
4_8	dayclock or real-time clock update
5_8	retry

Bit 29	Retry Failed (RF) indicates that a fault was detected, a retry operation was initiated, and the retry failed. If this bit is not set, the original fault occurred during an unretriable microinstruction sequence.
Bit 28	Microprocessor Pointer (MP) indicates the microprocessor that was gating data to the microprocessor bus when the fault was detected. This bit is valid only if bit 27 is set.
Bit 27	Microprocessor Bus Miscompare (MBM) indicates that the comparator for the microprocessor bus detected a fault.
Bit 26	GRS Parity Check (GPC) indicates that a GRS parity fault was detected.
Bit 25	Local Memory 2 Upper (LM2U) indicates that a parity fault was detected on the upper half of local memory 2.
Bit 24	Local Memory 2 Lower (LM2L) indicates that a parity fault was detected on the lower half of local memory 2.
Bit 23	Local Memory 1 Upper (LM1U) indicates that a parity fault was detected on the upper half of local memory 1.
Bit 22	Local Memory 1 Lower (LM1L) indicates that a parity fault was detected on the lower half of local memory 1.
Bit 21	Instruction Decode Storage (IDS) indicates that a parity fault was detected on the instruction decode storage.
Bit 20	Shifter Miscompare (SM) indicates that the comparator for the shifter logic detected a fault.
Bit 19	Dispatcher Miscompare (DM) indicates that the comparator for the dispatcher logic detected a fault.
Bit 18	Logic Function Miscompare (LFM) indicates that the comparator for the logic function matrix detected a fault.
Bit 17	Shift PROM Parity (SPP) indicates that the control memory for the shifter detected a parity fault.

Bit 16	Shift Input Selector (SIS) indicates that a parity fault was detected on the GRS, internal data bus, or storage operand data on the shift input selector.
Bits 15–11	Are zeros.
Bits 10–0	Micro Address indicates approximate address of the failing microinstruction.

8.3.7. Delayed Storage Check Interrupts

Delayed Storage Check interrupts have a unique entrance location, and the status associated with them is stored in the normal status location.

Delayed Check interrupts report hardware faults detected by the CPU, the SIU, and the MSU. Delayed check status words provide diagnostic information for fault isolation and fault recovery. Delayed Check interrupts report retriable CPU faults that have been successfully retried. Bits 35–33 of the status word indicate the system component that is reporting the fault, as follows:

Bits 35–33	Description
0_8	SIU Delayed Check (see 6.2.3.1)
2_8	Main Storage Unit Delayed Check (see 6.3.2.6)
3_8 or 4_8 or 5_8	CPU Delayed Check

Storage Delayed Check Handling

When the CPU receives a Delayed Check interrupt from the SIU or MSU (when bits 35–33 of the status word = 0_8, 1_8, or 2_8), the interrupt is deferred for a certain period of time, depending on the type of instruction being executed. The following cases exist:

- If the CPU receives a Delayed Check interrupt during the execution of a non-interruptable instruction P, the CPU will defer handling the interrupt until after the completion of the instruction P + 1.
- If the CPU receives a Delayed Check interrupt during the execution of an interruptable (repeated) instruction, it will defer handling the interrupt until recurrence of instruction is complete.
- If the CPU receives a Delayed Check interrupt associated with the first operand of a repeated instruction, and if this operand is prefetched before the repeated instruction begins, the interrupt is not handled until every recurrence of the repeated instruction is complete.

Delayed Check Interrupt Status for Storage Interface Checks

0	1	1	MR	OP	0	MUE	ANA	RDPC	WDPC	CPC	APC	Absolute Address
35	34	33	32	31	30	29	28	27	26	25	24	23 … 0

where:

Bits 35–33	The code of 3_8 identifies the status word as a storage interface check status word.
Bit 32	Macro–instruction Reference (MR) indicates that the fault was encountered during the execution of an instruction. If this bit is not set, the fault was encountered during the execution of an auxiliary operation (dayclock update, interrupt status handling, etc.).
Bit 31	OPerand (OP) indicates that the fault was encountered during the fetching or storing of an operand. If this bit is not set, the fault was encountered during the fetch of an instruction. This bit is valid only when bit 32 is set.
Bit 30	Is zero.
Bit 29	Multiple Uncorrectable Error (MUS) indicates that a storage reference encountered a multiple bit uncorrectable error on the requested word.
Bit 28	Address Not Available (ANA) indicates that a storage reference encountered one of the following: ■ The addressed word is not available, or ■ the storage reference encountered an SIU/MSU interface fault.
Bit 27	Read Data Parity Check (RDPC) indicates that a storage reference encountered a read data parity error.
Bit 26	Write Data Parity Check (WDPC) indicates that a storage reference encountered a write data parity error.
Bit 25	Control Parity Check (CPC) indicates that a storage reference encountered a control signal parity check.
Bit 24	Address Parity Check (APC) indicates that a storage reference encountered an address parity.
Bits 23–0	The absolute address associated with the fault.

Delayed Check Interrupt Status Internal Check 1

1	0	0	MO	Zeros	CS13	CS12	CS11	CS10	CS9	CS8	CS7	CS6	CS5	CS4	CS3	CS2	CS1	CS0	Micro Address
35	34	33	32–30	29–25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10–0

where:

Bits 35–33	The code of 4_8 identifies the status word as an internal check 1 status word.
Bits 32–30	Macro–Operation (MO) indicates the three–bit code being executed when the fault was detected.

Bits 32-30	Macro-Operation
0_8	idle loop
1_8	macroinstruction
2_8	software interrupt
3_8	SSP operation
4_8	dayclock or real-time clock update
5_8	retry

Bits 29-25	Are zeros.
Bit 24	Control Store Card 13 (CS13) indicates that a fault was detected on microcontrol store card 13.
Bit 23	Control Store Card 12 (CS12) indicates that a fault was detected on microcontrol store card 12.
Bit 22	Control Store Card 11 (CS11) indicates that a fault was detected on microcontrol store card 11.
Bit 21	Control Store Card 10 (CS10) indicates that a fault was detected on microcontrol store card 10.
Bit 20	Control Store Card 9 (CS9) indicates that a fault was detected on microcontrol store card 9.
Bit 19	Control Store Card 8 (CS8) indicates that a fault was detected on microcontrol store card 8.
Bit 18	Control Store Card 7 (CS7) indicates that a fault was detected on microcontrol store card 7.
Bit 17	Control Store Card 6 (CS6) indicates that a fault was detected on microcontrol store card 6.
Bit 16	Control Store Card 5 (CS5) indicates that a fault was detected on microcontrol store card 5.
Bit 15	Control Store Card 4 (CS4) indicates that a fault was detected on microcontrol store card 4.
Bit 14	Control Store Card 3 (CS3) indicates that a fault was detected on microcontrol store card 3.
Bit 13	Control Store Card 2 (CS2) indicates that a fault was detected on microcontrol store card 2.
Bit 12	Control Store Card 1 (CS1) indicates that a fault was detected on microcontrol store card 1.
Bit 11	Control Store Card 0 (CS0) indicates that a fault was detected on microcontrol store card 0.
Bits 10-0	Micro Address indicates address of the failing microinstruction.

Delayed Check Interrupt Status Internal Check 2 Format

1	0	1	MO	0	MP	MBM	GPC	LM2U	LM2L	LM1U	LM1L	IDS	SM	DM	LFM	SPP	SIS	Zeros	Micro Address
35	34	33	32 30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15 11	10 0

where:

Bits 35–33 The code of 5_8 identifies the status word as an internal check 2 status word.

Bits 32–30 Macro–Operation (MO) indicates the three–bit code being executed when the fault was detected.

Bits 32–30	Macro–Operation
0_8	idle loop
1_8	macroinstruction
2_8	software interrupt
3_8	SSP operation
4_8	dayclock or real–time clock update
5_8	retry

Bit 29 Is zero.

Bit 28 Microprocessor Pointer (MP) indicates the microprocessor that was gating data to the microprocessor bus when the fault was detected. This bit is valid only if bit 27 is set.

Bit 27 Microprocessor Bus Miscompare (MBM) indicates that the comparator for the microprocessor bus detected a fault.

Bit 26 GRS Parity Check (GPC) indicates that a GRS parity fault was detected.

Bit 25 Local Memory 2 Upper (LM2U) indicates that a parity fault was detected on the upper half of local memory 2.

Bit 24 Local Memory 2 Lower (LM2L) indicates that a parity fault was detected on the lower half of local memory 2.

Bit 23 Local Memory 1 Upper (LM1U) indicates that a parity fault was detected on the upper half of local memory 1.

Bit 22 Local Memory 1 Lower (LM1L) indicates that a parity fault was detected on the lower half of local memory 1.

Bit 21 Instruction Decode Storage (IDS) indicates that a parity fault was detected on the instruction decode storage.

Bit 20 Shifter Miscompare (SM) indicates that the comparator for the shifter logic detected a fault.

Bit 19 Dispatcher Miscompare (DM) indicates that the comparator for the dispatcher logic detected a fault.

Bit 18	Logic Function Miscompare (LFM) indicates that the comparator for the logic function matrix detected a fault.
Bit 17	Shift PROM Parity (SPP) indicates that the control memory for the shifter detected a parity fault.
Bit 16	Shift Input Selector (SIS) indicates that a parity fault was detected on GRS, internal data bus, or storage operand data on the shift input selector.
Bits 15-11	Are zeros.
Bits 10-0	Micro Address indicates approximate address of the failing microinstruction.

8.3.8. Multiprocessor Interrupt Synchronization

All external interrupt requests, for example those generated by IOUs, are presented to each CPU. An interlocked synchronization mechanism is provided to assure that only one CPU actually accepts each interrupt request.

If a decision is made by a CPU to honor an interrupt request, the propagation of synchronization control is delayed until the interrupt request is acknowledged, and the interrupt request is deactivated.

A CPU always retains interrupt synchronization control. In a multiprocessor system where the other CPU is offline or powered down, and if, during initial load, a CPU is selected for taking the initial load interrupt, the SSP selects that CPU to retain interrupt synchronization control.

8.4. Input/Output Interrupts

All IOU status information is reported via interrupt. There are three interrupt mechanisms: 1) the Machine Check interrupts, 2) the Normal interrupts, and 3) the Table interrupts. All IOU status that cannot be associated with a particular subchannel is reported via Machine Check interrupt. This includes all errors on the IOU/MSU interface and all internal IOU errors.

All IOU status associated with an Internally Specified Index (ISI) word or block multiplexer subchannel is reported via normal interrupt. All IOU status associated with an Externally Specified Index (ESI) word subchannel is placed in a status table that is controlled by the status table subchannel. Entries into the status table may generate a Table interrupt request depending upon the interrupt mask.

8.4.1. Machine Check Interrupts

When a hardware fault not associated with a particular subchannel, or storage fault, or internal IOU fault is detected, a Machine Check Status Word (MCSW) is built and held in the IOU. When interrupts are allowed by a CPU, the MCSW is stored in a reserved storage address location and a Machine Check interrupt request is generated. The following is the format for an IOU Machine Check Status Word.

IOU Machine Check Status Word Format

Zeros	SSPE	CDPE	RAPC	CSPE	CFPE	CPC	DAPE	CAPE	ARC	MUE	MSNA	MSRD	ANA	MSWD	MSWC	MSAC	IOU No.	Subchannel Address
35	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12 11 10	9 0

where:

Bits 35–28	Are zeros.
Bit 27	SIOF Stack Parity Error (SSPE) indicates that a parity error has been detected on one of the Start I/O Fast Release (SIOF) stacks.
Bit 26	Channel Descriptor Parity Error (CDPE) indicates that the IOU detected a parity error on the channel descriptor integrated General Register (IGR) stack.
Bit 25	Retry CCW Address Parity Check (RAPC) indicates that the IOU has detected a parity error on the retry CCW address field of the IGR stack (only if compatible channel feature is installed).
Bit 24	Channel Status Parity Error (CSPE) indicates that the IOU has detected a parity error on the subchannel status field of the IGR stack.
Bit 23	Control Flag Parity Error (CFPE) indicates that an IGR parity error has been detected on the control flag field of a subchannel control word.
Bit 22	Count Parity Error (CPE) indicates that an IGR parity error has been detected on the data count or format count field of a subchannel control word.
Bit 21	Data Address Parity Error (DAPE) indicates that an IGR parity error has been detected on the data address field of a subchannel control word.
Bit 20	CCW Address Parity Error (CAPE) indicates that an IGR parity error has been detected on the next Channel Command Word (CCW) address field of a subchannel control word.
Bit 19	Arithmetic Redundancy Check (ARC) indicates that a fault has been detected by the duplicated arithmetic unit.
Bit 18	Multiple Uncorrectable Error (MUE) indicates that a multiple uncorrectable error has occurred within the main storage read data word. See ANA, bit 15.
Bit 17	Main Storage Not Available (MSNA) indicates that the IOU detected an unavailable MSU.
Bit 16	Main Storage Read Data check (MSRD) indicates that the IOU detected a parity error on read data from the MSU.
Bit 15	Address Not Available (ANA) indicates that the MSU detected an unavailable address. *NOTE: ANA and MUE together indicate that the MSU detected a special code on the requested word.*
Bit 14	Main Storage Write Data check (MSWD) indicates that the MSU detected a write data parity error on an IOU request.
Bit 13	Main Storage Write Control check (MSWC) indicates that the MSU detected a write control parity error on an IOU request.

Bit 12 Main Storage Address Check (MSAC) indicates that the MSU detected an address parity error on an IOU request.

Bits 11–10 IOU Number indicates the IOU that is generating the interrupt.

Bits 9–0 Subchannel Address contains the subchannel address for all errors.

NOTE: The subchannel address field contains the last subchannel that generated machine check status. The bits within the MCSW are a composite of faults from all subchannels since the CPU acknowledged the last MCSW. This field is not valid if SSPE is set.

8.4.2. Normal Interrupts

All status information that is associated with an ISI word, block multiplexer, or status table subchannel is reported via the Normal interrupt mechanism. When interrupts are allowed, a two- or three- word Channel Status Word (CSW) is stored in reserved storage locations, and a Normal interrupt request is generated. The format description of the CSW for the various types of subchannels are in the following subsections.

8.4.2.1. ISI Word Interface CSW

IOU	Subchannel Address	Next CCW Address
35 34	33 24	23 0

Not Used	ATN	Zeros	Subchannel Status	Residual Word Count
71 68	67	66 60	59 52	51 36

External Interrupt Status Word
107 72

where:

Bits 35–34 IOU contains the IOU number.

Bits 33–24 Subchannel Address contains the address of the subchannel presenting the status information.

Bits 23–0 Next CCW Address contains the next CCW address at the time the status information is stored.

NOTE: A CSW associated with a string of data chained CCWs may have the incorrect next address. However, it will contain an address within the chain initiated by the last command issued by a SIOF or a command chain operation.

If a Transfer In Channel (TIC) is encountered during a command or data chain operation, the next CCW address may contain the TIC address, the TIC to address, or the TIC to address plus two.

Bits 71-68 Not Used.

Bit 67 Attention (ATN) indicates that a valid External Interrupt status word has been stored as word 3 of a CSW.

Bits 66-60 Are zeros.

Bits 59-52 Subchannel Status indications are set when a program information, software fault, or hardware fault condition is detected by the subchannel. Subchannel status is reported to the software via the subchannel status field of a CSW. The subchannel status byte includes the following status indications.

Bit 59 Is zero.

Bit 58 Monitor Interrupt indicates that the subchannel has detected a monitor condition. The monitor condition is defined as word count exhausted, CC flag clear, CD flag clear, MONitor (MON) flag set, no hardware fault detected by the subchannel, and a data request from the peripheral.

Bit 57 Program Check indicates that the hardware detected one of the following software faults:

1. A data chain CCW specified a data count of zero.
2. A CCW contained an invalid command for an operation other than a data chain.
3. A CCW address specified by a TIC command was not on a double-word boundary.
4. A command CCW with a data count of zero specified data chaining.

Bit 56 Internal Hardware Check indicates that an operation for this subchannel encountered an internal hardware fault.

Bit 55 Storage Check indicates that a storage request for this subchannel encountered a storage fault.

Bit 54 Channel Module Bus Check indicates that the channel module interface detected a peripheral parity error on input or a through data parity error on output.

Bit 53 Channel Module El Check indicates that the channel module was unable to report a device External Interrupt (EI) word because it contains a parity error.

Bit 52 Is zero.

Bits 51–36 Residual Word Count contains the residual word count for the associated CCW. The residual word count for an output operation is the number of words left in the buffer of the channel module when the operation is terminated, plus the decremented word count that is contained in the IOU's internal update copy of the active CCW. On input operations the IOU's internal CCW word count is the residual word count. If a storage check or an internal check occurred during an input transfer, the count field is not valid.

NOTE: *A CSW associated with a string of data chained CCWs for an output operation may have a residual count larger than the original count for that CCW. However, the software will still be able to determine the exact number of words transferred to the peripheral by examining the list of CCWs.*

Bits 107–72 External Interrupt Status Word contains the 36-bit EI word presented by the peripheral. This field is valid only if the Attention bit of the CSW is set.

8.4.2.2. Block Multiplexer CSW

IOU	Subchannel Address	Next CCW Address
35 34	33 24	23 0

Residual Byte Count	Device Status	Subchannel Status	Residual Word Count
71 68	67 60	59 52	51 36

where:

Bits 35–34 IOU contains the IOU number.

Bits 33–24 Subchannel Address contains the address of the subchannel presenting the status information.

Bits 23–0 Next CCW Address contains the next CCW address at the time the status information is stored.

NOTE: *A CSW associated with a string of data chained CCWs may have the incorrect next address. However, it will contain an address within the chain initiated by the last command issued by a SIOF or a command chain operation.*

If a TIC is encountered during a command or data chain operation, the next CCW address may contain the TIC CCW address, the CCW address the TIC CCW points to, or the CCW address the TIC CCW points to plus two.

Bits 71–68 Residual Byte Count indicates the number of unfilled bytes on input and unused bytes on output for the last data word transferred under control of the current CCW.

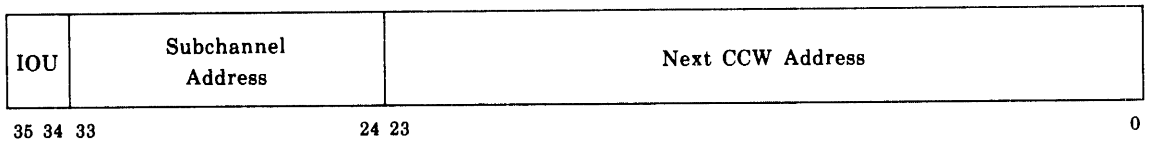
Bits 67–60	Device Status contains a device-generated status byte if one or more of the following conditions are detected and reported by the device:
Bit 67	Attention
Bit 66	Status Modifier
Bit 65	Control Unit End
Bit 64	Busy
Bit 63	Channel End
Bit 62	Device End
Bit 61	Unit Check
Bit 60	Unit Exception
Bit 59–52	Subchannel Status channel status indications are set when a status condition, a software fault, or a hardware fault is detected by the subchannel. Channel status is reported to the software via the subchannel status field of a CSW. The subchannel status byte includes the following status indications.
Bit 59	Is zero.
Bit 58	Incorrect Length indicates that the number of words specified in the CCWs for an I/O operation did not equal the number of bytes requested or offered by the device. The SIL CCW flag disables the reporting of an incorrect length condition.
Bit 57	Program Check indicates that the hardware detected one of the following software faults: 1. A data chain CCW specified a data count of zero. 2. A CCW contained an invalid command for an operation other than a data chain. 3. A CCW specified by a TIC command was not on a double-word boundary. 4. A CCW did not specify only one data format. 5. A command CCW with a data count of zero specified data chaining.
Bit 56	Internal Hardware Check indicates that an operation for this subchannel encountered an internal hardware fault.
Bit 55	Storage Check indicates that a storage request for this subchannel encountered a storage fault.
Bit 54	Channel Bus Check indicates that the channel detected a bus input parity error from the peripheral or an output data parity error.
Bit 53	Channel Control Check indicates that the channel interface detected a control signal sequence fault, an interface timed out fault, or a DISCONNECT IN signal from a control unit.

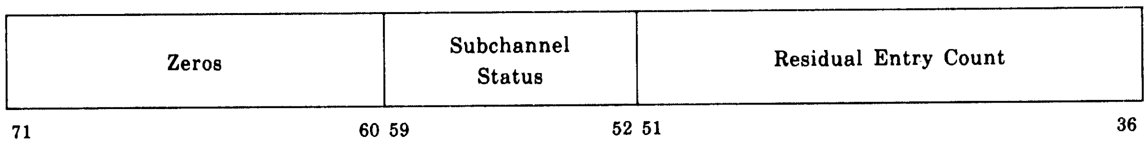
Bit 52 Device Not Available Check indicates that the device is offline or powered down. This condition is indicated to the IOU by the SELECT IN signal during the initial selection sequence.

Bits 51-36 Residual Word Count contains the residual word count for the associated CCW. The residual word count for an output operation is the number of words left in the buffer of the channel module when the operation is terminated, plus the decremented word count that is contained in the IOU's internal updated copy of the active CCW. On input operations, the IOU's internal CCW word count is the residual word count. If a storage check or an internal check occurred during an input data transfer, the count field is not valid. Incomplete words transferred are defined in the residual byte count field.

NOTE: *A CSW associated with a string of data chained CCWs for an output operation may have a residual count larger than the original count for that CCW. However, the software will still be able to determine the exact number of words transferred to the peripheral by examining the list of CCWs.*

8.4.2.3. Status Table Subchannel CSW





where:

Bits 35-34 IOU contains the IOU number.

Bits 33-24 Subchannel Address contains the address of the status table subchannel (1777_8).

Bits 23-0 Next CCW Address contains the next CCW address at the time the status information is stored.

NOTE: *A CSW associated with a string of data chained CCWs may have the incorrect next address. However, it will contain an address within the chain initiated by the last command issued by a SIOF or a command chain operation.*

If a TIC is encountered during a command or data chain operation, the next CCW address may contain the TIC address, the TIC to address, or the TIC to address plus two.

Bits 71-60 Are zeros.

Bits 59–52	Subchannel Status Subchannel status indications are set when a status condition, a software fault, or a hardware fault is detected by the subchannel. Subchannel status is reported to the software via the subchannel status field of a CSW. The subchannel status byte includes the following status indications.
Bit 59	Is zero.
Bit 58	Monitor Interrupt indicates that the status table subchannel has detected the MON flag in a CCW. The subchannel generates an interrupt request as soon as possible after the entry count has been decremented to zero. Since data chaining occurs during the last data transfer controlled by the old CCW, the MON indication is presented after the last transfer of the old buffer rather than after one or more transfers are made under control of the new CCW.
Bit 57	Program Check indicates that the hardware detected one of the following software faults: 1. A data chain CCW specified a data count of zero. 2. The data address field of a CCW did not specify a four-word boundary. 3. A CCW address specified by a TIC command was not on a double-word boundary. 4. The table entry count has been exhausted and data chaining has not been specified.
Bit 56	Internal Hardware Check indicates that an operation for the status table subchannel encountered an internal hardware fault.
Bit 55	Storage Check indicates that a storage request for the status table subchannel encountered a storage fault.
Bits 54–52	Are zeros.
Bits 51–36	Residual Entry Count contains the residual entry count for the associated CCW.

8.4.2.4. Test Subchannel CSW

IOU No.	Subchannel Address	Integrated General Register
35 34	33 24	23 0

Residual Word Count	P	Data Address
71 61	60	59 36

Not Used	Sub. Mode	Subchannel Status	P	P	CCW Flag	P	FCF	Residual Word Count
107 106	105 103	102 91	90	89	88 80	79	78 77	76 72

where:

Bits 35–34	IOU No. contains the IOU number.
Bits 33–24	Subchannel Address contains the address of the subchannel presenting the information.
Bits 23–0	IGR contains the next CCW address at the time the status information is stored.
Bits 76–61	Residual Word Count contains the format and residual data count at the time the status information is stored.
Bit 60	P contains the parity bit generated as odd parity for IGR bits 47–24.
Bits 59–36	Data Address contains the data address at the time the status information is stored.
Bits 107–106	Not Used.
Bits 105–103	Subchannel Mode contains the mode of the channel at the time the CSW was written.
Bits 102–91	Subchannel Status Integrated General Register (IGR) bits 95–84.
Bit 90	P contains the parity bit generated as odd parity for IGR bits 23–0.
Bit 89	P contains the parity bit generated as odd parity for IGR bits 74–66.
Bits 88–80	CCW Flag contains the CCW flags active at the time status information is stored in IGR bits 74–66.
Bit 79	P contains the parity bit generated as odd parity for IGR bits 65–48.
Bits 78–77	The Format Control Flags (FCF) specify the location within a word of the first data character. In quarter-word mode, the format control flags specify the location of the first quarter word within the first word of data for each CCW.

78	77	
0	0	Specifies that the first quarter word be selected from or placed in bits 35–27 of the first data word.
0	1	Specifies that the first quarter word be selected from or placed in bits 26–18 of the first data word.
1	0	Specifies that the first quarter word be selected from or placed in bits 17–9 of the first data word.
1	1	Specifies that the first quarter word be selected from or placed in bits 8–0 of the first data word.

As each quarter word is used (input or output), the format count is updated by the IOU by +1. The counter counts modulo 4 with the carry from the fourth count used to update the data address field. The counts are updated after the data transfer, which leaves the IOU internal CCW pointing to the next word to be transferred.

In half-word mode, the format control flags specify the location of the first half word within the first word of data for each CCW.

77	
0	Specifies that the first half word be selected from or placed in bits 17-0 of the first data word.
1	Specifies that the first half word be selected from or placed in bits 35-18 of the first data word.

As each half word is used (input or output), the format flag is updated by the IOU by +1. The counter counts module 2 with the carry from the second count used to update the word count field. The counts are updated after the data transfer, which leaves the IOU internal CCW pointing to the next word to be transferred.

The IGR bits 65-64 control packing of ESI data characters into a 36-bit word. The value of these bits are unpredictable for non-ESI subchannels or for an ESI subchannel whose last operation was a forced External Function (EF).

8.4.3. Table Interrupts

All status information for ESI word subchannels is stored in a status table controlled by the status table subchannel. Each table entry consists of a two- or three-word Table Status Word (TSW). However, four words must be reserved for each table entry because each table entry is started on a four-word boundary. When an entry is made into the table, a table interrupt request is generated and a table interrupt CSW is stored in reserved storage. The table interrupt CSW format is:

Table Interrupt Channel Status Word

IOU	Status Table Subchannel Address	Next CCW Address
35 34	33 24	23 0

Zeros	Residual Entry Count
71 52	51 36

All status information for the status table subchannel (MON, storage checks, program checks, etc.) is reported via Normal interrupt and not by Table interrupt.

ESI Word Interface Table Status Word (TSW)

IOU	Subchannel Address	Next CCW Address
35 34	33 24	23 0

Not Used	ATN	Zeros	Subchannel Status	Residual Character Count
71 68	67	66 60	59 52	51 36

External Interrupt Status Word

107 72

where:

Bits 35–34 IOU contains the IOU number.

Bits 33–24 This field contains the subchannel address presenting the status information.

Bits 23–0 This field contains the next CCW address at the time the status information is stored.

NOTE: A CSW associated with a string of data chained CCWs may have the incorrect next address. However, it will contain an address within the chain initiated by the last command issued by a SIOF or a command chain operation.

If a TIC is encountered during a command or data chain operation, the next CCW address may contain the TIC address, the TIC to address, or the TIC to address plus two.

Bit 67 Attention (ATN) indicates that a valid EI status word has been stored as word 3 of a CSW.

Bits 66–60 Are zeros.

Bits 59–52 Subchannel Status indications are set when a status condition, a software fault, or a hardware fault is detected by the subchannel. Channel status is reported to the software via the subchannel status field of a CCW. The subchannel status byte includes the following status indications.

Bit 59 Is zero.

Bit 58 Monitor Interrupt indicates that the subchannel has detected a MON condition. The MON condition is defined as character count exhausted, MON flag set, and no hardware fault detected by the subchannel.

Bit 57 Program Check indicates that the hardware detected one of the following software faults:

1. A data chain CCW specified a character count of zero.
2. A CCW contained an invalid command for an operation other than a data chain.
3. The command code contained the forced EF command and the character count was not equal to one.
4. A CCW address specified by a TIC command was not on a double-word boundary.
5. A command CCW with a data count of zero specified data chaining.

6. The command code contained the forced EF command and data chaining was specified.

7. A Write command CCW specified command chaining.

Bit 56	Internal Hardware Check indicates that an operation for this subchannel encountered an internal hardware fault.
Bit 55	Storage Check indicates that a storage request for this subchannel encountered a storage fault.
Bits 54–52	Are zeros.
Bits 51–36	This field contains the residual character count for the associated CCW.
Bits 107–72	External Interrupt Status Word contains the 36-bit EI word presented by the peripheral. This field is valid only if the Attention bit of the CSW is set.

9. Instruction Repertoire

9.1. General

This section describes the operation performed by each instruction in the 1100/70 Systems user repertoire. These descriptions are grouped by types of instructions. (A summary of the instruction repertoire is given in Appendix C.)

An introduction to each group presents information that is common to all instructions in the group. The detailed descriptions of the individual instruction have the following format:

- Instruction name - Mnemonic code Octal function code (i.e., Load A - L,LA 10)
- Symbolic description of the operation performed by the instruction. The symbols used are defined in Appendix A.
- Textual description of the operation performed by the instruction.
- Sequentially numbered notes that provide special information related to the instruction, if appropriate.

For all instructions, any possible value may be used in the a-, x-, h-, i-, and u-fields unless an exception to this rule is stated in the notes. Any possible value may be used in the j-field except when j is a minor-function-code designator or when an exception is stated in the notes.

If the value of the j-field is 16_8 and 17_8 (an immediate operand specification) and the value of the x-field is 0, the h-, i-, and u-fields make up the 18-bit operand. If the h- and i-fields are 1 and the value of the u-field is 0177777, the resulting operand is 0, not all 1's. A -0 can be generated as an immediate operand only by load negative instructions using x-, h-, i-, and u-fields of 0.

If the value of the a-field of the instruction is 017 (A15) and the instruction makes use of more than one arithmetic register (A_{a+1} or A_{a+2}), those registers are located at General Register Set (GRS) location 034 and 035, or 0174 and 0175, depending on the value of GRS Selection designator (D6). If automatic index register incrementation occurs, the value of A_a or X_a is not affected. However, the value of U or U+1 (if U < 0200) or A_{a+1} (for two pass instructions, which require both U and U+1) may be affected; if X_x is referenced as one of these operands, the updated index value is used.

9.2. Load Instructions

The single-precision load instructions transfer data to the arithmetic section where a 36-bit word is always formed. The 36-bit word is then transferred to the register specified by the a-field of the instruction. Single-precision data-word transfers from storage to the arithmetic section are controlled by the value in the j-field.

For the double-precision load instructions, the j-field is a minor function code and full 72-bit data transfers result.

9.2.1. Load A - L,LA 10

$(U) \rightarrow A_a$

The contents of U is transferred under j-field control to the arithmetic section and then to A_a.

9.2.2. Load Negative A - LN,LNA 11

$-(U) \rightarrow A_a$

The contents of U is transferred under j-field control to the arithmetic section. The ones complement of the value in the arithmetic section is transferred to A_a.

9.2.3. Load Magnitude A - LM,LMA 12

$|(U)| \rightarrow A_a$

The contents of U is transferred under j-field control to the arithmetic section. If the sign bit (bit 35) of the value in the arithmetic section is 1, it is complemented; if the sign bit is 0, it is not complemented. The final value (always positive) is transferred from the arithmetic section to A_a.

For j-field values 0, 3-7 (quarter word not set), and 17, sign bit 35 is controlled by sign extension.

1. This instruction is the same as Load A (see 9.2.1) for j = H1, H2, Q1-Q4, or S1-S6.

9.2.4. Load Negative Magnitude A - LNMA 13

$-|(U)| \rightarrow A_a$

The contents of U is transferred under j-field control to the arithmetic section. If the sign bit (bit 35) of the value in the arithmetic section is 0, the value is complemented; if the sign bit is 1, it is not complemented. The final value (always negative) is transferred from the arithmetic section to A_a.

For j-field values 3-7 (quarter word not set), and 17_8, sign bit 35 is controlled by sign extension.

1. This instruction may be used to load -0 into an A-register by using $j = 16_8$ or 17_8, and $x = h = i = u = 0$.
2. This instruction is the same as Load Negative A (see 9.2.2) for j = H1, H2, Q1-Q4, or S1-S6.

9.2.5. Load R – L,LR 23

$(U) \rightarrow R_a$

The contents of U is transferred under j-field control to the arithmetic section and then to the R-register specified by the a-field (R_a).

1. If the CPU is in user mode, an attempt to Load RO causes a Guard Mode interrupt.

9.2.6. Load X Modifier – LXM 26

$(U) \rightarrow X_{a17-0}$;
X_{a35-18} unchanged

The contents of U is transferred under j-field control to the arithmetic section; the low-order 18 bits of the value in the arithmetic section is transferred to the lower half (bits 17–0) of the X-register specified by the a-field (X_a); the upper half (bits 35 through 18) of the X_a-register remains unchanged.

1. This instruction loads only the low-order 18 bits of the specified X_a-register, even if D7 = i = 1 to specify 24-bit indexing.

9.2.7. Load X – L,LX 27

$(U) \rightarrow X_a$

The contents of U is transferred under j-field control to the arithmetic section and then to the X-register specified by the a-field (X_a).

9.2.8. Load X Increment – LXI 46

$(U) \rightarrow (X_a)_{35-18}$;
$(X_a)_{17-0}$ unchanged

The contents of U is transferred under j-field control to the arithmetic section; the low-order 18 bits of the value in the arithmetic section is transferred to the upper half (bits 35–18) of the X-register specified by the a-field (X_a). The lower half (bits 17–0) of the X_a-register remains unchanged.

1. This instruction loads the full high-order 18 bits of the specified X_a-register even if D7 = i = 1 to specify 24-bit indexing.

9.2.9. Double Load A – DL f = 71, j = 13

$(U, U+1) \rightarrow A_a, A_{a+1}$

The contents of U and U+1 are transferred to the arithmetic section and then to A_a and A_{a+1}, respectively.

9.2.10. Double Load Negative A – DLN 71,14

-(U, U+1) $\rightarrow A_a, A_{a+1}$

The contents of U and U+1 are transferred to the arithmetic section where the 72-bit value is complemented and then transferred to A_a and A_{a+1}, respectively.

9.2.11. Double Load Magnitude A – DLM 71,15

$|(U, U+1)| \rightarrow A_a, A_{a+1}$

The contents of U and U+1 are transferred to the arithmetic section. If the sign bit (bit 35) of U is 1, the 72-bit value in the arithmetic section is complemented; if the sign bit is 0, the 72-bit value is not complemented. The final value (always positive) is transferred from the arithmetic section to A_a and A_{a+1}.

9.3. Store Instructions

The single-length store instructions transfer data from a control register specified by the a-field to the storage location or control register addressed by U. Exceptions to this are the Store Constant instructions. (See 9.3.5.)

Single-length data-word transfers to storage are controlled by the j-field. If j = 16_8 or 17_8, no data is stored. An interrupt will occur if:

- U < 200_8 and an Executive register is specified in user mode;
- U ≥ 200_8 and a storage limit violation occurs; or
- U ≥ 200_8 and a write-protection violation occurs.

Indexing, index incrementation/decrementation, and indirect addressing function normally in all cases.

9.3.1. Store A – S,SA 01

$(A_a) \rightarrow U$

The contents of A_a is transferred under j-field control to location U.

1. If j = 16_8 or 17_8, no data is stored.

9.3.2. Store Negative A – SN,SNA 02

$-(A_a) \rightarrow U$

The complement of the value of A_a is transferred under j-field control to location U.

1. If j = 16_8 or 17_8, no data is stored.

9.3.3. Store Magnitude A – SM,SMA 03

$|(A_a)| \rightarrow U$

If the sign bit (bit 35) of the value of A_a is 1, the value is complemented. The final value (always positive) is transferred under j-field control to location U.

1. If j = 16_8 or 17_8, no data is stored.

9.3.4. Store R – S,SR 04

$(R_a) \rightarrow U$

The contents of the R-register specified by the a-field is transferred under j-field control to location U.

1. If j = 16_8 or 17_8, no data is stored.

9.3.5. Store Constant Instructions – XX 05; a = 00–07

Constant $\rightarrow$ U

A constant value specified by the a-field is transferred under j-field control to location U. The following octal constant values may be stored:

SZ	a = 0	000000 000000	Zero
SNZ	a = 1	777777 777777	Ones
SP1	a = 2	000000 000001	Plus One
SN1	a = 3	777777 777776	Minus One
SFS	a = 4	050505 050505	Fieldata Blanks
SFZ	a = 5	606060 606060	Fieldata Zeros
SAS	a = 6	040040 040040	ASCII Blanks
SAZ	a = 7	060060 060060	ASCII Zeros

9.3.6. Store X – S,SX 06

$(X_a) \rightarrow U$

The contents of the X-register specified by the a-field is transferred under j-field control to location U.

1. If j = 16_8 or 17_8, no data is stored.

9.3.7. Double Store A – DS 71,12

$(A_a, A_{a+1}) \rightarrow U, U+1$

The contents of A_a and A_{a+1} are transferred to locations U and U+1, respectively.

9.3.8. Block Transfer - BT 22

$(X_x + u) \rightarrow X_a + u$, repeat k times;
k = the initial count in the repeat count register

A source word is transferred under j-field control to the arithmetic section, and then under j-field control to a destination word-location. The repeat count is decreased by 1. The source-to-destination transfer step is repetitively performed until the repeat count has been decreased to 0. The x-field specifies the X-register used with the u-field to determine the effective source word-address. The a-field specifies the X-register used in determining the effective destination word-address.

1. A word containing the desired repeat count in the rightmost 18-bit positions must be loaded in the repeat count register (R1) before performing the Block Transfer (BT) instruction.

2. If the initial repeat count is ±0, no data is transferred. If it is -0, then +0 is written into bits 17-0 of the repeat count.

3. If j = 16_8 or 17_8, no data is transferred; however, the repeat count is decreased to 0.

4. If the x-field is 0, no data is transferred. The contents of the X-register specified by the a-field remains unchanged, regardless of the contents of the a- and h-fields.

5. If an interrupt occurs before the repeat count has decreased to 0, the termination pass occurs at the conclusion of the currently active data transfer. The remnant repeat count is stored in R1. When the interrupt is honored, the captured P value is the address of the BT instruction or the address of the Execute instruction that led to the BT instruction. Thus, this address can be preserved and, when the interrupt has been processed, it is possible to return to the BT instruction and continue executing this instruction at the point where it was terminated for the interrupt. If the BT instruction was entered by means of an Execute instruction, the h-field of the Execute instruction must be 0 so that, when the program returns to the Execute instruction, the effective U address will again lead to the BT instruction. If the BT instruction specifies indirect addressing (i = 1), the h-field must be 0 to enable the program to return to the same effective U address and complete the BT instruction in the event of an interrupt.

6. If there is no indirect addressing (i = 0), the h-field is normally 1. If h = 0, no incrementation/decrementation of the index registers occurs. When h = 0, the source and destination addresses are the initial contents of the index registers used repetitively for every transfer performed. Thus, no more than one data transfer is effectively performed.

7. If the x-field is not 0, but the a-field is 0, the a-field references index register zero (X0), and proper operation occurs.

9.4. Fixed-Point Arithmetic Instructions

The fixed-point arithmetic instructions perform integer or fractional addition, subtraction, multiplication, and division. In a single-precision arithmetic instruction, the transfer of data from location U in storage to the arithmetic section is under control of the contents of the j-field of the instruction. For double-precision and parallel half-word and third-word arithmetic operations, the value in the j-field is a minor function code.

For all arithmetic instructions, indexing, index incrementation/decrementation, and indirect addressing function normally.

The overflow and carry designators are set according to the results of the operation for all add and add-negative instructions except add and add-negative halves and thirds.

The sign of the result is determined by the rules of algebra except for add and add-negative instructions where both operands are 0. In this case, the result is +0, except for add instructions where both operands are -0, and add-negative instructions where the minuend (A_a) is -0 and the subtrahend (U) is +0.

9.4.1. Add to A - A,AA 14

$(A_a) + (U) \rightarrow A_a$

The contents of U is transferred under j-field control to the arithmetic section. The 36-bit value in the arithmetic section is added algebraically to the contents of A_a. The sum is stored in A_a.

9.4.2. Add Negative to A - AN,ANA 15

$(A_a) - (U) \rightarrow A_a$

The contents of U is transferred under j-field control to the arithmetic section. The 36-bit value in the arithmetic section is subtracted algebraically from the contents of A_a. The difference is stored in A_a.

9.4.3. Add Magnitude to A - AM,AMA 16

$(A_a) + |(U)| \rightarrow A_a$

The contents of U is transferred under j-field control to the arithmetic section. If the sign bit (bit 35) of the 36-bit value in the arithmetic section is 1, the value is complemented; if the sign bit is 0, the value is not complemented. The final 36-bit value in the arithmetic section (always positive) is added algebraically to the contents of A_a. The sum is stored in A_a.

Only valid for j = 3-7, 17_8.

1. This instruction is the same as Add to A (see 9.4.1) for j = H1, H2, Q1-Q4, or S1-S6.

9.4.4. Add Negative Magnitude to A - ANM,ANMA 17

$(A_a) - |(U)| \rightarrow A_a$

The contents of U is transferred under j-field control to the arithmetic section. If the sign bit (bit 35) of the 36-bit value in the arithmetic section is 1, the value is complemented; if the sign bit is 0, the value is not complemented. The final 36-bit value in the arithmetic section (always positive) is subtracted algebraically from the contents of A_a. The difference is stored in A_a.

Only valid for j = 3-7, 17_8.

1. This instruction is the same as Add Negative to A (see 9.4.2) for j = H1, H2, Q1-Q4, or S1-S6.

9.4.5. Add Upper - AU 20

$(A_a) + (U) \rightarrow A_{a+1}$

The contents of U is transferred under j-field control to the arithmetic section. The 36-bit value in the arithmetic section is added algebraically to the contents of A_a. The sum is stored in A_{a+1}. The contents of U and A_a remain unchanged.

9.4.6. Add Negative Upper - ANU 21

$(A_a) - (U) \rightarrow A_{a+1}$

The contents of U is transferred under j-field control to the arithmetic section. The 36-bit value in the arithmetic section is subtracted algebraically from the contents of A_a. The difference is stored in A_{a+1}. The contents of U and A_a remain unchanged.

9.4.7. Add to X - A,AX 24

$(X_a) + (U) \rightarrow X_a$

The contents of U is transferred under j-field control to the arithmetic section. The 36-bit value in the arithmetic section is added algebraically to the contents of the X-register specified by the a-field. The sum is stored in the X-register specified by the a-field.

9.4.8. Add Negative to X - AN,ANX 25

$(X_a) - (U) \rightarrow X_a$

The contents of U is transferred under j-field control to the arithmetic section. The 36-bit value in the arithmetic section is subtracted algebraically from the contents of the X-register specified by the a-field. The difference is stored in the X-register specified by the a-field.

9.4.9. Multiply Integer - MI 30

$(A_a) * (U) \rightarrow A_a, A_{a+1}$

The contents of U is transferred under j-field control to the arithmetic section. The contents of A_a is multiplied algebraically by the 36-bit value in the arithmetic section, producing a 72-bit product. The most significant 36 bits of the product (including sign bits) are stored in A_a. The least significant 36 bits of the product are stored in A_{a+1}.

1. Bit positions 71 and 70 of the product are always sign bits. The product of any two 35-bit positive integers cannot exceed a 70-bit positive integer.

9.4.10. Multiply Single Integer - MSI 31

$(A_a) * (U) \rightarrow A_a$

The contents of U is transferred under j-field control to the arithmetic section. The contents of A_a is multiplied algebraically by the 36-bit value in the arithmetic section, producing a 72-bit product. The least significant 36 bits of the product are stored in A_a. The most significant 36 bits of the product are lost.

1. The 36-bit result stored in A_a does not represent the product as a signed number if the leftmost 37 bits of the 72-bit product formed in the arithmetic section are not identical.

9.4.11. Multiply Fractional - MF 32

$(A_a) * (U) \rightarrow A_a, A_{a+1}$

The contents of U is transferred under j-field control to the arithmetic section. The contents of A_a is multiplied algebraically by the 36-bit value in the arithmetic section, producing a 72-bit product that is shifted left circularly one bit position. The leftmost 36 bits of the shifted product, including the sign bit, are stored in A_a. The rightmost 36 bits are stored in A_{a+1}.

1. This instruction performs an operation identical to the Multiply Integer instruction (see 9.4.9), except that the 72-bit result of the multiplication process is shifted left circularly one bit position prior to storing it in A_a and A_{a+1}.

2. The rightmost bit of the result in A_{a+1} is a sign bit, which is identical to the leftmost bit of the result in A_a.

9.4.12. Divide Integer - DI 34

$(A_a, A_{a+1}) \div (U) \rightarrow A_a$;
remainder $\rightarrow A_{a+1}$

The contents of U is transferred under j-field control to the arithmetic section. The 72-bit signed number in A_a and A_{a+1} is divided algebraically by the 36-bit value in the arithmetic section. The 36-bit signed quotient is stored in A_a. The remainder retains the sign of the dividend (the leftmost bit of the initial contents of A_a) and is stored in A_{a+1}.

1. The absolute value of the 72-bit signed dividend (A_a, A_{a+1}) should be less than the absolute value of the divisor (j-determined portion of U) multiplied by 2^{35}. If this relationship is not satisfied and the Arithmetic Exception Interrupt designator (D20) is 0, A_a and A_{a+1} are cleared to 0 and the Divide Check designator (D23) is set to 1. If this relationship is not satisfied and D20 is 1, A_a and A_{a+1} remain unchanged, D23 is set to 1, and a Divide Check interrupt results. This includes the case when the divisor equals 0.

9.4.13. Divide Single Fractional - DSF 35

$(A_a) \div (U) \rightarrow A_{a+1}$

The contents of U is transferred under j-field control to the arithmetic section. The contents of A_a is divided algebraically by the 36-bit value in the arithmetic section. The 36-bit signed quotient is stored in A_{a+1}. The remainder is lost. The contents of A_a remain unchanged.

1. The absolute value of the dividend (A_a) should be less than the absolute value of the divisor (j-determined portion of U). If this relationship is not satisfied and the Arithmetic Exception Interrupt designator (D20) is 0, A_{a+1} is cleared to 0 and the Divide Check designator (D23) is set to 1. If this relationship is not satisfied and D20 is 1, A_{a+1} remains unchanged, D23 is set to 1, and a Divide Check interrupt results. This includes the case when the divisor equals 0.

2. This instruction performs an operation like that of Divide Integer instruction (see 9.4.12), except that the quotient appears to be shifted one bit to the right.

9.4.14. Divide Fractional - DF 36

$(A_a, A_{a+1}) \div (U) \rightarrow A_a;$
remainder $\rightarrow A_{a+1}$

The contents of U is transferred under j-field control to the arithmetic section. The 72-bit signed number in A_a and A_{a+1} is divided algebraically by the 36-bit value in the arithmetic section. The 36-bit signed quotient is stored in A_a. The remainder retains the sign of the dividend (the leftmost bit of the original contents of A_a) and is stored in A_{a+1}.

1. The absolute value of the leftmost half of the dividend (A_a) should be less than the absolute value of the divisor (j-determined portion of U). If this relationship is not satisfied and the Arithmetic Exception interrupt designator (D20) is 0, A_a and A_{a+1} are cleared to 0 and the Divide Check designator (D23) is set to 1. If this relationship is not satisfied and D20 is 1, A_a and A_{a+1} remain unchanged, D23 is set to 1, and a Divide Check interrupt results. This includes the case when the divisor equals 0.

2. This instruction performs an operation identical to Divide Integer instruction (see 9.4.12), except that the quotient appears to be shifted one bit to the right.

9.4.15. Double-Precision Fixed-Point Add - DA 71,10

$(A_a, A_{a+1}) + (U, U+1) \rightarrow A_a, A_{a+1}$

The 72-bit signed number from U and U+1 is added algebraically to the 72-bit signed number from A_a and A_{a+1}. The 72-bit sum is stored in A_a and A_{a+1}.

9.4.16. Double-Precision Fixed-Point Add Negative - DAN 71,11

$(A_a, A_{a+1}) - (U, U+1) \rightarrow A_a, A_{a+1}$

The 72-bit signed number from U and U+1 is subtracted algebraically from the 72-bit signed number from A_a and A_{a+1}. The 72-bit difference is stored in A_a and A_{a+1}.

9.4.17. Add Halves - AH 72,04

$(A_a)_{35-18} + (U)_{35-18} \rightarrow A_{a\ 35-18};$
$(A_a)_{17-0} + (U)_{17-0} \rightarrow A_{a\ 17-0}$

The contents of each half (18-bit portion) of U is added algebraically to the contents of the corresponding half of A_a. The sums are stored in the corresponding halves of A_a.

1. There is no interaction between the upper and lower halves of the operands. A carry from bit position 17 is propagated to bit 0, rather than bit 18. A carry from bit position 35 is propagated to bit 18, rather than bit 0.

9.4.18. Add Negative Halves - ANH 72,05

$(A_a)_{35-18} - (U)_{35-18} \rightarrow A_{a\ 35-18};$
$(A_a)_{17-0} - (U)_{17-0} \rightarrow A_{a\ 17-0}$

The contents of each half (18-bit portion) of U is subtracted algebraically from the contents of the corresponding half of A_a. The differences are stored in the corresponding halves of A_a.

1. There is no interaction between the upper and lower halves of the operands. A borrow from bit position 17 is propagated to bit 0, rather than bit 18. A borrow from bit position 35 is propagated to bit 18, rather than bit 0.

9.4.19. Add Thirds - AT 72,06

$(A_a)_{35-24} + (U)_{35-24} \rightarrow A_{a\ 35-24};$
$(A_a)_{23-12} + (U)_{23-12} \rightarrow A_{a\ 23-12};$
$(A_a)_{11-0} + (U)_{11-0} \rightarrow A_{a\ 11-0}$

The contents of each third (12-bit portion) of U is added algebraically to the contents of the corresponding third of A_a. The sums are stored in the corresponding thirds of A_a.

1. A carry from bit position 11, 23, or 35 is propagated to bit 0, 12, or 24, respectively, rather than to bit 12, 24, or 0.

9.4.20. Add Negative Thirds - ANT 72,07

$(A_a)_{35-24} - (U)_{35-24} \rightarrow A_{a\ 35-24};$
$(A_a)_{23-12} - (U)_{23-12} \rightarrow A_{a\ 23-12};$
$(A_a)_{11-0} - (U)_{11-0} \rightarrow A_{a\ 11-0}$

The contents of each third (12-bit portion) of U is subtracted algebraically from the contents of the corresponding third of A_a. The differences are stored in the corresponding thirds of A_a.

1. A borrow from bit position 11, 23, or 35 is propagated to bit 0, 12, or 24, respectively, rather than to bit 12, 24, or 0.

9.5. Floating-Point Arithmetic Instructions

Floating-point arithmetic operations allow for efficient computation involving numerical data with a wide range of magnitudes. In all floating-point arithmetic instructions, indexing, index incrementation/decrementation, and indirect addressing function normally.

The greatest precision is obtained in floating-point arithmetic operations when the floating-point input operands are normalized numbers. Certain floating-point operations produce undefined results if normalized input operands are not used. The supporting notes indicate which instructions are affected.

9.5.1. Floating Add - FA 76,00

$(A_a) + (U) \rightarrow A_a$;
RESIDUE $\rightarrow A_{a+1}$ if D17 = 1

The single-precision floating-point number from location U is added to the single-precision floating-point number from A_a. The resulting sum is normalized and then stored in single-precision floating-point format in A_a. If Floating-Point Residue Store Enable designator (D17) is 1, the residue in single-precision floating-point format is stored in A_{a+1}.

1. The result stored in A_a is a normalized number even if either or both of the input operands are not normalized. No attempt is made to normalize the residue stored in A_{a+1}.
2. A Floating-Point Characteristic Overflow/Underflow interrupt may occur.
3. If the mantissa of the most significant word of the result is ±0, the word stored depends on Floating-Point Zero Format Selection designator (D8).
4. The sign of the most significant word of the result is the sign of the large input operand. The sign of the other operand is assigned to the residue.

9.5.2. Floating Add Negative - FAN 76,01

$(A_a) - (U) \rightarrow A_a$;
RESIDUE $\rightarrow A_{a+1}$ if D17 = 1

The single-precision floating-point number from location U is subtracted from the single-precision floating-point number from A_a. The resulting difference is normalized and then stored in single-precision floating-point format in A_a. If the Floating-Point Residue Store Enable designator (D17) is 1, the residue in single-precision floating-point format is stored in A_{a+1}.

1. The result stored in A_a is a normalized number even if either or both of the input operands are not normalized. No attempt is made to normalize the residue stored in A_{a+1}.
2. A Floating-Point Characteristic Overflow/Underflow interrupt may occur.
3. If the mantissa of the most significant word of the result is ±0, the word stored depends on the Floating-Point Zero Format Selection designator (D8).
4. The Floating Add Negative operation is identical to the Floating Add operation described in 9.5.1, except that the ones complement of the contents of location U is used as the second operand.
5. The sign of the most significant word of the result is the sign of the large input operand. The sign of the other operand is assigned to the residue.

9.5.3. Double-Precision Floating Add - DFA 76,10

$(A_a, A_{a+1}) + (U, U+1) \rightarrow A_a, A_{a+1}$

The double-precision floating-point number from locations U and U+1 is added to the double-precision floating-point number from A_a and A_{a+1}. The resulting sum is normalized and then stored in double-precision floating-point format in A_a and A_{a+1}.

1. The result stored is a normalized number, even if either or both of the input operands are not normalized.

2. A Floating-Point Characteristic Overflow/Underflow interrupt may occur.

3. If the exponent value of the sum is less than -1024, and the Double-Precision Underflow designator (D5) and Arithmetic Exception Interrupt designator (D20) are 1, a Floating-Point Characteristic Underflow interrupt does not occur. Instead, +0 is stored in A_a and A_{a+1}. If D20 is 0, D5 is ignored.

4. If the mantissa produced is floating-point zero, the result stored is +0, regardless of the signs and characteristics of the input operands.

9.5.4. Double-Precision Floating Add Negative - DFAN 76,11

$(A_a, A_{a+1}) - (U, U+1) \rightarrow A_a, A_{a+1}$

The double-precision floating-point number from locations U and U+1 is subtracted from the double-precision floating-point number from A_a and A_{a+1}. The resulting difference is normalized and then stored in double-precision floating-point format in A_a and A_{a+1}.

1. The result stored is a normalized number, even if either or both of the input operands are not normalized.

2. A Floating-Point Characteristic Overflow/Underflow interrupt may occur.

3. If the exponent value of the sum is less than -1024, and the Double-Precision Underflow designator (D5) and Arithmetic Exception Interrupt designator (D20) are 1, a Floating-Point Characteristic Underflow interrupt does not occur. Instead, +0 is stored in A_a and A_{a+1}. If D20 is 0, D5 is ignored.

4. If the mantissa produced is floating-point zero, the result stored is +0, regardless of the signs and characteristics of the input operands.

9.5.5. Floating Multiply - FM 76,02

$(A_a) * (U) \rightarrow A_a$ (and A_{a+1} if D17 = 1)

The single-precision floating-point number from A_a is multiplied by the single-precision floating-point number from location U. The resulting double-length product is packed into two single-precision floating-point numbers. The most significant portion of the product in single-precision floating-point format is stored in A_a. If the Floating-Point Residue Store Enable designator (D17) is 1, the least significant portion of the product in single-precision floating-point format is stored in A_{a+1}.

1. If either or both input operands are not normalized numbers, the results are undefined.

The following notes apply only if both input operands are normalized numbers.

2. A Floating-Point Characteristic Overflow/Underflow interrupt may occur.

3. The portion of the product stored in A_a is a normalized number. No attempt is made to normalize the number stored in A_{a+1}.

4. The algebraic rule for signs applies to the portions of the product stored in A_a and A_{a+1}.

5. If the mantissa of either or both input operands is 0, the following applies:

   a. A Floating-Point Characteristic Overflow/Underflow interrupt never occurs, regardless of the values of the characteristics of the input operands.

   b. If the Floating-Point Zero Format Selection designator (D8) is 0, the result stored in A_a is +0 regardless of the signs of the input operands.

   c. If D8 is 1 and if the exponent value is in the range -128 through +127, the most significant product-word will reflect the magnitude of the characteristic produced and the sign produced by the mantissa arithmetic.

   d. If the exponent value of the most significant product-word is greater than +127 or less than -128, the result stored in A_a is ±0, depending on the signs of the input operands.

6. The value of D8 has no effect on the least significant product-word. When the mantissa for the least significant product-word is 0, it is packed with the appropriate characteristic. If the characteristic of the residue is less than -128, the result stored in A_{a+1} is ±0, depending on the signs of the operands.

   A characteristic overflow of the most significant word can occur; however, the characteristic of the residue could be in the range 000 through 377. In this case, the result stored in A_a is ±0 depending on the algebraic rule of the sign, and the residue is packed with the appropriate characteristic and stored in A_{a+1}.

7. If the characteristic of the number stored in A_a is greater than or equal to 27, the characteristic of the number stored in A_{a+1} is 27 less than the characteristic in A_a.

9.5.6. Double-Precision Floating Multiply - DFM 76,12

$(A_a, A_{a+1}) * (U, U+1) \rightarrow A_a, A_{a+1}$

The double-precision floating-point number from A_a and A_{a+1} is multiplied by the double-precision floating-point number from locations U and U+1. The product is normalized and stored in double-precision floating-point format in A_a and A_{a+1}.

1. If either or both input operands are not normalized numbers, the results are undefined.

The following notes apply only if both operands are normalized numbers.

2. A Floating-Point Characteristic Overflow/Underflow interrupt may occur.

3. The result stored in A_a and A_{a+1} is always a normalized number.

4. The algebraic rule for signs applies, except for the special cases covered in notes 5b and 6.

5. If the mantissa of either or both input operands is 0, the following applies:

   a. A Floating-Point Characteristic Overflow/Underflow interrupt never occurs, regardless of the values of the characteristics of the input operands.

b. The result stored in A_a and A_{a+1} is +0, regardless of the signs of the input operands.

6. If the exponent value of the product is less than -1024, and the Double-Precision Underflow designator (D5) and Arithmetic Exception Interrupt designator (D20) are 1, a Floating-Point Characteristic Underflow interrupt does not occur. Instead +0, regardless of the signs of the input operands, is stored in A_a and A_{a+1}. If D20 is 1 and D5 is 0, the interrupt occurs. If D20 is 0, D5 is ignored.

9.5.7. Floating Divide - FD 76,03

$(A_a) \div (U) \rightarrow A_a$;
REMAINDER $\rightarrow A_{a+1}$ if D17 = 1

The single-precision floating-point number from A_a is divided by the single-precision floating-point number from location U. The quotient is stored in A_a in single-precision floating-point format. If the Floating-Point Residue Store Enable designator (D17) is 1, the remainder is stored in A_{a+1} in single-precision floating-point format.

1. If either or both input operands are not normalized numbers, the results are not defined.

The following notes apply only if both operands are normalized numbers.

2. A Floating-Point Characteristic Overflow/Underflow interrupt may occur.

3. If the mantissa of the divisor (U) is 0, a Divide Check interrupt occurs.

4. The quotient stored in A_a is a normalized number. No attempt is made to normalize the remainder that is stored in A_{a+1} when D17 is 1.

5. The algebraic rule for signs applies to the quotient stored in A_a. The sign of the dividend is assigned to the remainder stored in A_{a+1}.

6. If the mantissa, but not the divisor (U), of the dividend (A_a), is 0 the following applies:

   a. A Floating-Point Characteristic Overflow/Underflow interrupt never occurs, regardless of the characteristics of the operands.

   b. If the Floating-Point Zero Format Selection designator (D8) is 0, the quotient stored in A_a is +0, regardless of the signs of the operands.

   c. If D8 is 1 and the exponent value of the quotient is greater than +128 or less than -128, the quotient stored in A_a is ±0, depending on the signs of the input operands.

7. If the exponent value of the remainder is less than -128, the remainder stored in A_{a+1} is ±0, depending on the sign of the dividend from A_a.

8. If the characteristic of the dividend from A_a is greater than or equal to 27, then the characteristic of the number stored in A_{a+1} for the remainder is 27 or 26 less than the characteristic of the dividend.

9.5.8. Double-Precision Floating Divide - DFD 76,13

$(A_a, A_{a+1}) \div (U, U+1) \rightarrow A_a, A_{a+1}$

The double-precision floating-point number from A_a and A_{a+1} is divided by the double-precision floating-point number from locations U and U+1. The quotient is stored in A_a and A_{a+1} in double-precision floating-point format. The remainder is not retained.

1. If either or both of the input operands are not normalized numbers, the results are undefined.

The following notes apply only if both operands are normalized numbers.

2. A Floating-Point Characteristic Overflow/Underflow interrupt may occur,
3. If the mantissa of the divisor is 0, a Divide Check interrupt occurs.
4. The result stored in A_a and A_{a+1} is always a normalized number.
5. The algebraic rule for signs applies, except for the special cases explained in notes 6b and 7.
6. If the dividend mantissa (A_a, A_{a+1}) is 0 and the divisor mantissa (U, U+1) is not 0, the following applies:
   a. A Floating-Point Characteristic Overflow/Underflow interrupt never occurs, regardless of the values of the characteristics of the input operands.
   b. The result stored in A_a and A_{a+1} is +0 regardless of the signs of the operands.
7. If the exponent value of the quotient is less than -1024, and the Double-Precision Underflow designator (D5) and Arithmetic Exception Interrupt designator (D20) are 1, a Floating-Point Characteristic Underflow interrupt does not occur. Instead +0, regardless of the of the input operands, is stored in A_a and A_{a+1}. If D20 is 1 and D5 is 0 the interrupt occurs. If D20 is 0, D5 is ignored.

9.5.9. Load and Unpack Floating - LUF 76,04

$|(U)|_{34-27} \rightarrow A_{a\ 7-0}$, zero fill;
$(U)_{26-0} \rightarrow A_{a+1 \text{ bits } 26-0}$, sign fill

The single-precision floating-point number from location U is transferred to the arithmetic section and unpacked. The absolute value of the biased characteristic of the input operand is transferred to bits 7 through 0 of A_a; bits 35 through 8 of the A_a are filled with 0's. The mantissa of the input operand is transferred to bits 26 through 0 of A_{a+1}; bits 35 through 27 of A_{a+1} are filled with bits identical to the sign of the floating-point number in U.

1. No attempt is made to normalize the operand.

9.5.10. Double Load and Unpack Floating - DFU 76,14

$|(U, U+1)_{70-60}| \rightarrow A_{a\ 10-0}$, zero fill;
$(U, U+1)_{59-36} \rightarrow A_{a+1\ \text{bits}\ 23-0}$, sign fill;
$(U, U+1)_{35-0} \rightarrow A_{a+2}$

The double-precision floating-point number from locations U and U+1 is transferred to the arithmetic section and unpacked. The absolute value of the biased characteristic of the input operand is transferred to bits 10 through 0 of A_a; bits 35 through 11 of A_a are filled with 0's. The leftmost 24 bits of the mantissa, $(U)_{23-0}$, are transferred to bits 23 through 0 of A_{a+1}; bits 35 through 24 of A_{a+1} are filled with bits identical to the sign of the floating-point number in locations U and U+1. The rightmost 36 bits of the mantissa (U+1) are transferred to A_{a+2}.

1. No attempt is made to normalize the operand.

9.5.11. Load and Convert to Floating - LCF 76,05

$(U)_{35} \rightarrow A_{a+1\ \text{bit}\ 35}$;
$[\text{NORMALIZED}\ (U)]_{26-0} \rightarrow A_{a+1\ \text{bits}\ 26-0}$;
if $(U)_{35} = 0$, $(A_a)_{7-0} \pm$ NORMALIZING COUNT $\rightarrow A_{a+1\ \text{bits}\ 34-27}$;
if $(U)_{35} = 1$, ones complement of $[(A_a)_{7-0} \pm$ NORMALIZING COUNT$] \rightarrow A_{a+1\ \text{bits}\ 34-27}$

The fixed-point number from location U is sent to the arithmetic section where it is shifted right or left, as required, to normalize it. The normalizing shift count is added to the characteristic from the rightmost eight bits of A_a if a normalizing right-shift is required. It is subtracted from the characteristic if a normalizing left-shift is required. The adjusted characteristic (complemented if U is negative) is packed with the normalized value from U to form a single-precision floating-point number. The packed result is stored in A_{a+1}. The contents of A_a remains unchanged.

1. A Floating-Point Characteristic Overflow/Underflow interrupt may occur.

2. The 28 leftmost bits from A_a are ignored; $(A_a)_{7-0}$ must be prebiased.

3. If the resultant mantissa is 0, the following applies:

   a. If the Floating-Point Zero Format Selection designator (D8) is 0, the result stored in A_a is +0.

   b. If D8 is 1, and the resultant characteristic is in the range 000 through 377, the characteristic is packed with the 0 mantissa and stored in A_a.

   c. If D8 is 1, and the resultant characteristic is a negative number, ±0 is stored in A_a, depending on the sign of the input operand.

9.5.12. Double Load and Convert to Floating - DFP, DLCF 76,15

$(U)_{35} \rightarrow A_{a+1\ \text{bit}\ 35}$;
$[\text{NORMALIZED}\ (U, U+1)]_{59-0} \rightarrow A_{a+1\ \text{bits}\ 23-0}$ and A_{a+2}
if $(U)_{35} = 0$, $(A_a)_{10-0} \pm$ NORMALIZING COUNT $\rightarrow A_{a+1\ \text{bits}\ 34-24}$;
if $(U)_{35} = 1$, ones complement of $[(A_a)_{10-0} \pm$ NORMALIZING COUNT$] \rightarrow A_{a+1\ \text{bits}\ 34-24}$

The double-precision fixed-point number from locations U and U+1 is sent to the arithmetic section where it is shifted right or left, if necessary, to normalize it. The normalizing shift count is added to the characteristic from the rightmost 11 bits of A_a if a normalizing right-shift is required. It is subtracted from the characteristic if a normalizing left-shift is required. The adjusted characteristic (complemented if U is negative) is packed with the normalized value from U and U+1 to form a double-precision floating-point number, and the packed result is stored in A_{a+1} and A_{a+2}. The contents of A_a remains unchanged.

1. A Floating-Point Characteristic Overflow/Underflow interrupt may occur.

2. The 25 leftmost bits from A_a are ignored; $(A_a)_{10-0}$ must be prebiased.

3. If the 72-bit input operand from U and U+1 is ±0, the result stored is +0, regardless of the sign of the 72-bit operand.

4. If the adjusted characteristic represents a negative number when the Arithmetic Exception Interrupt designator (D20) and the Double-Precision Underflow designator (D5) are 1, a Floating-Point Characteristic Underflow interrupt does not occur. Instead +0, regardless of the sign of the 72-bit operand, is stored.

9.5.13. Floating Expand and Load - FEL 76,16

If $(U)_{35} = 0$, $(U)_{35-27} + 01600 \rightarrow A_{a\ 35-24}$;
if $(U)_{35} = 1$, $(U)_{35-27} - 01600 \rightarrow A_{a\ 35-24}$;
$(U)_{26-3} \rightarrow A_{a\ 23-0}$;
$(U)_{2-0} \rightarrow A_{a+1 \text{ bits } 35-33}$;
$(U)_{35} \rightarrow A_{a+1 \text{ bits } 32-0}$

The single-precision floating-point input operand from location U is transferred to the arithmetic section. The three fields of the operand are expanded to form a double-precision floating-point number as follows:

- The sign bit is stored in bits 71 and 32 through 0.
- The 8-bit characteristic that includes a bias of 0200 is modified to an 11-bit characteristic that includes a bias of 02000; it is stored in bits 70 through 60.
- The 27-bit mantissa is stored in bits 59 through 33.

The result is transferred to A_a and A_{a+1}.

1. If the operand is not in the normalized single-precision floating-point format, the result stored is undefined.

The following notes apply only if the input operand is a normalized number.

2. If the mantissa of the input operand is ±0, the result stored in A_a and A_{a+1} is +0, regardless of the sign of the operand.

3. A Floating-Point Characteristic Overflow/Underflow interrupt will not occur as a result of this instruction.

9.5.14. Floating Compress and Load - FCL 76,17

If $(U)_{35} = 0$, $(U)_{35-24} - 01600 \rightarrow A_{a\ 35-27}$;
if $(U)_{35} = 1$, $(U)_{35-24} + 01600 \rightarrow A_{a\ 35-27}$;
$(U)_{23-0} \rightarrow A_{a\ 26-3}$;
$(U+1)_{35-33} \rightarrow A_{a\ 2-0}$

The double-precision floating-point operand from locations U and U+1 is transferred to the arithmetic section. The three fields of the operand are compressed to form a single-precision floating-point number as follows:

- The sign bit is stored in bit 35.
- The 11-bit characteristic that includes a bias of 02000 is modified to an 8-bit characteristic that includes a bias of 0200; it is stored in bits 34 through 27.
- The 27 leftmost bits of the mantissa (bits 23 through 0 from location U and bits 35 through 33 from location U+1) are stored in bits 26 through 0.

The result is transferred to A_a.

The following notes apply only if the operand is a normalized number.

1. If Arithmetic Exception Interrupt designator (D20) = 1, a Floating-Point Characteristic Overflow interrupt occurs if the characteristic of the operand is greater than +127, and a Floating-Point Characteristic Underflow interrupt occurs if the characteristic of the operand is less than -128. The Characteristic Underflow designator (D21) is set when an underflow condition is detected and the Characteristic Overflow designator (D22) is set when an overflow condition is detected.
2. The contents of $U+1_{32-0}$ is ignored.
3. If the operand is not a normalized number or is equal to ±0, the result stored in A_a is +0, regardless of the sign of the input operand.

9.5.15. Magnitude of Characteristic Difference to Upper - MCDU 76,06

$\left| \, |(A_a)|_{35-27} - |(U)|_{35-27} \right| \rightarrow A_{a+1 \text{ bits } 8-0}$;
zeros $\rightarrow A_{a+1 \text{ bits } 35-9}$

The absolute value of the characteristic of the single-precision floating-point number from location U is subtracted from the absolute value of the characteristic of the single-precision floating-point number from A_a.

The absolute value of the 9-bit difference is stored in bits 8 through 0 of A_{a+1}. Bits 35 through 9 of A_{a+1} are zero filled. The contents of A_a is not changed.

1. The mantissas from locations U and A_a are ignored.

9.5.16. Characteristic Difference to Upper - CDU 76,07

$|(A_a)|_{35-27} - |(U)|_{35-27} \rightarrow A_{a+1}$ bits 8-0;
sign bits to A_{a+1} bits 35-9

The absolute value of the characteristic of the single-precision floating-point number from location U is subtracted from the absolute value of the characteristic of the single-precision floating-point number from A_a. The 9-bit signed difference is stored in bits 8 through 0 of A_{a+1}. Bits 35 through 9 of A_{a+1} are filled with bits identical to the sign of the difference. The contents of A_a is not changed.

1. The mantissas from location U and from A_a are ignored.

9.6. Search and Masked-Search Instructions

There are six search instructions; each compares the contents of either one or two A-registers with the contents of storage locations or control registers.

There are eight masked-search instructions, each compares contents of predefined bit positions of either one or two A-registers with the contents of the corresponding bit positions of storage locations or control registers.

Both search and masked-search instructions are multistage instructions. The various stages required to perform these instructions are as follows:

- An initial stage
- Repeated test stages (any number from 0 to 262,143)
- Termination stage

If indirect addressing is specified, it proceeds prior to initiation of the first test stage.

The initial stage prepares the control section and the arithmetic section for the test stages. The following steps are performed during the initial stage:

- The program address register (P-register) is incremented: $(P) + 1 \rightarrow P$
- The contents of the repeat count register (R1) is transferred to the index subsection.
- The contents of the specified A-registers are transferred to the arithmetic section.
- The contents of the masked register (R2) is transferred to the arithmetic section for a masked-search instruction.

These steps are performed only during the initial stage and are not repeated during the test stages.

The rightmost 18 bit positions of R1 contain the repeat count; that is, the maximum number of test stages to be performed. R1 must be loaded with the desired repeat count prior to initiating a search or masked search instruction. If the initial repeat-count is +0, the termination stage is initiated immediately following the completion of the initial stage; there are no test stages. If the repeat count is not 0, a series of one or more test stages is initiated.

During each test stage, the value U is formed in the index subsection. For the search instructions, an input operand is transferred to the arithmetic section under j-field control. The inputs to the test process are the values obtained using the effective U address and the A-register or registers specified by the instruction.

For the masked-search instructions, the contents of the j-field is a minor function code. The inputs to the test process are:

- the logical product of the masked from R2 and the input operand addressed by U
- the logical products of the masked and the specified A-registers

Each bit of the logical product is the logical product of the contents of corresponding bit positions of the two words. The logical product of two bits gives the same results as the Logical $\boxed{\text{AND}}$.

The search and masked-search instructions include algebraic and alphanumeric comparisons. During an algebraic comparison, the leftmost bit of each of the 36-bit values is considered to be a sign bit; a positive number is always recognized as being greater than a negative number. During an alphanumeric comparison, the leftmost bit of each of the 36-bit values is considered to be a numeric bit rather than a sign bit.

If the test process shows that the specified conditions are met, the repeat count is decreased by 1 and the termination stage is initiated. If the specified conditions are not met, the repeat count is decreased by 1 and examined. If the decreased repeat count is 0, the termination stage is initiated. If the decreased repeat count is not 0, another test stage is normally initiated. It should be noted that if $x = 0$, $X_i = 0$, or $h = 0$, the same value for U will be formed in each test stage.

The termination stage is initiated if the initial repeat count is 0, if the repeat count is decreased from 1 to 0 during the test stage, or if the conditions specified by the search or masked-search instruction are detected during a test stage. If an interrupt is detected during either an initial stage or one of the test stages, the termination stage is initiated immediately following that stage. The P-register is reset and the repeat count is stored so that the search instruction can be resumed when the interrupt condition has been satisfied.

The termination stage is used to transfer the current repeat count from the index subsection to the rightmost 18-bit positions of R1. The contents of the P-register may or may not be changed during the termination stage, as follows:

- If the termination stage is entered as a result of detecting that the initial repeat count is 0 during the initial stage or that the decreased repeat count is 0 during a test stage for which the specified conditions are not detected (no find), the contents of the P-register is not changed during the termination stage. The P-register contains the address of the instruction following the search or masked-search instruction, or the address of the instruction following the Execute instruction that referenced the search or masked-search instruction (next instruction condition).

- If the termination stage is entered as a result of detecting the specified conditions during a test stage (a find has been made), the P-register is incremented during the termination stage: $(P) + 1 \rightarrow P$. Since the P-register was also incremented during the initial stage, it now contains the address of the search or masked-search instruction (or the address of the Execute instruction that referenced it) + 2 (skip next instruction condition).

- If the termination stage is entered as a result of recognizing an interrupt, the P-register is decreased by 1 during the termination stage: $(P) - 1 \rightarrow P$. This decrease offsets the incrementation of the P-register performed during the initial stage; the P-register now contains the address of the search or masked-search instruction, or the Execute instruction that referenced it. This address can be preserved so that, when the interrupt condition has been satisfied, the search or masked-search can be resumed at the point where it was terminated for the interrupt. If the search or masked-search instruction is entered by means of an Execute instruction, the h-field of the Execute instruction should be 0 (no incrementation) so that when the program returns to the Execute instruction after an interrupt, the effective U address will again lead to the search or masked-search instruction.

If the search or masked-search instruction specifies indirect addressing (i-field is 1), the h-field should be 0, to enable the program to return to the same effective U address and resume the search or masked search after an interrupt.

For equality searches (SE, SNE, MSE, MSNE), +0 does not equal -0; for arithmetic searches (SLE, SG, SW, SNW, MSLE, MSE, MSW, MSNW), +0 is greater than -0; for alphanumeric searches (MASL, MASG), -0 is greater than +0.

When a search or masked-search is resumed after an interrupt, the initial stage is again performed to prepare the control section for the remaining test stages and to transfer the contents of the specified A-register to the arithmetic section for the comparisons performed in the test stages. When h = 1 (that is, index register incrementation is specified), if the a- and x-fields reference the same control register, the contents of that register will have been altered by the index incrementation that occurred before the search or masked search was interrupted. As a result, when the search or masked search is resumed, the value referenced by the a-field to be used in the test stages is no longer the original test value used before the interrupt occurred. Therefore, when h = 1, the a-field and x-field should not specify the same control register so that the search or masked-search instruction can be resumed in the event of an interrupt.

9.6.1. Search Equal - SE 62

Skips NI if (U) = (A_a), else repeat

During the initial stage, the contents of the repeat count register (R1) is transferred to the index subsection, the contents of A_a is transferred to the arithmetic section, and the P-register is incremented.

If the initial repeat count is 0, the Next Instruction (NI) is initiated.

If the initial repeat count is not 0, the first test stage is initiated. During each test stage, the repeat count is decreased and the contents of U is transferred to the arithmetic section under j-field control. This value from U is compared with the value from A_a and:

- If (U) = (A_a), the termination stage is initiated. This stage stores the remnant repeat count and increments the P-register. (Skip to NI.)
- If (U) ≠ (A_a) and the repeat count is not 0, another test stage is initiated.
- If (U) ≠ (A_a) and the repeat count is 0, the termination stage stores 0 as the remnant repeat count and the P-register is not incremented.

1. +0 is not equal to -0.

9.6.2. Search Not Equal - SNE 63

Skips NI if (U) $\neq$ (A_a), else repeat

During the initial stage, the contents of the repeat count register (R1) is transferred to the index subsection, the contents of A_a is transferred to the arithmetic section, and the P-register is incremented.

If the initial repeat count is 0, the Next Instruction (NI) is initiated.

If the initial repeat count is not 0, the first test stage is initiated. During each test stage, the repeat count is decreased and the contents of U is transferred to the arithmetic section under j-field control. The value from U is compared with the value from A_a and:

- If (U) $\neq$ (A_a), the termination stage is initiated. The termination stage stores the remnant repeat count and increments the P-register. (Skip NI.)
- If (U) = (A_a) and the repeat count is not 0, another test stage is initiated.
- If (U) = (A_a) and the repeat count is 0, the termination stage is initiated. The termination stage stores 0 as the remnant repeat count and the P-register is not incremented.

1. +0 is not equal to -0.

9.6.3. Search Less Than or Equal/Search Not Greater - SLE,SNG 64

Skips NI if (U) $\leq$ (A_a), else repeat

During the initial stage, the contents of the repeat count register (R1) is transferred to the index subsection, the contents of A_a is transferred to the arithmetic section, and the P-register is incremented.

If the initial repeat count is 0, the NI is initiated.

If the initial repeat count is not 0, the first test stage is initiated. During each test stage, the repeat count is decreased and the contents of U is transferred to the arithmetic section under j-field control. The value from U is compared with the value from A_a and:

- If (U) $\leq$ (A_a), the termination stage is initiated. The termination stage stores the remnant repeat count and increments the P-register. (Skip NI.)
- If (U) > (A_a) and the repeat count is not 0, another test stage is initiated.
- If (U) > (A_a) and the repeat count is 0, the termination stage is initiated. The termination stage stores 0 as the remnant repeat count and the P-register is not incremented.

1. +0 is greater than -0.

9.6.4. Search Greater - SG 65

Skips NI if (U) > (A_a), else repeat

During the initial stage, the contents of the repeat count register (R1) is transferred to the index subsection, the contents of A_a is transferred to the arithmetic section, and the P-register is incremented.

If the initial repeat count is 0, the Next Instruction (NI) is initiated.

If the initial repeat count is not 0, the first test stage is initiated. During each test stage, the repeat count is decreased and the contents of U is transferred to the arithmetic section under j-field control. The value from U is compared with the value from A_a and:

- If $(U) > (A_a)$, the termination stage is initiated. The termination stage stores the remnant repeat count and increments the P-register. (Skip NI.)
- If $(U) \leq (A_a)$ and the repeat count is not 0, another test stage is initiated.
- If $(U) \leq (A_a)$ and the repeat count is 0, the termination stage is initiated. The termination stage stores 0 as the remnant repeat count and the P-register is not incremented.

1. +0 is greater than -0.

9.6.5. Search Within Range – SW 66

Skips NI if $(A_a) < (U) \leq (A_{a+1})$, else repeat

During the initial stage, the contents of the repeat count register (R1) is transferred to the index subsection, the contents of A_a and A_{a+1} are transferred to the arithmetic section, and the P-register is incremented.

If the initial repeat count is 0, the Next Instruction (NI) is initiated.

If the initial repeat count is not 0, the first test stage is initiated. During each test stage, the repeat count is decreased and the contents of U is transferred to the arithmetic section under j-field control. The value from U is compared with the value from A_a and:

- If $(U) > (A_a)$ and $(U) \leq (A_{a+1})$, the termination stage is initiated. The termination stage stores the remnant repeat count and increments the P-register. (Skip NI.)
- If $(U) \leq (A_a)$ or $(U) > (A_{a+1})$, and the repeat count is not 0, another test stage is initiated.
- If $(U) \leq (A_a)$ or $(U) > (A_{a+1})$, and the repeat count is 0, the termination stage is initiated. The termination stage stores 0 as the remnant repeat count, and the P-register is not incremented.

1. +0 is greater than -0.
2. Normally, $(A_a) < (A_{a+1})$. However, if $(A_a) \geq (A_{a+1})$, there is no value from U that can satisfy the conditions $(A_a) < (U) \leq (A_{a+1})$.

9.6.6. Search Not Within Range – SNW 67

Skips NI if $(U) \leq (A_a)$ or $(U) > (A_{a+1})$, else repeat

During the initial stage, the contents of the repeat count register (R1) is transferred to the index subsection, the contents of A_a and A_{a+1} are transferred to the arithmetic section, and the P-register is incremented.

If the initial repeat count is 0, the Next Instruction (NI) is initiated.

If the initial repeat count is not 0, the first test stage is initiated. During each test stage, the repeat count is decreased and the contents of U is transferred to the arithmetic section under j-field control. The value from U is compared with the value from A_a and:

- If $(U) \leq (A_a)$ or $(U) > (A_{a+1})$, the termination stage is initiated. The termination stage stores the remnant repeat count and increments the P-register. (Skip NI.)
- If $(U) > (A_a)$ and $(U) \leq (A_{a+1})$, and the repeat count is not 0, another test stage is initiated.
- If $(U) > (A_a)$ and $(U) \leq (A_{a+1})$, and the repeat count is 0, the termination stage is initiated. The termination stage stores 0 as the remnant repeat count, and the P-register is not incremented.

1. Normally, $(A_a) < (A_{a+1})$. If, however, $(A_a) \geq (A_{a+1})$, there is no value from U that will not satisfy the conditions $(U) \leq (A_a)$ or $(U) > (A_{a+1})$.
2. +0 is greater than -0.

9.6.7. Masked Search Equal – MSE 71,00

Skips NI if $(U) \boxed{\text{AND}} (R2) = (A_a) \boxed{\text{AND}} (R2)$, else repeat

During the initial stage, the contents of the repeat count register (R1) is transferred to the index subsection, the contents of A_a and R2 are transferred to the arithmetic section, the logical product of the values from A_a and R2 is formed, and the P-register is incremented.

If the initial repeat count is 0, the Next Instruction (NI) is initiated.

If the initial repeat count is not 0, the first test stage is initiated. During each test stage, the repeat count is decreased and the contents of U is transferred to the arithmetic section. $(U) \boxed{\text{AND}} (R2)$ is compared to $(A_a) \boxed{\text{AND}} (R2)$ and:

- If $(U) \boxed{\text{AND}} (R2) = (A_a) \boxed{\text{AND}} (R2)$, the termination stage is initiated. This stage stores the remnant repeat count and increments the P-register (skip NI).
- If $(U) \boxed{\text{AND}} (R2) \neq (A_a) \boxed{\text{AND}} (R2)$ and the repeat count is not 0, another test stage is initiated.
- If $(U) \boxed{\text{AND}} (R2) \neq (A_a) \boxed{\text{AND}} (R2)$ and the repeat count is 0, the termination stage stores 0 as the remnant repeat count, and the P-register is not incremented.

1. +0 is not equal to -0.

9.6.8. Masked Search Not Equal – MSNE 71,01

Skips NI if $(U) \boxed{\text{AND}} (R2) \neq (A_a) \boxed{\text{AND}} (R2)$, else repeat.

During the initial stage, the contents of the repeat count register (R1) is transferred to the index subsection, the contents of A_a and R2 are transferred to the arithmetic section, the logical product of the values from A_a and R2 is formed, and the P-register is incremented.

If the initial repeat count is 0, the Next Instruction (NI) is initiated.

If the initial repeat count is not 0, the first test stage is initiated. During each test stage, the repeat count is decreased and the contents of U is transferred to the arithmetic section. $(U) \boxed{\text{AND}} (R2)$ is compared to $(A_a) \boxed{\text{AND}} (R2)$ and:

- If $(U) \boxed{\text{AND}} (R2) \neq (A_a) \boxed{\text{AND}} (R2)$, the termination stage is initiated. This stage stores the remnant repeat count and increments the P-register (skip NI).
- If $(U) \boxed{\text{AND}} (R2) = (A_a) \boxed{\text{AND}} (R2)$ and the repeat count is not 0, another test stage is initiated.
- If $(U) \boxed{\text{AND}} (R2) = (A_a) \boxed{\text{AND}} (R2)$ and the repeat count is 0, the termination stage stores 0 as the remnant repeat count and the P-register is not incremented.

1. +0 is not equal to -0.

9.6.9. Masked Search Less Than or Equal/Not Greater - MSLE,MSNG 71,02

Skips NI if $(U) \boxed{\text{AND}} (R2) \leq (A_a) \boxed{\text{AND}} (R2)$, else repeat

During the initial stage, the contents of the repeat count register (Rl) is transferred to the index subsection, the contents of A_a and R2 are transferred to the arithmetic section, the logical product of the values from A_a and R2 is formed, and the P-register is incremented.

If the initial repeat count is 0, the Next Instruction (NI) is initiated.

If the initial repeat count is not 0, the first test stage is initiated. During each test stage, the repeat count is decreased and the contents of U is transferred to the arithmetic section. $(U) \boxed{\text{AND}} (R2)$ is compared to $(A_a) \boxed{\text{AND}} (R2)$ and:

- If $(U) \boxed{\text{AND}} (R2) \leq (A_a) \boxed{\text{AND}} (R2)$, the termination stage is initiated. This stage stores the remnant repeat count and increments the P-register (skip NI).
- If $(U) \boxed{\text{AND}} (R2) > (A_a) \boxed{\text{AND}} (R2)$ and the repeat count is not 0, another test stage is initiated.
- If $(U) \boxed{\text{AND}} (R2) > (A_a) \boxed{\text{AND}} (R2)$ and the repeat count is 0, the termination stage stores 0 as the remnant repeat count, and the P-register is not incremented.

1. +0 is greater than -0.

9.6.10. Masked Search Greater - MSG 71,03

Skips NI if $(U) \boxed{\text{AND}} (R2) > (A_a) \boxed{\text{AND}} (R2)$, else repeat

During the initial stage, the contents of the repeat count register (Rl) is transferred to the index subsection, the contents of A_a and R2 are transferred to the arithmetic section, the logical product of the values from A_a and R2 is formed, and the P-register is incremented.

If the initial repeat count is 0, the Next Instruction (NI) is initiated.

If the initial repeat count is not 0, the first test stage is initiated. During each test stage, the repeat count is decreased and the contents of U is transferred to the arithmetic section. $(U)\ \boxed{\text{AND}}\ (R2)$ is compared to $(A_a)\ \boxed{\text{AND}}\ (R2)$ and:

- If $(U)\ \boxed{\text{AND}}\ (R2) > (A_a)\ \boxed{\text{AND}}\ (R2)$, the termination stage is initiated. This stage stores the remnant repeat count and increments the P-register. (Skip NI.)

- If $(U)\ \boxed{\text{AND}}\ (R2) \leq (A_a)\ \boxed{\text{AND}}\ (R2)$ and the repeat count is not 0, another test stage is initiated.

- If $(U)\ \boxed{\text{AND}}\ (R2) \leq (A_a)\ \boxed{\text{AND}}\ (R2)$ and the repeat count is 0, the termination stage stores 0 as the remnant repeat count and the P-register is not incremented.

1. +0 is greater than -0.

9.6.11. Masked Search Within Range - MSW 71,04

Skips NI if $(A_a)\ \boxed{\text{AND}}\ (R2) < (U)\ \boxed{\text{AND}}\ (R2) \leq (A_{a+1})\ \boxed{\text{AND}}\ (R2)$, else repeat.

During the initial stage, the contents of the repeat count register (R1) is transferred to the index subsection, the contents of A_a, A_{a+1}, and R2 are transferred to the arithmetic section, the logical products of the values from A_a and R2 and the values from A_{a+1} and R2 are formed, and the P-register is incremented.

If the initial repeat count is 0, the Next Instruction (NI) is initiated.

If the initial repeat count is not 0, the first test stage is initiated. During each test stage, the repeat count is decreased and the contents of U is transferred to the arithmetic section. The logical products are compared and:

- If $(U)\ \boxed{\text{AND}}\ (R2) > (A_a)\ \boxed{\text{AND}}\ (R2)$ and $(U)\ \boxed{\text{AND}}\ (R2) \leq (A_{a+1})\ \boxed{\text{AND}}\ (R2)$, the termination stage is initiated. This stage stores the remnant repeat count and increments the P-register. (Skip NI.)

- If $(U)\ \boxed{\text{AND}}\ (R2) \leq (A_a)\ \boxed{\text{AND}}\ (R2)$ or $(U)\ \boxed{\text{AND}}\ (R2) > (A_{a+1})\ \boxed{\text{AND}}\ (R2)$ and the repeat count is not 0, another test stage is initiated.

- If $(U)\ \boxed{\text{AND}}\ (R2) \leq (A_a)\ \boxed{\text{AND}}\ (R2)$ or $(U)\ \boxed{\text{AND}}\ (R2) > (A_{a+1})\ \boxed{\text{AND}}\ (R2)$ and the repeat count is 0, the termination stage stores 0 as the remnant repeat count and the P-register is not incremented.

1. Normally, $(A_a)\ \boxed{\text{AND}}\ (R2) < (A_{a+1})\ \boxed{\text{AND}}\ (R2)$. If, however, $(A_a)\ \boxed{\text{AND}}\ (R2) \geq (A_{a+1})\ \boxed{\text{AND}}\ (R2)$, no possible value of U will satisfy the search condition.

2. +0 is greater than -0.

9.6.12. Masked Search Not Within Range - MSNW 71,05

Skips NI if $(U)\ \boxed{\text{AND}}\ (R2) \leq (A_a)\ \boxed{\text{AND}}\ (R2)$ or $(U)\ \boxed{\text{AND}}\ (R2) > (A_{a+1})\ \boxed{\text{AND}}\ (R2)$, else repeat

During the initial stage, the contents of the repeat count register (R1) is transferred to the index subsection, the contents of A_a and R2 are transferred to the arithmetic section, the logical products of the values from A_a and R2 and the values from A_{a+1} and R2 are formed, and the P-register is incremented.

If the initial repeat count is 0, the Next Instruction (NI) is initiated.

If the initial repeat count is not 0, the first test stage is initiated. During each test stage, the repeat count is decreased and the contents of U is transferred to the arithmetic section. The logical products are compared and:

- If $(U) \boxed{\text{AND}} (R2) \leq (A_a) \boxed{\text{AND}} (R2)$ or $(U) \boxed{\text{AND}} (R2) > (A_{a+1}) \boxed{\text{AND}} (R2)$, the termination stage is initiated. This stage stores the remnant repeat count and increments the P-register. (Skip NI.)
- If $(U) \boxed{\text{AND}} (R2) > (A_a) \boxed{\text{AND}} (R2)$ and $(U) \boxed{\text{AND}} (R2) \leq (A_{a+1}) \boxed{\text{AND}} (R2)$ and the repeat count is not 0, another test stage is initiated.
- If $(U) \boxed{\text{AND}} (R2) > (A_a) \boxed{\text{AND}} (R2)$ and $(U) \boxed{\text{AND}} (R2) \leq (A_{a+1}) \boxed{\text{AND}} (R2)$ and the repeat count is 0, the termination stage stores 0 as the remnant repeat count and the P-register is not incremented.

1. Normally, $(A_a) \boxed{\text{AND}} (R2) < (A_{a+1}) \boxed{\text{AND}} (R2)$. If, however, $(A_a) \boxed{\text{AND}} (R2) \geq (A_{a+1}) \boxed{\text{AND}} (R2)$, every possible value of U will satisfy at least one of the following conditions:

   $(U) \boxed{\text{AND}} (R2) \leq (A_a) \boxed{\text{AND}} (R2)$

   $(U) \boxed{\text{AND}} (R2) > (A_{a+1}) \boxed{\text{AND}} (R2)$

2. +0 is greater than -0.

9.6.13. Masked Alphanumeric Search Less Than or Equal – MASL 71,06

Skips NI if $(U) \boxed{\text{AND}} (R2) \leq (A_a) \boxed{\text{AND}} (R2)$, else repeat

During the initial stage, the contents of the repeat count register (R1) is transferred to the index subsection, the contents of A_a and R2 are transferred to the arithmetic section, the logical product of the values from A_a and R2 is formed, and the P-register is incremented.

If the initial repeat count is 0, the Next Instruction (NI) is initiated.

If the initial repeat count is not 0, the first test stage is initiated. During each test stage, the repeat count is decreased and the contents of U is transferred to the arithmetic section. $(U) \boxed{\text{AND}} (R2)$ is compared alphanumerically to $(A_a) \boxed{\text{AND}} (R2)$, and:

- If $(U) \boxed{\text{AND}} (R2) \leq (A_a) \boxed{\text{AND}} (R2)$, the termination stage is initiated. This stage stores the remnant repeat count and increments the P-register. (Skip NI.)
- If $(U) \boxed{\text{AND}} (R2) > (A_a) \boxed{\text{AND}} (R2)$ and the repeat count is not 0, another test stage is initiated.
- If $(U) \boxed{\text{AND}} (R2) > (A_a) \boxed{\text{AND}} (R2)$ and the repeat count is 0, the termination stage stores 0 as the remnant repeat count, and the P-register is not incremented.

1. -0 is greater than +0.

9.6.14. Masked Alphanumeric Search Greater - MASG 71,07

Skips NI if (U) $\boxed{\text{AND}}$ (R2) > (A_a) $\boxed{\text{AND}}$ (R2), else repeat

During the initial stage, the contents of the repeat count register (R1) is transferred to the index subsection, the contents of A_a and R2 are transferred to the arithmetic section, the logical product of the values from A_a and R2 is formed, and the P-register is incremented.

If the initial repeat count is 0, the Next Instruction (NI) is initiated.

If the initial repeat count is not 0, the first test stage is initiated. During each test stage, the repeat count is decreased and the contents of U is transferred to the arithmetic section. (U) $\boxed{\text{AND}}$ (R2) is compared alphanumerically to (A_a) $\boxed{\text{AND}}$ (R2), and:

- If (U) $\boxed{\text{AND}}$ (R2) > (A_a) $\boxed{\text{AND}}$ (R2), the termination stage is initiated. This stage stores the remnant repeat count and increments the P-register (skip NI).
- If (U) $\boxed{\text{AND}}$ (R2) $\leq$ (A_a) $\boxed{\text{AND}}$ (R2) and the repeat count is not 0, another test stage is initiated.
- If (U) $\boxed{\text{AND}}$ (R2) $\leq$ (A_a) $\boxed{\text{AND}}$ (R2) and the repeat count is 0, the termination stage stores 0 as the remnant repeat count, and the P-register is not incremented.

1. -0 is greater than +0.

9.7. Test (or Skip) Instructions

Test instructions are used to read one or more words from storage or control registers and test for certain conditions. The result of the test is used to determine whether the instruction addressed by the incremented contents of the P-register (next instruction) should be performed or skipped.

The Next instruction (NI) is always read from storage. If the decision is made to skip NI, it is discarded, the P-register is incremented a second time, and the contents of the P-register is then used to address the following instruction.

Indirect addressing, indexing, and index register incrementation/decrementation operate normally.

9.7.1. Test Even Parity - TEP 44

Skips NI if (U) $\boxed{\text{AND}}$ (A_a) has even parity.

The value from U is transferred to the arithmetic section under j-field control, where it is used with the contents of A_a to form a 36-bit logical product.

If (U) $\boxed{\text{AND}}$ (A_a) has an even number of 1 bits, the Next Instruction (NI) is skipped, and the instruction following NI is performed.

If (U) $\boxed{\text{AND}}$ (A_a) has an odd number of 1 bits, NI is performed.

9.7.2. Test Odd Parity - TOP 45

Skips NI if (A_a) $\boxed{\text{AND}}$ (U) has odd parity.

The contents of U is transferred to the arithmetic section under j-field control, where it is used with the contents of A_a to form a 36-bit logical product.

If (U) $\boxed{\text{AND}}$ (A_a) has an odd number of 1 bits, the Next Instruction (NI) is skipped and the instruction following NI is performed.

If (U) $\boxed{\text{AND}}$ (A_a) has an even number of 1 bits, NI is performed.

9.7.3. Test Less Than or Equal/Test Not Greater Than Modifier - TLEM,TNGM 47

Skips NI if $(U)_{17-0} \leq (X_a)_{17-0}$;
always $(X_a)_{17-0} + (X_a)_{35-18} \rightarrow X_{a\ 17-0}$

The contents of U is transferred to the arithmetic section under j-field control. The contents of the index register addressed by the a-field (X_a) is transferred to the arithmetic section. The rightmost 18 bits of the value from U is subtracted from the rightmost 18 bits of the value from X_a (this is performed as if the leftmost 18 bits of each operand were 0's).

If $(U)_{17-0} \leq (X_a)_{17-0}$ (the sign of the difference is positive), the Next Instruction (NI) is skipped and the instruction following NI is performed.

If $(U)_{17-0} > (X_a)_{17-0}$ (the sign of the difference is negative), NI is performed.

In either case, the leftmost 18 bits from X_a are added to the rightmost 18 bits from X_a, and the sum is stored in the rightmost 18 bit positions of X_a. The leftmost 18 bit positions of X_a are not changed.

1. If a = 0, index register zero (X0) is referenced.

2. +0 is less than -0.

3. Both $X_{a\ 17-0}$ and the value from U are considered an 18-bit numeric value with a positive sign implied.

4. Only the rightmost 18 bits of the value from U are involved in the operation. Values of 0, 1, or 3 in the j-field yield the same results. Values of 16_8 or 17_8 in the j-field yield the same result.

5. If h = 1 and a = x, the specified index register is incremented or modified only once.

9.7.4. Test Zero - TZ 50

Skips NI if (U) = ±0

The contents of U is transferred to the arithmetic section under j-field control.

If the value transferred is ±0, the Next Instruction (NI) is skipped and the instruction following NI is performed.

If the value transferred is not ±0, NI is performed.

1. The contents of the a-field is ignored.

9.7.5. Test Nonzero - TNZ 51

Skips NI if (U) $\neq$ ±0

The contents of U is transferred to the arithmetic section under j-field control.

If the value transferred is not ±0, the Next Instruction (NI) is skipped and the instruction following NI is performed.

If the value transferred is ±0, NI is performed.

1. The contents of the a-field is ignored.

9.7.6. Test Equal - TE 52

Skips NI if (U) = (A_a)

The contents of U is transferred to the arithmetic section under j-field control. The contents of A_a is also transferred to the arithmetic section.

If (U) = (A_a), the Next Instruction (NI) is skipped and the instruction following NI is performed.

If (U) $\neq$ (A_a), NI is performed.

1. +0 is not equal to -0.

9.7.7. Test Not Equal - TNE 53

Skips NI if (U) $\neq$ (A_a)

The contents of U is transferred to the arithmetic section under j-field control. The contents of A_a is also transferred to the arithmetic section.

If (U) $\neq$ (A_a), the Next Instruction (NI) is skipped and the instruction following NI is performed.

If (U) = (A_a), NI is performed.

1. +0 is not equal to -0.

9.7.8. Test Less Than or Equal/Test Not Greater - TLE,TNG 54

Skips NI if (U) $\leq$ (A_a)

The contents of U is transferred to the arithmetic section under j-field control. The contents of A_a is also transferred to the arithmetic section.

If (U) $\leq$ (A_a), the Next Instruction (NI) is skipped and the instruction following NI is performed.

If (U) > (A_a), NI is performed.

1. +0 is greater than -0.

9.7.9. Test Greater - TG 55

Skips NI if (U) > (A_a)

The contents of U is transferred to the arithmetic section under j-field control. The contents of A_a is also transferred to the arithmetic section.

If (U) > (A_a), the Next Instruction (NI) is skipped and the instruction following NI is performed.

If (U) $\leq$ (A_a), NI is performed.

1. +0 is greater than -0.

9.7.10. Test Within Range - TW 56

Skips NI if (A_a) < (U) $\leq$ (A_{a+1})

The contents of U is transferred to the arithmetic section under j-field control. The contents of A_a and A_{a+1} are also transferred to the arithmetic section.

If (A_a) < (U) $\leq$ (A_{a+1}), the Next Instruction (NI) is skipped and the instruction following NI is performed.

If (U) $\leq$ (A_a) or (U) > (A_{a+1}), NI is performed.

1. +0 is greater than -0.
2. Normally, (A_a) < (A_{a+1}). If, however, (A_a) $\geq$ (A_{a+1}), there is no value of U that can satisfy the condition (A_a) < (U) $\leq$ (A_{a+1}).

9.7.11. Test Not Within Range - TNW 57

Skips NI if (U) $\leq$ (A_a) or (U) > (A_{a+1})

The contents of U is transferred to the arithmetic section under j-field control. The contents of A_a and A_{a+1} are also transferred to the arithmetic section.

If (U) $\leq$ (A_a) or (U) > (A_{a+1}), the Next Instruction (NI) is skipped and the instruction following NI is performed.

If (U) > (A_a) and (U) $\leq$ (A_{a+1}), NI is performed.

1. +0 is greater than -0.
2. Normally, (A_a) < (A_{a+1}). If, however, (A_a) $\geq$ (A_{a+1}), every possible value of U will satisfy at least one of the following conditions: (U) $\leq$ (A_a), or (U) > (A_{a+1}).

9.7.12. Test Positive - TP 60

Skips NI if $(U)_{35} = 0$

The contents of U is transferred to the arithmetic section under j-field control.

If the sign bit (bit 35) of the value from U is 0, the Next Instruction (NI) is skipped and the instruction following NI is performed.

If the sign bit is 1, NI is performed.

1. The contents of the a-field is ignored.
2. Always skip when j = H1, H2, Q1-Q4, or S1-S6.

9.7.13. Test Negative - TN 61

Skips NI if $(U)_{35} = 1$

The contents of U is transferred to the arithmetic section under j-field control.

If the sign bit (bit 35) of the value from U is 1, the Next Instruction (NI) is skipped and the instruction following NI is performed.

If the sign bit is 0, NI is performed.

1. The contents of the a-field is ignored.
2. Never skip when j = H1, H2, Q1-Q4, or S1-S6.

9.7.14. Double-Precision Test Equal - DTE 71,17

Skips NI if $(U, U+1) = (A_a, A_{a+1})$

The contents of U, U+1, A_a, and A_{a+1} are transferred to the arithmetic section. U, U+1 and A_a, A_{a+1} are 72-bit operands.

If $(U, U+1) = (A_a, A_{a+1})$, the Next Instruction (NI) is skipped and the instruction following NI is performed.

If $(U, U+1) \neq (A_a, A_{a+1})$, NI is performed.

1. +0 is not equal to -0.

9.8. Shift Instructions

Each shift instruction transfers either one or two words to the arithmetic section, moves or shifts the bits of the words, and stores the shifted word or words in one or two control registers.

The following basic types of shifts are provided for both single-word (36-bit input operand) and double-word (two 36-bit words treated as a 72-bit input operand) operations:

- Right circular

  For a right-circular shift, a shift count of *n* moves the contents of all bit positions of the register, which holds the input operand, *n* bit positions to the right. Bits shifted out the right end of the register appear in the leftmost bit positions vacated by the shift.

- Left circular

  For a left-circular shift, a shift count of *n* moves the contents of all bit positions of the register, which holds the input operand, *n* place to the left. Bits shifted out the left end of the register appear in the rightmost bit positions vacated by the shift.

  For example: A shift count of 6 for a right-circular shift applied to 765432101234_8 (the input operand) produces 347654321012_8. The same result is produced using a shift count of 30 for a left-circular shift.

  For a single-word circular shift, a shift count of 72 or 36 produces the same result as a shift count of 0 (no shift). A shift count of 37 produces the same effect as a shift count of 1; a shift count of 38 produces the same effect as a shift count of 2, and so on.

- Right logical

  For a right-logical shift, a shift count of *n* moves the contents of all bit positions of the register, which holds the input operand, *n* places to the right. Bits shifted out the right end of the register are lost. The leftmost bit positions vacated by the shift are zero filled.

  For example: A shift count of 6 for a right-logical shift applied to 765432101234_8 (the input operand) produces 007654321012_8.

- Left Logical

  For a left-logical shift, a shift count of *n* moves the contents of all bit positions of the input operand register *n* places to the left. Bits shifted out the left end of the register are lost. The rightmost bit positions vacated by the shift are zero filled.

  For example: A shift count of 6 for a left-logical shift applied to 765432101234_8 (the input operand) produces 543210123400_8.

- Right algebraic

  For an algebraic shift (right only, since no left algebraic shift is provided), a shift count of *n* moves the contents of all bit positions of the register, which holds the input operand, *n* places to the right. Bits shifter out the right end of the register are lost. The bit positions vacated by the shift are filled with bits identical to the leftmost bit (sign bit) of the original input operand.

  For example: A shift count of 6 for an algebraic shift applied to 765432101234_8 (the input operand) produces 777654321012_8.

The two Load Shift and Count instructions are basically left-circular shift instructions. The shift count is determined by the configuration of the bits of the input operand. If the two leftmost bits are not identical, the shift count is 0. If the two leftmost bits are identical, the operand is shifted left circularly by a minimum amount to position the bits of the input operand so that the two leftmost bits are not identical. The shift count is the count of the number of bit positions shifted. If all bits of an input operand are identical, no amount of circular shifting will position its bits so that the two leftmost bits are not identical. In this instance, the shift count is 35 (single-word operand) or 71 (double-word operand). The shift count is stored in a control register.

For all shift instructions, except the two Load Shift and Count instructions, the input operands are specified by one or two A-registers, and the shift count is specified by bits 6 through 0 of the effective U. Indirect addressing, indexing, and index register incrementation/ decrementation operate normally for all shift instructions.

The shift count can be any number between 0 and 72. If a shift count of 73 to 127 (111_8 through 177_8) is specified, the result produced is undefined. The value in the u-field of the shift instruction and the value of X_m (if $x \neq 0$) must be chosen accordingly.

For the two Load Shift and Count instructions, the effective U specifies the input operand address just as for the other load instructions. The scaled result is loaded in the specified A-register (A_a, A_{a+1} for Double Load Shift and Count instruction). The number of shifts required for scaling is stored in the next consecutive register A_{a+1} (or A_{a+2} for Double Load Shift and Count instruction).

9.8.1. Single Shift Circular - SSC 73,00

Shifts (A_a) right circularly U places

The contents of A_a is transferred to the arithmetic section. The shift count from bits 6 through 0 of U is transferred to the arithmetic section. The value from A_a is shifted right circularly by the number of bit positions specified by the shift count. The shifted value is stored in A_a.

1. The result stored is not defined for shift counts greater than 72.

2. If $36 \leq n \leq 72$, a shift count of n produces the same result as a shift count of n–36.

9.8.2. Double Shift Circular - DSC 73,01

Shifts (A_a, A_{a+1}) right circularly U places

The contents of A_a and A_{a+1} are transferred to the arithmetic section. The shift count from bits 6 through 0 of U is transferred to the arithmetic section. The 72-bit value from A_a and A_{a+1} is shifted right circularly the number of bit positions specified by the shift count. The shifted value is stored in A_a and A_{a+1}.

1. The result stored is not defined for shift counts greater than 72.

9.8.3. Single Shift Logical - SSL 73,02

Shifts (A_a) right U places, zero fill

The contents of A_a is transferred to the arithmetic section. The shift count from bits 6 through 0 of U is transferred to the arithmetic section. The value from A_a is right shifted the number of bit positions specified by the shift count. Bits shifted out of the rightmost bit positions are lost; the vacated leftmost bit positions are zero filled. The shifted value is stored in A_a.

1. The stored result is not defined for shift counts greater than 72.

2. If $36 \leq U \leq 72$, the result stored in A_a is +0.

9.8.4. Double Shift Logical - DSL 73,03

Shifts (A_a, A_{a+1}) right U places, zero fill

The contents of A_a and A_{a+1} are transferred to the arithmetic section. The shift count from bits 6 through 0 of U is transferred to the arithmetic section. The 72-bit value from A_a and A_{a+1} is right shifted the number of bit positions specified by the shift count. Bits shifted out of the rightmost bit positions are lost; the vacated leftmost bit positions are zero filled.

1. The result stored is not defined for shift counts greater than 72.

9.8.5. Single Shift Algebraic - SSA 73,04

Shifts (A_a) right U places, sign fill

The contents of A_a is transferred to the arithmetic section. The shift count from bits 6 through 0 of U is transferred to the arithmetic section. The value from A_a is right shifted the number of bit positions specified by the shift count. Bits shifted out of the rightmost bit positions are lost; bits identical to the contents of bit 35 of the initial value from A_a appear in the vacated leftmost bit positions. The shifted count is stored in A_a.

1. The result stored is not defined for shift counts greater than 72.

2. If $35 \leq U \leq 72$, all bits of the result stored in A_a are identical to the leftmost bit of the input operand from A_a.

9.8.6. Double Shift Algebraic - DSA 73,05

Shifts (A_a, A_{a+1}) right U places, sign fill

The contents of A_a and A_{a+1} are transferred to the arithmetic section. The shift count from bits 6 through 0 of U is transferred to the arithmetic section. The 72-bit value from A_a and A_{a+1} is right shifted the number of bit positions specified by the shift count. Bits shifted out of the rightmost bit positions are lost; bits identical to the contents of bit 35 of the initial value from A_a appear in the vacated leftmost bit positions. The shifted value is stored in A_a and A_{a+1}.

1. The result stored is not defined for shift counts greater than 72.

9.8.7. Load Shift and Count - LSC 73,06

$(U) \rightarrow A_a$;
shift (A_a) left circularly until $(A_a)_{35} \neq (A_a)_{34}$;
number of shifts $\rightarrow A_{a+1}$

The contents of location U is transferred to a nonaddressable 36-bit register in the arithmetic section and then shifted left circularly the minimum number of bit positions that will make bit 35 unequal to bit 34. The resultant scaled number is transferred to A_a and the shift count to A_a+1.

1. If bit 35 of the value from location U is not equal to bit 34, the number is already scaled and no shift occurs: $(U) \rightarrow A_a$; $+0 \rightarrow A_{a+1}$.

2. If the value from location U is ±0: $(U) \rightarrow A_a$, the shift count is 35, and $43_8 \rightarrow A_{a+1}$.

9.8.8. Double Load Shift and Count - DLSC 73,07

$(U, U+1) \rightarrow A_a, A_{a+1}$;
shift (A_a, A_{a+1}) left circularly until $(A_a, A_{a+1})_{71} \neq (A_a, A_{a+1})_{70}$;
number of shifts $\rightarrow A_{a+2}$

The contents of U and U+1 are transferred to a nonaddressable 72-bit register in the arithmetic section and then shifted left circularly the minimum number of bit positions that will make bit 71 unequal to bit 70. The resultant scaled number is transferred to A_a and A_{a+1} and the shift count to A_{a+2}.

1. If bit 71 of the value from U and U+1 is not equal to bit 70, the double length number is already scaled and no shift occurs: $(U) \rightarrow A_a$; $(U+1) \rightarrow A_{a+1}$; $+0 \rightarrow A_{a+2}$.

2. If the double-length value from locations U and U+1 is ±0: $(U) \rightarrow A_a$; $(U+1) \rightarrow A_{a+1}$; the shift count is 71; $107_8 \rightarrow A_{a+2}$.

9.8.9. Left Single Shift Circular - LSSC 73,10

Shifts (A_a) left circularly U places

The contents of A_a is transferred to the arithmetic section. The shift count from bits 6 through 0 of U is transferred to the arithmetic section. The value from A_a is shifted left circularly the number of bit positions specified by the shift count. The shifted value is stored in A_a.

1. The result stored is undefined for shift counts greater than 72.

2. If $36 \leq n \leq 72$, a shift count of n produces the same result as a shift count of n-36.

9.8.10. Left Double Shift Circular - LDSC 73,11

Shifts (A_a, A_{a+1}) left circularly U places

The contents of A_a and A_{a+1} are transferred to the arithmetic section. The shift count from bits 6 through 0 of U is transferred to the arithmetic section. The 72-bit value from A_a and A_{a+1} is shifted left circularly the number of bit positions specified by the shift count. The shifted value is stored in A_a and A_{a+1}.

1. The result stored is undefined for shift counts greater than 72.

9.8.11. Left Single Shift Logical - LSSL 73,12

Shifts (A_a) left U places, zero fill

The contents of A_a is transferred to the arithmetic section. The shift count from bits 6 through 0 of U is transferred to the arithmetic section. The value from A_a is left shifted the number of bit positions specified by the shift count. Bits shifted out of the leftmost bit positions are lost; the vacated rightmost bit positions are zero filled. The shifted value is stored in A_a.

1. The result stored is undefined for shift counts greater than 72.
2. If $36 \leq U \leq 72$, the result stored in A_a is +0.

9.8.12. Left Double Shift Logical - LDSL 73,13

Shifts (A_a, A_{a+1}) left U places, zero fill

The contents of A_a and A_{a+1} are transferred to the arithmetic section. The shift count from bits 6 through 0 of U is transferred to the arithmetic section. The 72-bit value from A_a and A_{a+1} is left shifted the number of bit positions specified by the shift count. Bits shifted out of the leftmost bit positions are lost; the vacated rightmost bit positions are zero filled. The shifted value is stored in A_a and A_{a+1}.

1. The result stored is undefined for shift counts greater than 72.

9.9. Unconditional Jump Instructions

A jump is a change in the sequence by which instructions are executed. It is accomplished by placing a new value in the P-register. Each unconditional jump instruction performs a unique operation in addition to the common operation of placing a new value in the P-register.

If the relative "jump to" address is less than 0200, the next instruction is taken from the storage location addressed by the value rather than from a control register.

The Jump Keys instruction can be used to specify either a conditional or an unconditional jump. The Halt Jump/Halt Keys and Jump instruction specifies an unconditional jump, but the halt portion is conditional. Both of these instructions are included in the section on conditional jump instructions (See 9.11).

9.9.1. Store Location and Jump - SLJ 72,01

Relative $P+1 \rightarrow U_{17-0}$;
jump to U+1

The P-register is incremented. An 18-bit relative return address is stored in the rightmost 18 bits of the location specified by the operand address. The value of the operand address plus one is transferred to the P-register as the "jump to" address. The upper half of the operand is unchanged.

1. The contents of the a-field is ignored.
2. If $U < 0200$, the 18-bit relative return address is stored in the rightmost 18 bits of the control register addressed by U, and the leftmost 18 bit positions of that control register are unchanged.
3. The relative return address is stored in the low-order 18 bits of a word. If this 18-bit relative return address is larger than 16 bits, the two high-order bits will be interpreted as h- and i-bits if the address is used in an instruction. The instruction may produce erroneous results.

9.9.2. Load Modifier and Jump - LMJ 74,13

Relative P+1 → X_{a17-0};
jump to U

The P-register is incremented. An 18-bit relative return address is stored in the rightmost 18 bits of the index register specified by the a-field. The leftmost 18 bits of that index register are not affected. The value of the operand is transferred to the P-register as the "jump to" address.

1. If the GRS Selection designator (D6) is 0 and the value in the a-field is 0, the relative return address is stored in index register zero (X0).
2. If index register incrementation is specified, the relative return address is stored in the index register specified by the a-field after the new value for X_m is stored in the index register specified by the x-field. As a consequence, if the value in the a-field is not 0 and it is the same as the value in the x-field, it makes no difference whether the value in the h-field is 0 or 1.

9.9.3. Allow All Interrupts and Jump - AAIJ 74,07

Allows all interrupts and jumps to U

This instruction allows interrupts prevented by the occurrence of an interrupt or the execution of a Prevent All Interrupts and Jump instruction.

1. The contents of the a-field is ignored.
2. The Allow All Interrupts and Jump instruction does not affect the Dayclock interrupt when it is disabled by the Disable Dayclock instruction and enabled by the Enable Dayclock instruction.

9.10. Bank Descriptor Selection Instructions

Each program may be composed of or associated with a large number of program or data segments; of these, up to four may be active at any given time. Bank Descriptor selection instructions allow a program to select which segments are among the four that are currently active.

9.10.1. Load Bank and Jump - LBJ 07,17

Loads BDR selection by bit position 34–33;
jump to U

The LBJ instruction loads the Bank Descriptor Register (BDR) selected in bits 34 and 33 of the index register specified by the a-field of the instruction word (X_a) with a new bank descriptor, stores the old bank descriptor specifications and relative program address in X_a as return information, and then jumps to the location specified by the operand address. An Address Exception interrupt occurs if the BDR in X_a is not available for use as defined by the designator register. The new bank descriptor is located by adding the Bank Descriptor Index (BDI) contained in bits 18 through 29 of X_a to the bank descriptor table pointer selected by bit 35 of X_a. If bit 35 is 0, the user pointer and table are selected; if bit 35 is 1 and if the EXEC Bank Descriptor Table Pointer Enable designator (D19) is 1, the Executive pointer and table are selected. An Address Exception interrupt also occurs if the BDI value exceeds the length of the table or if bit 35 of X_a is 1 and D19 is 0.

Before the new bank descriptor values are actually loaded, the old bank descriptor is located and the use-count field is decreased by 1 under storage lock. An Addressing Exception interrupt occurs if the C-flag of the old bank descriptor is 1 and the use count is decreased to 0, or if the use count is decreased from 0 to all 1's. The new bank descriptor is loaded in the bank descriptor register, the P-flag is transferred to Privileged Instruction and GRS Protection designator (D2), and the W-flag of the new bank descriptor is placed in the appropriate write-protection bit of the designator register (D13 through D16).

When the new bank descriptor is located, the associated use count field is increased by 1 under storage lock, and an Addressing Exception interrupt occurs if the R-flag of the bank descriptor is 1, if there is a V-flag violation, or if the use count field is increased from all 1's to 0.

The specifications of the old bank descriptor are copied from the GRS processor state area into the upper half of X_a, the relative program address is copied into the lower half of X_a, and the specifications of the new bank descriptor are then stored in the GRS. The operand address is formed and a jump to that location is effected. If both an address exception and jump address guard mode limits violation occur during the execution of this instruction, the address exception will be taken.

The following are the formats of X_a before and after execution of the instruction:

X_a Before Execution of LDJ, LIJ, or LBJ instruction

E	BDR	0 — 0	New BDI	Not Used
35	34 33	32 30	29 18	17 0

X_a After Execution of LDJ, LIJ, or LBJ instruction

E	BDR	0 — 0	Old BDI	Relative Program Address
35	34 33	32 30	29 18	17 0

9.10.2. Load I-Bank Base and Jump - LIJ 07,13

Ignores $(X_a)_{34-33}$;
if D12 = 0, load BDRO;
if D12 = 1, select BDR1;
jump to U

The LIJ instruction is executed as a special case of the LBJ instruction. Bits 34-33 of X_a are ignored; if the BDR Selector designator (D12) is 0, BDR0 is selected, and if D12 is 1, BDR1 is selected.

9.10.3. Load D-Bank Base and Jump - LDJ 07,12

Ignores $(X_a)_{34-33}$;
if D12 = 0 load BDR2;
if D12 = 1, select BDR3;
jump to U

The LDJ instruction is executed as a special case of the LBJ instruction. Bits 34-33 of X_a are ignored; if the BDR selector designator (D12) is 0, BDR2 is selected, and if D12 is 1, BDR3 is selected.

9.11. Conditional Jump Instructions

Each of the conditional jump instructions performs a test for a specific condition (or set of conditions). If the condition is satisfied, the value U is transferred to the P-register and the instruction addressed by U is performed next. If the condition is not satisfied, the Next Instruction (NI) is performed.

9.11.1. Jump Greater and Decrement - JGD 70

Jumps to U if $(\text{Control Register})_{ja} > 0$;
goes to NI if $(\text{Control Register})_{ja} \leq 0$;
always $(\text{Control Register})_{ja} -1 \rightarrow \text{Control Register}_{ja}$

If the 36-bit signed number in the control register addressed by the rightmost 7 bits of the ja-field is greater than 0 (bit 35 is 0) and the number does not consist of all 0's, the instruction at location U is executed next. If the number is less than, or equal to, 0 (bit 35 is 1) or the number is all 0's, the next instruction is performed. In either case, the number is decreased by 1 and the difference is stored in the control register addressed by the ja-field.

1. A Guard Mode interrupt occurs (if guard mode is set) when the ja-field specifies a value in the range 40_8 through 100_8, or 120_8 through 177_8 when the Privileged Instruction and GRS Protection designator (D2) is 1. This is true regardless of the value of the GRS Selection designator (D6) (A-, X-, and R-register set selector).

2. The leftmost bit in the j-field is ignored.

9.11.2. Double-Precision Jump Zero - DJZ 71,16

Jumps to U if $(A_a, A_{a+1}) = \pm 0$;
goes to NI if $(A_a, A_{a+1}) \neq \pm 0$

If A_a and A_{a+1} is ± 0, the instruction at location U is performed next. If the 72-bit operand contained in A_a and A_{a+1} is ± 0, the next instruction is performed.

9.11.3. Jump Positive and Shift - JPS 72,02

Jumps to U if $(A_a)_{35} = 0$;
goes to NI if $(A_a)_{35} = 1$;
always shift (A_a) left circularly one bit position

If bit 35 of A_a is 0, the instruction at location U is performed next. If bit 35 is 1, the NI is performed. The contents of A_a is always shifted left circularly one bit position.

1. The bit shifted out of bit 35 of A_a is shifted to bit 0 of A_a.

9.11.4. Jump Negative and Shift – JNS 72,03

Jumps to U if $(A_a)_{35} = 1$;
goes to NI if $(A_a)_{35} = 0$;
always shift (A_a) left circularly one bit position

If bit 35 of A_a is 1, the instruction at location U is performed next. If bit 35 is 0, the NI is performed. The contents of A_a is always shifted left circularly one bit position.

1. The bit shifted out of bit 35 of A_a is shifted to bit 0 of A_a.

9.11.5. Jump Zero – JZ 74,00

Jumps to U if $(A_a) = \pm 0$;
goes to NI if $(A_a) \neq \pm 0$

If (A_a) is ±0, the instruction at location U is performed next. If A_a does not contain ±0, the NI is performed.

9.11.6. Jump Nonzero – JNZ 74,01

Jumps to U if $(A_a) \neq \pm 0$;
goes to NI if $(A_a) = \pm 0$

If (A_a) is not ±0, the instruction at location U is performed next. If (A_a) is ±0, the NI is performed.

9.11.7. Jump Positive – JP 74,02

Jumps to U if $(A_a)_{35} = 0$;
goes to NI if $(A_a)_{35} = 1$

If bit 35 of A_a is 0, the instruction at location U is performed next. If bit 35 is 1, the NI is performed.

9.11.8. Jump Negative – JN 74,03

Jumps to U if $(A_a)_{35} = 1$;
goes to NI if $(A_a)_{35} = 0$

If bit 35 of A_a is 1, the instruction at location U is performed next. If bit 35 is 0, the NI is performed.

9.11.9. Jump/Jump Keys - J,JK 74,04

Jumps to U if a = 0 or if a = set JUMP SELECT function;
goes to NI if neither is true

If the a-field contains all 0's, the instruction at location U is performed next. If the a-field contains a value in the range of 1 through 15 (1_8 through 17_8) and the correspondingly numbered JUMP SELECT function is set, the instruction at location U is performed next; if the correspondingly numbered JUMP SELECT function is not set, the next instruction is performed.

1. The 15 JUMP SELECT functions are enabled by the SSP console display keyboard.

2. Care should be exercised in using a value other than all 0's in the a-field if the program is to run concurrently with one or more other programs. Any other program may include a Jump Keys instruction with the same value in the a-field and specify that it is to be run with the corresponding JUMP SELECT function.

9.11.10. Halt Jump/Halt Keys and Jump - HJ,HKJ 74,05

Stops if [a=0 $\boxed{\text{OR}}$ if (a-field $\boxed{\text{AND}}$ set SELECT STOPS function) $\neq$ 0] $\boxed{\text{AND}}$ D = 0;
on restart or continuation jump to U

If the a-field contains all 0's, the execution of program instruction halts. If the a-field contains a 1 in a bit position that corresponds to an enabled HALT SELECT function, the program halts. In either case, a manual restart causes the instruction at location U to be executed next.

If neither of the conditions are fulfilled, the instruction at U is performed and the program does not halt.

1. Unless the central processing unit (CPU) is operating in privileged mode (D2 is 0) when a halt condition is satisfied for a Halt Keys and Jump instruction, it does not actually halt. Instead, it proceeds with the jump.

2. The four HALT SELECT functions are enabled via the SSP display keyboard.

3. If the Program Address Register is manually changed while the CPU is halted, program execution will resume at the new address when the CPU is restarted.

9.11.11. Jump No Low Bit - JNB 74,10

Jumps to U if $(A_a)_0 = 0$;
goes to NI if $(A_a)_0 = 1$

If bit 0 of A_a is 0, the instruction at location U is performed next. If bit 0 is 1, the next instruction is performed.

1. If the Jump No Low Bit instruction is used to determine whether the value in A_a is an even or an odd integer, consideration must be given to the sign of the value.

9.11.12. Jump Low Bit - JB 74,11

Jumps to U if $(A_a)_0 = 1$;
goes to NI if $(A_a)_0 = 0$

If bit 0 of A_a is 1, the instruction at location U is performed next. If bit 0 is 0, the next instruction is performed.

1. If a Jump Low Bit instruction is used to determine whether the value in A_a is an even or an odd integer, consideration must be given to the sign of the value.

9.11.13. Jump Modifier Greater and Increment - JMGI 74,12

Jumps to U if $(X_a)_{17-0} > 0$;
goes to NI if $(X_a)_{17-0} \leq 0$;
always $(X_a)_{17-0} + (X_a)_{35-18} \rightarrow X_{a\ 17-0}$

If the signed number in bits 17 through 0 of the X-register specified by the a-field is greater than 0 (bit 17 is 0) and the number is not all 0's, the instruction at location U is performed next. If the number is less than or equal to 0 (bit 17 is 1) or the number is all 0's, the next instruction is performed. In either case, the signed number in bits 35 through 18 of the X-register is added to the signed number in bits 17 through 0, and the sum is stored in bits 17 through 0, of the X-register.

1. The number in $X_{a\ 17-0}$ before the addition is tested rather than the number resulting from the addition.
2. If a = x and h = 1, the specified index register is effectively modified only once for each execution of the instruction.

9.11.14. Jump Overflow - J0 74,14; a = 0

Jumps to U if D1 = 1;
goes to NI if D1 = 0

Where the a-field is an extension of the f- and j-fields.

If the Overflow designator (D1) is 1, the instruction at location U is performed next. If D1 is 0, the next instruction is performed.

1. Performing the Jump Overflow instruction does not change D1.

9.11.15. Jump Floating Underflow - JFU 74,14,01

Jumps to U if D21 = 1, clear D21;
goes to NI if D21 = 0

If the Characteristic Underflow designator (D21) is 1, the instruction at location U is performed next, and D21 is cleared by the instruction. If D21 is 0, the next instruction is performed.

9.11.16. Jump Floating Overflow - JFO 74,14,02

Jumps to U if D22 = 1, clear D22;
goes to NI if D22 = 0

If the Characteristic Overflow designator (D22) is 1, the instruction at location U is performed next and D22 is cleared by the instruction. If D22 is 0, the next instruction is performed.

9.11.17. Jump Divide Fault - JDF 74,14,03

Jumps to U if D23 = 1, clear D23;
goes to NI if D23 = 0

If the Divide Fault designator (D23) is 1, the instruction at location U is performed next and D23 is cleared by the instruction. If D23 is 0, the next instruction is performed.

9.11.18. Jump No Overflow - JNO 74,15,00

Jumps to U if D1 = 0;
goes to NI if D1 = 1

If the Overflow designator (D1) is 0, the instruction at location U is performed. If D1 is 1, the next instruction is performed.

1. Executing the Jump No Overflow instruction does not change D1.

9.11.19. Jump No Floating Underflow - JNFU 74,15,01

Jumps to U if D21 = 0;
goes to NI if D21 = 1;
clears D21

If the Characteristic Underflow designator (D21) is 0, the instruction at location U is performed next. If D21 is 1, the next instruction is performed. D21 is cleared by the instruction.

9.11.20. Jump No Floating Overflow - JNFO 74,15,02

Jumps to U if D22 = 0;
goes to NI if D22 = 1;
clears D22

If the Characteristic Overflow designator (D22) is 0, the instruction at location U is performed next. If D22 is 1, the next instruction (NI) is performed. D22 is cleared by the instruction.

9.11.21. Jump No Divide Fault - JNDF 74,15,03

Jumps to U if D23 = 0;
goes to NI if D23 = 1;
clears D23

If the Divide Fault designator (D23) is 0, the instruction at location U is performed. If D23 is 1, the next instruction is performed. D23 is cleared by the instruction.

9.11.22. Jump Carry – JC 74,16

Jumps to U if D0 = 1;
goes to NI if D0 = 0

If the Carry designator (D0) is 1, the instruction at location U is performed next. If D0 is 0, the next instruction is performed.

1. The contents of the a-field is ignored.

2. Performing the Jump Carry instruction does not change D0.

9.11.23. Jump No Carry – JNC 74,17

Jumps to U if D0 = 0;
goes to NI if D0 = 1

If the Carry designator (D0) is 0, the instruction at location U is performed next. If D0 is 1, the next instruction is performed.

1. The contents of the a-field is ignored.

2. Performing the Jump No Carry instruction does not change D0.

9.12. Logical Instructions

The three logical operations are the Logical Inclusive OR (referred to as the Logical OR and symbolized by $\boxed{\text{OR}}$), the Logical Exclusive OR (symbolized by $\boxed{\text{XOR}}$); and the Logical AND (symbolized by $\boxed{\text{AND}}$). Each of these instructions uses two input operands. One input operand is obtained from location U and the other from an A-register. Table 9-1 lists the four possible combinations of the two bits from any bit position of the two input operands and the result produced for that bit position for each of the three basic operations.

Table 9-1. Truth Table for Logical OR, XOR, and AND

Input Bits		Output (Result) Bit		
First Operand	Second Operand	OR	XOR	AND
0	0	0	0	0
0	1	1	1	0
1	0	1	1	0
1	1	1	0	1

The Masked Load Upper instruction performs a compound logical operation; the contents of selected bit positions of one operand are merged with the contents of the remaining bit positions of a second operand.

9.12.1. Logical OR - OR 40

$(A_a)\ \boxed{\text{OR}}\ (U) \rightarrow A_{a+1}$

The contents of A_a is transferred to the arithmetic section. The contents of U is transferred to the arithmetic section under j-field control. A 36-bit result is formed in the arithmetic section, as follows:

- The result contains a 1 in each bit position for which the corresponding bit position of either (or both) of the input operands contains a 1.
- The result contains a 0 in each bit position for which the corresponding bit position of both input operands contains a 0.

The result is stored in A_{a+1}.

9.12.2. Logical Exclusive OR - XOR 41

$(A_a)\ \boxed{\text{XOR}}\ (U) \rightarrow A_{a+1}$

The contents of A_a is transferred to the arithmetic section. The contents of U is transferred to the arithmetic section under j-field control. A 36-bit result is formed in the arithmetic section, as follows:

- The result contains a 1 in each bit position for which the corresponding bit position of either (but not both) of the input operands contains a 1.
- The result contains a 0 in each bit position for which the contents of the corresponding bit position of the input operands are both 0 or both 1.

The result is stored in A_{a+1}.

9.12.3. Logical AND - AND 42

$(A_a)\ \boxed{\text{AND}}\ (U) \rightarrow A_{a+1}$

The contents of A_a is transferred to the arithmetic section. The contents of U is transferred to the arithmetic section under j-field control. A 36-bit result is formed in the arithmetic section, as follows:

- The result contains a 1 in each bit position for which the corresponding bit position of both input operands contains a 1.
- The result contains a 0 in each bit position for which the corresponding bit position of either (or both) of the input operands contains a 0.

The result is stored in A_{a+1}.

9.12.4. Masked Load Upper - MLU 43

$[(U) \boxed{\text{AND}} (R2)] \boxed{\text{OR}} [(A_a) \boxed{\text{AND}} \text{NOT } (R2)] \rightarrow A_{a+1}$

The contents of A_a and R2 are transferred to the arithmetic section. The contents of U is transferred to the arithmetic section under j-field control. A 36-bit result is formed in the arithmetic section, as follows:

- The result contains a 1 in each bit position for which the corresponding bit position of the operand from U and the operand from R2 both contain 1 bits.
- The result contains a 1 in each bit position for which the corresponding bit position of the operand from A_a and the ones complement of the operand from R2 both contain 1 bits.
- The result contains 0 bits in the remaining bit positions.

The result is stored in A_{a+1}.

1. The desired value must be loaded in R2 (masked register) by an instruction preceding the Masked Load Upper instruction.

9.13. Miscellaneous Instructions

Each of the eight following instructions is classed as miscellaneous.

9.13.1. Load DR Designators - LPD 07,14

$U_{6,5,3-1} \rightarrow$ Designator register:

Bit 1 → D5	Bit 5 → D17
Bit 2 → D8	Bit 6 → D20
Bit 3 → D10	

Bits 1, 2, 3, 5, and 6 of U are transferred to the designator register bits (see D5, D8, D10, D17, and D20, respectively). These are the only designator bits that can be changed by a user program.

9.13.2. Store DR Designators - SPD 07,15

Designator register bits $\rightarrow U_{6-1}$; zeros $\rightarrow U_{17-7,0}$

D5 → Bit 1	D12 → Bit 4
D8 → Bit 2	D17 → Bit 5
D10 → Bit 3	D20 → Bit 6

D20, D17, D12, D10, D8, and D5 of the designator register are transferred to bits 6-1 of U, respectively. The upper half of the operation location is unaffected.

9.13.3. Execute - EX 72,10

Executes the instruction at U

The P-register is incremented provided the instruction was addressed by the contents of the P-register. The instruction at location U is transferred to the control section to replace the Execute instruction and becomes the next instruction to be performed.

1. The contents of the a-field are ignored.

2. The remote instruction, specified by U, is always obtained from a storage location.

3. Execute instructions may be cascaded; that is, the instruction in the remote location may be an Execute instruction. Cascaded Executes can not be interrupted except for immediate storage checks or guard modes. Time outs for cascaded Executes are explained under the discussion of the Privileged Instruction and GRS Protection designator (D2) definition.

4. The P-register is incremented only once, when the original Execute instruction is obtained for execution.

5. Generally, an interrupt cannot occur between the time an Execute instruction is started and the instruction (or instructions) that it leads to has been completed, except when an Execute instruction leads to a repeated instruction (see 9.3.8 and 9.6). An interrupt cannot occur between the start of the Execute instruction and the completion of the initial stage of the repeated instruction. The interrupt, however, can cause initiation of a termination stage immediately following completion of the initial stage, or any time thereafter, in order to permit the interrupt to occur.

6. If an Execute instruction leads to a repeated instruction, index register incrementation should not be specified for the Execute instructions or for any indirect addressing sequence involved (see 9.3.8. note 6, and 9.6).

9.13.4. Executive Request - ER 72,11

Generates Executive Request interrupt to MSR + 222_8

An Executive Request interrupt is generated.

1. A Guard Mode/Storage Limits interrupt will occur if indirect addressing is specified (i is 1, D7 is 0) and the operand address causes a storage limits violation.

2. The contents of the a-field is ignored.

9.13.5. Test and Set - TS 73,17; a = 00

If $(U)_{30} = 1$, generates Test and Set interrupt;
if $(U)_{30} = 0$, goes to NI, then $01_8 \rightarrow U_{35-30}$;
$(U)_{29-0}$ unchanged

A storage cycle is initiated to read and then write the operand specified by the operand address. If bit 30 of the operand is 1, a Test and Set interrupt occurs. If bit 30 of the operand is 0, the next instruction is performed. The write portion of the storage cycle includes writing 01_8 in bits 35 through 30 of the storage operand. Bits 29 through 0 at location U are neither examined nor altered.

1. If $U < 200_8$, always interrupt.

9.13.6. Test and Set and Skip - TSS 73,17,01

If $(U)_{30} = 1$, goes to NI;
if $(U)_{30} = 0$, skips NI;
and $01_8 \rightarrow U_{35-30}$;
$(U)_{29-0}$ unchanged

A storage cycle is initiated to read and then write the operand specified by the operand address. If bit 30 of the operand is 0, the Next Instruction (NI) is skipped. The write portion of the storage cycle included writing 01_8 in bits 35 through 30 of storage location U. If bit 30 of the operand is 1, the NI is performed. Bits 29 through 0 at location U are neither examined nor altered.

1. If $U < 200_8$, always execute NI.

9.13.7. Test and Clear and Skip - TCS 73,17; a = 02

If $(U)_{30} = 0$, performs NI;
if $(U)_{30} = 1$, skips NI;
and clears $(U)_{35-30}$;
$(U)_{29-0}$ unchanged

A storage cycle is initiated to read and then write the operand specified by the operand address. If bit 30 of the operand is 0, the Next Instruction (NI) is performed. If bit 30 of the operand is 1, the NI is skipped. The write portion of the storage cycle includes writing 0's in bit 35 through 30 of storage location U. Bits 29 through 0 at location U are neither examined nor altered.

1. If $(U) < 0200$, always execute NI.

9.13.8. No Operation - NOP 74,06

Proceeds to next instruction

The NOP instruction ensures that there is an interval between the end of the instruction that precedes it and the start of the one that follows it.

1. The contents of the a-field is ignored.
2. The only effects that the values in the x-, h-, i-, and u-fields can have on the operation is the index register incrementation obtained when $x \neq 0$ and $h = 1$, and the indirect addressing delay introduced when $i = 1$ and the Relocation and Storage Suppression designator (D7) is 0.

9.13.9. Store Register Set - SRS 72,16

Transfers GRS areas defined in A_a to consecutive storage starting at address U

A_a contains an address and count for each of two GRS areas. These areas are stored consecutively, starting at the location specified by the operand address of the instruction. If either or both count values are 0, no transfer occurs to the respective areas. Relative addresses less than 0200 are treated as storage addresses, not GRS addresses.

The following is the format of A_a for the SRS instruction:

0 0	Area 2 Count	0 0	Area 2 Address	0 0	Area 1 Count	0 0	Area 1 Address
35 34	33 27	26 25	24 18	17 16	15 9	8 7	6 0

9.13.10. Load Register Set - LRS 72,17

Transfers from consecutive storage, starting at location U, to GRS areas defined in A_a

The format of A_a and the operation of the LRS instruction are like that of SRS, except that information is transferred from the location specified by the operand address to the area specified by A_a. Relative addresses less than 0200 are treated as storage addresses, not GRS addresses.

9.13.11. Test Relative Address - TRA 72,15

Determines if a relative address specified in A_a is within a given relative addressing range

The TRA instruction provides a means to determine whether a specific relative address is within a given relative addressing range. The operand address is the first word of a 4-word packet defining an addressing environment used to test the relative address. The packet contains a designator register, a bank descriptor table pointer, four bank descriptor indexes, and four E bits, in the following format:

Word 0	Designator Register					
Word 1	Bank Descriptor Table Pointer					
Word 2	E 0	ignored	BDI 0	E 2	ignored	BDI 2
Word 3	E 1	ignored	BDI 1	E 3	ignored	BDI 3
	35	34 30	29 18	17	16 12	11 0

The relative address to be tested is contained in $X_{a\ 17-0}$. This relative address is translated into an absolute address within the addressing environment specified by the above packet. Relative addresses less than 200_8 are treated as storage addresses, not GRS addresses. The four E-bits within the packet determine whether the Bank Descriptor Table (BDT) pointer in the packet (E is 0) or the EXEC BDT pointer (E is 1) contained in GRS location 040 is used to reference the respective bank descriptor. The TRA ignores the EXEC Bank Descriptor Table Pointer Enable designator (D19), does not check table length violations, does not cause Module Select Register (MSR) basing, and will not produce a Guard Mode interrupt as a result of a relative address out of limits.

To determine the order of descriptor usage during testing, the BDR Selector designator (D12) is used. When D12 is 1, the order is 1, 3, 0, 2. When D12 is 0, the order is 0, 2, 1, 3. If U = 044, the D12 value is obtained from the current D12 in the hardware designator register if the address for the designator register, is 044 (U = 044); otherwise, D12 is obtained from the designator register in the packet.

The results of this instruction are stored in X_a and indicated by skip or no skip. If the relative address tested is within limits, the number of the bank descriptor register, within whose limits the relative address exists, is stored in $X_{a\ 34-33}$, and the absolute address produced is stored in $X_{a\ 23-00}$. If the relative address does not fall within any limits, X_a is cleared to 0 and the Next Instruction (NI) is executed. If the relative address tested is within limits, the write protect bit of the bank descriptor within whose limits the relative address exists is tested. If it is 0, the NI is skipped; if it is 1, the NI is executed.

9.13.12. Increase Instructions - XX 05; a = 10–17

The operand specified by the operand address is transferred under j-field control to the arithmetic section, increased by a value specified by the a-field control to the arithmetic section, and stored under j-field control in the location specified by the operand address; the operation is performed under storage lock (test and set). If the initial operand or the result is 0, the Next Instruction (NI) is executed; otherwise, the NI is skipped. The following values may be selected by the a-field:

mnemonic	a-field	increase value
INC	10	Increases operand by 1. If initial operand or result is 0, execute NI; if not 0, skip NI.
DEC	11	Decreases operand by 1. If initial operand or result is 0, execute NI; if not 0, skip NI.
INC2	12	Increases operand by 2. If initial operand or result is 0, execute NI; if not 0, skip NI.
DEC2	13	Decreases operand by 2. If initial operand or result is 0, execute NI; if not 0, skip NI.
ENZ	14–17	Increases operand by 0. If initial operand or result is 0 execute NI; if not 0, skip NI.

The increase and zero test operations depend on the j-field values to some degree. Certain j-field values extend or interpret the sign of the operand (W, XH1–XH2, T1–T3); for these values, the increase is a ones complement, sign-extended operation, and either +0 or -0 satisfies the zero test. The remaining j-field values do not consider the sign of the operand (H1–H2, Q1–Q4, S1–S6); for these values, the increase is a twos complement, field-size operation, and only +0 satisfies the zero test. If U is less than 0200, a full 36-bit operand is used, and the j-field determines whether a ones complement or twos complement operation is performed and whether -0 also satisfies the zero test.

9.14. Optional Extended Instruction Set

The extended instruction set contains twenty instructions. The instructions move and compare character strings, execute decimal arithmetic, edit decimal strings, and convert data to and from the binary decimal and character formats.

Normal index register incrementation is supported on the extended instruction set, except for Bit Move with Translation and Control (BMTC), Bit Move Long (BIML), and Bit Compare Long (BICL). Indirect addressing is supported on all twenty instructions.

If an interrupt occurs before the bit count has decreased to 0 for the BIML and BMTC instructions, the remnant bit count is stored in the R1 register. The current parameters of the source and destination index registers and of the control string index register (BMTC only) are restored to the respective index registers. When the interrupt is honored, the captured P value is the address of the BIML, BICL, or BMTC instruction. When the interrupt has been processed, it is possible to return and continue to execute the instruction at the point where it was terminated for the interrupt. If the BIML, BICL, or BMTC instructions were entered by means of an Execute instruction, the h-field of the Execute instruction must be 0 so that, when the program returns to the Execute instruction, the effective U address of the Execute instruction will lead to the BIML, BICL, or BMTC instructions. The BIML, BICL, or BMTC instruction h-field must be 0 to enable the program to return to the same effective U address of the first descriptor word and complete the instructions in the event of an interrupt. Results are unpredictable if the h-field equals 1.

When using the Bit Compare (BIC), Bit Move (BIM), BIML, BMTC, BICL, and Store A Quarter Word (SAQW) instructions, software must provide protection on word boundaries. For example, if a BMTC instruction modified bits 23–19 of a word, software must protect all 36 bits of the word because the hardware modifies the word by a read, merge, write operation without storage lock.

If U < 0200 occurs for any of the addresses generated by the following instructions, storage is referenced rather than GRS:

- BIt Move (BIM)
- BIt Compare (BIC)
- Bit Move with Translation and Control (BMTC)
- BIt Compare Long (BICL)
- BIt Move Long (BIML)
- Byte to DEcimal (BDE)
- DEcimal to Byte (DEB)
- EDit DEcimal (EDDE)
- Load A Quarter Word (LAQW)
- Store A Quarter Word (SAQW)

If the initial bit count for the BIM, BIML, BIC, BICL, or BMTC instructions is equal to 0, these instructions function as NOPs. That is, the 0 bit count is detected and execution of the instruction is terminated.

9.14.1. Bit Move – BIM 37,10

The Bit Move (BIM) instruction moves a source string of bits that starts on any boundary to a destination string that also starts on any bit boundary.

The BIM instruction requires the following parameters:

1. Initial source word address.
2. Starting bit address for source string.
3. Initial destination word address.
4. Starting bit address for destination string.
5. Length of string bits.

The operand address specifies a two-word descriptor that contains the instruction parameters. The BIM instruction formats for the descriptor word are:

Descriptor Word 1

Not Used	SSB	DSB	X_d	Not Used	Relative Destination Address (Ud)
35 34	33 28	27 22	21 18	17 16	15 0

where:

Bits 35–34	Not used.
Bits 33–28	Source Start Bit (SSB) specifies the starting bit of the source string. Bits 33-28 = 0 specifies bit 35 as the starting bit, and bits 33-28 = 35_{10} specifies bit 0 as the starting bit.
Bits 27–22	Destination Start Bit (DSB) specifies the starting bit of the destination string. Bits 27–22 = 0 specifies bit 35 as the starting bit, and bits 27–22 = 35_{10} specifies bit 0 as the starting bit.
Bits 21–18	Destination index register (X_d) in GRS is not modified.
Bits 15–0	Relative Destination Address (RDA) = $U_d + X_d$. Modification equals current relative destination address + 1.

Descriptor Word 2

Not Used	Bit Count	X_s	*	-	Relative Source Address (U_s)
35 33	32 22	21 18	17	16	15 0

where:

Bits 35–33	Not used.
Bits 32–22	Bit count specifies the number of bits to be moved.
Bits 21–18	Source index register (X_s) in GRS is not modified.
Bit 17	* Specifies modification of source address.
Bits 15–0	Relative Source Address = $U_s + X_s$. Modification equals current relative source address + h-field. If h = 1, BIM references sequential addresses for source data until the bit count expires. If h = 0, BIM continually references relative source address $U_s + X_s$ until the bit count expires.

All unused fields should be zero filled. If the value in either the SSB field or DSB field is greater than 35_{10} (43_8), the instruction results are unpredictable.

Any overlapping of source strings or descriptor words by the destination strings is not supported and will have unpredictable results.

Source index register equal to the destination index register is supported. When this condition occurs, sequential source string locations are moved to sequential destination string locations.

9.14.2. Bit Compare - BIC 37,11

The Bit Compare (BIC) instruction compares a source string of bits to a destination string of bits. The length of the strings can vary from one bit to 2047 bits. If the two strings are equal, the Overflow designator (D1) is set and the Carry designator (DO) is cleared. If the source string is less than the destination string, the overflow designator is cleared and the carry designator is cleared. If the source string is greater than the destination string, the overflow designator is cleared and the carry designator is set.

Carry (D0)	Overflow (D1)	
0	0	Source String < Destination String
0	1	Source String = Destination String
1	0	Source String > Destination String
1	1	Undefined

If the bit count field in descriptor word 2 is 0, the instruction is terminated with D1 set to 1 and D0 set to 0 to indicate that the source is the same as the destination string.

The BIC instruction requires the following parameters:

1. Initial source word address.
2. Starting bit address for source string.
3. Initial destination word address.
4. Starting bit address for destination string.
5. Length of string in bits.

The operand address specifies a two-word descriptor that contains the instruction parameters. The BIC instruction format for the descriptor words are:

Descriptor Word 1

Not Used	SSB	DSB	X_d	Not Used	Relative Destination Address (U_d)
35 34	33 28	27 22	21 18	17 16	15 0

where:

Bits 35–34	Not used.
Bits 33–28	Source Start Bit (SSB) specifies the starting bit of the source string. Bits 33–28 = 0 specifies bit 35 as the source start bit and bits 33–28 = 35_{10} specifies bit 0 as the source start bit.

Bits 27–22	Destination Start Bit (DSB) specifies the starting bit of the destination string. Bits 27–22 = 0 specifies bit 35 as the destination start bit and bits 27–22 = 35_{10} specifies bit 0 as the destination start bit.
Bits 21–18	Destination index register (X_d) in GRS is not modified.
Bits 15–0	Relative Destination Address = $U_d + X_d$. Modification equals current relative destination address + 1.

Descriptor Word 2

Not Used	Bit Count	X_s	*	–	Relative Source Address (U_s)
35 33	32 22	21 18	17	16	15 0

where:

Bits 35–33	Not used.
Bits 32–22	Bit Count specifies the number of bits to be compared.
Bits 21–18	Source index register (X_s) in GRS is not modified.
Bit 17	* Specifies modification of source address.
Bits 15–0	Relative Source Address = $U_s + X_s$. Modification equals current relative source address + h-field. If h = 1, BIC references sequential addresses for the source data until the bit count expires. If h = 0, BIC continually references relative source address $U_s + X_s$ until the bit count expires.

All unused fields should be zero filled. If the value in either the SSB field or DSB field is greater than 35_{10} (43_8), the instruction results are unpredictable.

Overlapping of source and destination strings is supported.

Source index register equal to the destination index register is supported. When this condition occurs, sequential string locations are compared to sequential destination string locations.

9.14.3. Extended Bit Instructions

The extended bit instructions (Bit Move with Translation and Control, Bit Compare Long, and Bit Move Long) are both versatile and have the functional capabilities for moving and comparing strings of variable length and variable starting bit boundaries.

9.14.3.1. Bit Move with Translation and Control – BMTC 37,12

The Bit Move with Translation and Control (BMTC) instruction moves a source string of characters to a destination string of characters. The character size of both the source and destination strings is variable from one to nine bits. Each source character can be translated with a translation table of 9-bit characters. In addition, the operation may be directed by a control string.

The following parameters are required:

1. Initial source word address.
2. Starting bit for source string.
3. Character size of source string.
4. Initial destination word address.
5. Starting bit for destination string.
6. Character size of destination string.
7. Length of destination string in bits.
8. Starting address of the translation table.
9. Starting address of the control string.

The format of the instruction is similar to the format of the BIM instruction. The operand address specifies a two-word descriptor that contains part of the instruction parameters. The destination character size, source character size, source relative address, destination relative address, and index register selection for control and translation are specified by the two-word descriptor.

The source start bit is specified by bits 35–30 of the source index register, and the destination start bit is specified by bits 35–30 of the destination index register.

Upon detection of an interrupt, the current values of the source start bit, source relative address, destination start bit, and destination relative address are stored in the source and destination index registers. If control is enabled, the current value of the control string address is stored in the control string X-register upon completion of the instruction or detection of an interrupt.

Descriptor Word 1

Not Used	DCS	Control Base X_c	X_d	–	–	Relative Destination Address (U_d)
35 30	29 26	25 22	21 18	17	16	15 0

where:

Bits 35–30	Not Used.
Bits 29–26	Destination Character Size (DCS).
Bits 25–22	Control base index register (X_c). When X_c is 0, control is not enabled.
Bits 21–18	Destination index register (X_d).
Bits 17–16	Not used.
Bits 15–0	Destination Word Relative Address = $U_d + X_D$. Modification equals current relative address +1.

For each 36-bit transfer, $(X_d)_{17-0} + 1 \rightarrow (X_d)_{17-0}$.

Descriptor Word 2

Not Used	SCS	X_t	X_s	*		Relative Source Address (U_s)
35 30	29 26	25 22	21 18	17	16	15 0

where:

Bits 35–30	Not used.
Bits 29–26	Source Character Size (SCS).
Bits 25–22	Translation base index register (X_t). When X_t is 0, translation is not enabled.
Bits 21–18	Source index register (X_s).
Bit 17	* Specifies modification of source string index register.
Bit 16	Not used.
Bits 15–0	Source Word Relative Address = $U_s + X_s$. Modification equals current relative address + h-field. If h = 1, BMTC references sequential address for source data until the bit count expires. If h = 0, BMTC continually references relative source address $U_s + X_s$ until the bit count expires.

For each 36-bit transfer, $(X_s)_{17-0} + h \rightarrow (X_s)_{17-0}$ and $(X_s)_{35-30}$ + remainder bits $\rightarrow (X_s)_{35-30}$.

R1 Register

Reserved	Destination Bit Count
35 24	23 0

where:

Bits 35–24	Not used.
Bits 23–0	Bit Count = R1 register bits.

Destination String Index Register

DSB	Reserved (must be zero)	Destination String Index
35 30	29 18	17 0

where:

Bits 35–30	Destination Start Bit (DSB).

Source String Index Register

SSB	Reserved (must be zero)	Source String Index
35 30	29 18	17 0

where:

Bits 35–30 Source Start Bit (SSB).

Translation Table Index Register

Not Used	Reserved (must be zero)	Translation String Index
35 24	23 18	17 0

Control String Index Register

Not Used	CSC	Reserved (must be zero)	Control String Index
35 32	31 30	29 18	17 0

where:

Bits 31–30 Control Start Character (CSC) selects which quarter word is the first control character.

Bits 31–30	Quarter Word Selected
00	Bits 35–27
01	Bits 26–18
10	Bits 17–9
11	Bits 8–0

The control string is in a packed quarter-word format (35–27, 26–18, 17–9, 8–0). Each control character is a syllable that identifies a particular operation to be executed.

Bits 8–6 of each control character specify the function to be performed, and bits 5–0 specify a repeat count (N). The following functions have been defined.

1. Move (bits 8–6 = 01_8): Move N characters from the source string to the destination string.

2. Skip in Source (bits 8–6 = 02_8): Skip N characters in the source string.

3. Move Without Translation (bits 8–6 = 03_8): Move N characters from the source string to the destination string without translation, even when translation is enabled.

4. Insert (bits 8–6 = 04_8): Insert N characters from the control string into the destination string. The N characters immediately follow the "Insert" control character. The N characters from the control string are not translated.

5. Compare and Move (bits 8–6 = 05_8): Move N characters from the source string to the destination string. If one of the source characters equals the character immediately following the control character, execution of the instruction is halted, the Overflow designator (D1) is set, and the execution of the next instruction is begun. If a comparison is not made, D1 is cleared. If translation is specified, the comparison is executed after translation of the source character, but before truncation when truncation is necessary. The Carry designator (DO) is unconditionally cleared when using the compare function.

6. Skip in Destination (bits 8–6 = 06_8): Skip N characters in the destination string.

7. Scan Equal (bits 8–6 = 07_8): Scan N characters from the source string without moving them to the destination string. If one of the source characters does equal the character immediately following the control character, execution of the instruction is halted, D1 is set, and the execution of the next instruction is begun. If translation is specified, the comparison is executed after translation of the source character. The D0 is unconditionally cleared when using the scan equal function.

8. Stop (bits 8–0 = 0_8): Execution of the instruction is halted, and execution of the next instruction is initiated.

The character size of the translation table is nine bits, and the translation table is in a packed quarter-word format. The least significant character of the translation table is stored in bits 35–27 of the word addressed by the base address of the translation table. The next least significant character of the translation table would be stored in bits 26–18 of the same word.

If the source character size is greater than the destination character size, the high-order bits of the source character are truncated. If the source character size is less than the destination character size, the source character is extended with high-order 0's. The truncation and extension of source characters is done after translation when translation is specified.

The Carry designator (D0) and Overflow designator (D1) are cleared to 0 at instruction initiation and updated during execution only by compare-and-move function codes and scan-equal function codes.

If R1 = 0 when the instruction is initiated, the instruction is terminated with D1 and D0 cleared to 0.

All unused fields should be zero filled. If the value in either the SSB field or DSB is greater than 35_{10} (43_8), the instruction results are unpredictable.

Any overlapping of source, destination, control strings, descriptor words, or index registers is not supported and will have unpredictable results.

The control string is retrieved from storage. References to the translation table are directed to storage.

If the source character size field or destination character size field equals 0 or any value greater than 11_8, the instruction results are unpredictable.

If the character start bit and if the character size of the string combine so that a character within the string is divided by a word boundary, the execution of the instruction will result in an illegal operation. Stores may result before the illegal operation is detected.

If the source index field (X_s) or destination index field (X_d) equals 0, the instruction results are unpredictable. If the bit count does not coincide with a destination character boundary, the instruction results are unpredictable.

If the bit count expires or if a compare is made (which terminates the instruction) during a compare and move and scan equal subfunction, then source index registers, destination index registers, control base registers, and the bit count are updated to reflect the current values at the point of termination.

9.14.3.2. Bit Compare Long - BICL 37,13

The Bit Compare Long (BICL) instruction compares a source string of characters to a destination string.

The following parameters are required:

- Initial source word address.
- Starting bit for source string.
- Initial destination word address.
- Starting bit for destination string.
- Length of string in bits.

The format of this instruction is similar to the format of the BIC instruction. The operand address specifies a two-word descriptor that contains part of the instruction parameters. The source relative address and destination relative address are specified by the two-word descriptor.

The source start bit is specified by bits 35-30 of the source index register, and the destination start bit is specified by bits 35-30 of the destination index register.

Upon detection of an interrupt, the current values of the source start bit, source relative address, destination start bit, destination relative address, and remnant bit count are stored in the source index register, destination index register, and register R1.

Descriptor Word 1

Not Used	X_d	-	-	Relative Destination Address (U_d)
35 22	21 18	17	16	15 0

where:

Bits 35-22	Not used.
Bits 21-18	Destination index register (X_d).
Bits 17-16	Not used.
Bits 15-0	Relative destination address (U_d). Destination Word Address = $U_d + X_d$. Modification equals current relative address + 1.

When the bit count has expired and the source string equals the destination string, the current values of the source start bit, source relative address, destination start bit, destination relative address, and bit count equal 0 are stored in the source index register, destination index register, and register R1. If instruction execution is terminated because of a miscompare of the source, destination strings, the contents of the source index register, destination index register, and register R1 are undefined.

Descriptor Word 2

Not Used	X_s	*	-	Relative Source Address (U_s)
35 22	21 18	17	16	15 0

where:

Bits 35–22	Not Used.
Bits 21–18	Source index register (X_s).
Bit 17	* Specifies modification of source address.
Bit 16	Not used.
Bits 15–0	Source Word Relative Address = $U_s + X_s$. Modification equals current relative address + h-field. If h = 1, BICL references sequential address for source data until bit count expires. If h = 0, BICL continually references relative source address $U_s + X_s$ until bit count expires.

Source String Index Register

SSB	Reserved (must be zero)	Source String Index
35 30	29 18	17 0

where:

Bits 35–30	Source Start Bit (SSB).

Destination String Index Register

DSB	Reserved (must be zero)	Destination String Index
35 30	29 18	17 0

where:

Bits 35–30	Destination Start Bit (DSB).

R1 Register

Reserved	Destination Bit Count
35 24	23 0

If the two strings are equal, the overflow designator (D1) is set and the carry designator (D0) is cleared. If the source string is less than the destination string, D0 and D1 are cleared. If the source string is greater than the destination string, D1 is cleared and D0 is set.

Carry (D0)	Overflow (D1)	
0	0	Source String < Destination String
0	1	Source String = Destination String
1	0	Source String > Destination String
1	1	Not Used

If R1 = 0 at instruction initiation, the instruction is terminated with D1 set to 1 and D0 set to 0 to indicate that the source is the same as the destination string.

Overlapping of source and destination string is supported. If the value in either the SSB field or DSB field is greater than 35_{10} (43_8), the instruction results are unpredictable. If the source index or destination index fields equal 0, or if the source index register equals the destination index register, the instruction results are unpredictable.

9.14.3.3. Bit Move Long – BIML 37,14

The Bit Move Long (BIML) instruction moves a source string of bits that starts on any bit boundary to a destination that also starts on any bit boundary.

The following parameters are required:

- Initial source word address.
- Starting bit for source string.
- Initial destination word address.
- Starting bit for destination string.
- Length of destination string in bits.

The format of this instruction is similar to the format of the BMTC instruction. The operand address specifies a 2-word descriptor that contains part of the instruction parameters. The source relative address, source index, destination relative address, and destination index are specified by the 2-word descriptor.

The source start bit is specified by bits 35–30 of the source index register, and the destination start bit is specified by bits 35–30 of the destination index register.

Upon completion of this instruction or detection of an interrupt, the current values of the source start bit, source relative address, destination start bit, destination relative address, and bit count are stored in the source and destination index registers and register R1.

Descriptor Word 1

Not Used	Destination Index X_d	–	–	Relative Destination Address (U_d)
35 — 22	21 — 18	17	16	15 — 0

For each destination address read, $X_{d17-0} + 1 \rightarrow X_{d17-0}$.

Descriptor Word 2

Not Used	Source Index X_s	*	–	Relative Source Address (U_s)
35 — 22	21 — 18	17	16	15 — 0

* *Specifies modification of source address ($U_s + X_s + h$)*

For each source address read, $X_{s\ 17-0} + h \rightarrow X_{s\ 17-0}$ and update bit offset $\rightarrow X_{s\ 35-30}$.

Source String Index Register

SSB	Reserved (must be zero)	Source String Index
35 30	29 18	17 0

where:

Bits 35–30 Source Start Bit (SSB).

Destination String Index Register

DSB	Reserved (must be zero)	Destination String Index
35 30	29 18	17 0

where:

Bits 35–30 Destination Start Bit (DSB).

R1 Register

Not Used	Destination Bit Count
35 24	23 0

where:

Bits 23–0 Destination bit count.

All unused fields should be zero filled. If the values in either the SSB or DSB field are greater than 35_{10} (43_8), the instruction results are unpredictable. Any overlapping of the source string, destination string, or descriptor words is not supported and will have unpredictable results. If the source index or destination index field equals 0, or if the source index register equals the destination index register, the instruction results are unpredictable.

9.14.4. Byte to Decimal – BDE 37,15

The Byte to DEcimal (BDE) instruction converts a string of either ASCII or External Comp 3 (the low-order eight bits of each nine-bit character contains two decimal digits) characters to a signed magnitude format. The result is placed in one, two, or three A-registers (A_a, A_{a+1}, and A_{a+2}). The result can also be scaled to the left.

The sign character can be detected in several different places within the numeric string. The following options are allowed:

- No sign character.

- A leading separate sign character. The first character contains the sign (ASCII only).
- A trailing separate sign character. The last character contains the sign.
- A leading included sign character. The first character contains the sign and a digit (ASCII only).
- A trailing included sign character. The last character contains the sign and a digit (ASCII only).

The BDE instruction requires the following parameters:

- Initial source string word address.
- Starting character address for source string.
- A-register(s) for storing the result.
- Length of string.
- Location of the sign field.
- ASCII or External Comp 3 specification.
- Skip count - the number of leading 0's to be placed.

The parameters for the BDE instruction are specified by the instruction and by the parameter A_a-register (A_a).

Parameter A_a-Register

SSCL	*	DAR	*	SSC	*	CF	SL	A	SC	*	DC
35	34 26	25 22	21 20	19 18	17	16	15 13	12 11	10 6	5	4 0

where:

Bit 35 Source Start Character Location (SSCL)

= 0: Bits 19–18 of the parameter A_a-register specify the source start character.

= 1: Bits 31–30 of the source index register specify the source start character. Bits 35–32 are ignored.

Bits 34–26 * Not Used.

Bits 25–22 Destination A-Register (DAR) specifies the location of the A-registers.

Bits 21–20 * Not Used.

Bits 19–18 Source Start Character (SSC)

= 0: Starting character is in bits 35–27.
= 1: Starting character is in bits 26–18.
= 2: Starting character is in bits 17–9.
= 3: Starting character is in bits 8–0.

Bit 17 * Not Used.

Bit 16 Character Format (CF)

= 0: specifies ASCII characters.
= 1: specifies External Comp 3 characters.

Bits 15-13	Sign Location (SL) within a string ASCII: = 0:, 1:, 2:, 3: No sign = 7: Leading separate sign = 6: Leading included sign = 5: Trailing separate sign = 4: Trailing included sign External Comp 3: = 0:, 1:, 2:, 3: No sign = 4:, 5:, 6:, 7: Trailing separate sign
Bits 12-11	Bits 12-11 of the parameter A_a-register (A_a) specify the number of A-registers that are to be loaded. If this field is 0, the DARs specified are not modified. If the skip count and digit count do not fill the specified A-registers, the remaining digits are filled with 0's. If the skip count plus digit count exceeds the number of A-registers specified, the excess source data is truncated and the transferred data plus sign is stored in the A-registers specified. If the skip count alone exceeds the number of A-registers specified, the A-registers are filled with 0's plus a positive sign.
Bits 10-6	Skip Count (SC) specifies the number of leading 0's to be inserted in front of the most significant destination digit.
Bit 5	* Not used.
Bits 4-0	Digit Count (DC) specifies the number of 9-bit characters to be referenced in the ASCII format. For the External Comp 3 format, it specifies the number of 4-bit characters to be used. The digit count does include the sign character. If the digit count is 0, the A-registers specified are filled with 0's plus a positive sign character. All unused fields in the parameter A-register should be zero filled to allow for future expansion.

9.14.5. Decimal to Byte - DEB 37,16

The DEcimal to Byte (DEB) instruction converts a string of decimal characters in a Binary Coded Decimal (BCD) signed magnitude format to either an ASCII or External Comp 3 format. The source packed decimal string is located in one, two, or three A-registers (A_a, A_{a+1}, A_{a+2}). The source string that starts with any decimal digit is transmitted to the destination string. The destination string is specified by the operand address ($U = u + X_m$); operand addresses less than 200 reference storage locations, not GRS locations.

The sign character is determined from bits 3-0 of the least significant A-register. If bits 3-0 equal 00_8, 02_8, 04_8, 06_8, 10_8, 12_8, 14_8, 16_8, or 17_8, the sign is positive. If bits 3-0 equal 01_8, 03_8, 05_8, 07_8, 11_8, 13_8, or 15_8, the sign is negative.

The following destination string options are allowed:

- No sign character.
- A leading separate sign character.
- A trailing separate sign character.
- A leading included sign character.
- A trailing included sign character.

A Decimal to Byte instruction requires the following parameters:

- Starting word address for destination string.
- Starting character address for destination string.
- A-register(s) for source string.
- Length of string.
- Location of the sign character.
- ASCII or External Comp 3 specification.
- Skip count – the number of leading decimal digits.

The parameters for the DEB instruction are specified by the instruction and by the parameter A_a–register (A_a).

Parameter A_a-Register

DSCL	*	SAR	*	DCS	*	CF	SL	A	SC	*	DC
35	34 26	25 22	21 20	19 18	17	16	15 13	12 11	10 6	5	4 0

where:

Bit 35 Destination Start Character Location (DSCL)

= 0: Bits 19–18 of the parameter A_a–register specify the destination start character.

= 1: Bits 31–30 of the destination index register specify the destination start character;

Bits 34–26 * Not used.

Bits 25–22 Source A–Register (SAR) specifies the location of the source A–registers.

Bits 21–20 * Not used.

Bits 19–18 Destination Start Character (DSC)

= 0: Starting character is in bits 35–27
= 1: Starting character is in bits 26–18
= 2: Starting character is in bits 17–9
= 3: Starting character is in bits 8–0

Bit 17 * Not used.

Bit 16	Character Format (CF) = 0: Specifies ASCII characters = 1: Specifies External Comp 3 characters
Bits 15–13	Sign Location (SL) within string ASCII: = 0:, 1:, 2:, 3: No sign = 7: Leading separate sign = 6: Leading included sign = 5: Trailing separate sign = 4: Trailing included sign External Comp 3: = 0:, 1:, 2:, 3: No sign = 4:, 5:, 6:, 7: Trailing separate sign
Bits 12–11	Bits 12–11 of the parameter A_a-register (A_a) specify the number of A–registers to be used. If this field is 0, the destination string specified is filled with 0's plus a positive sign. If the skip count plus the digit count requires more digits than are in the specified A–registers, the remaining digits are filled with 0's. If the skip count alone equals or exceeds the number of decimal digits specified in the A–registers, the destination string specified is filled with 0's plus a positive sign.
Bits 10–6	Skip Count (SC) specifies the number of leading decimal digits to be skipped.
Bit 5	* Not used.
Bits 4–0	For ASCII format, the Digit Count (DC) specifies the number of 9–bit destination characters to be referenced; it includes the sign characters for External Comp 3 format. The digit count specifies the number of 4–bit decimal digits to be stored in the destination string; it includes the sign digit. In separate sign format, the digit count is equal to the number of destination digits plus the sign. In included sign format, the digit count is equal to the number of destination digits where either the leading or trailing digit is combined with the sign character. If this field is 0, the destination string specified is not modified. All unused fields should be zero filled to allow for future expansion.

9.14.6. Edit Decimal – EDDE 37,17

The EDit DEcimal (EDDE) instruction moves a source string to a destination string. The operation is directed by a control string. The source string is in the binary–coded decimal–signed magnitude format in A–registers, and the destination string is in an ASCII format. The control string provides for blanking out leading 0's, inserting commas, decimal points, currency signs, and inserting plus and minus signs.

The Edit Decimal instruction requires the following parameters.

- Source string - A-registers in packed decimal format.
- Starting word address for destination string.
- Starting character address for destination string.
- Digit count.
- Starting word address for control string.
  Source String - 4-bit binary coded decimal characters
  Destination String - 9-bit ASCII characters
  Control String - 9-bit characters

Descriptor Word 1

A	SC	DC	CSC	X_c	*	U_c
35 34	33 29	28 24	23 22	21 18	17 16	15 0

where:

Bits 35-34	A-registers required (A) specifies the number of A-registers required to hold the decimal source string.
Bits 33-29	Skip Count (SC) specifies the number of leading decimal digits to be skipped.
Bits 28-24	Digit Count (DC) specifies the number of digits in the source field.
Bits 23-22	Control Start Character (CSC) specifies the location of the first control character as follows:

= 0: Starting control character is in bits 35-27.
= 1: Starting control character is in bits 26-18.
= 2: Starting control character is in bits 17-9.
= 3: Starting control character is in bits 8-0.

Bits 21-18	Control string index register (X_c).
Bits 17-16	Not used (*).
Bits 15-0	Relative control string address (U_c).

Descriptor Word 2

*	DSCL	DSC	X_d	*	U_d
35 25	24	23 22	21 18	17 16	15 0

where:

Bits 35-25	Not used (*). Bits 35-32 of the destination index register are ignored and need not be 0.

Bit 24	Destination Start Character Location (DSCL). If bit 24 is 0, bits 23-22 of descriptor word 2 specify the destination start character. If bit 24 is 1, bits 31-30 of the destination index register specify the destination start character.
Bits 23-22	Destination Start Character (DSC) specifies the destination of the first destination character as follows: = 0: Starting destination character is in bits 35-27. = 1: Starting destination character is in bits 26-18. = 2: Starting destination character is in bits 17-9. = 3: Starting destination character is in bits 8-0.
Bits 21-18	Destination index register (X_d).
Bits 17-16	Not used (*).
Bits 15-0	Relative destination address (U_d).

All unused fields in the instruction and descriptor words should be zero filled to allow for future expansion.

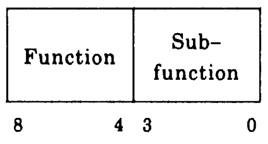
As each decimal digit is moved from the source string to the destination string, the digit is converted to an ASCII character. Characters moved from the control string to the destination string are not modified.

The Edit Decimal instruction has a syllable control string. Each 9-bit control character is a syllable that defines a particular operation to be executed. If an invalid or illegal control function is encountered, an Invalid Instruction interrupt is generated.

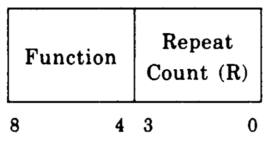
Although a Guard Mode interrupt is generated if 256 storage references are made during the Edit instruction, the interrupt is not handled until completion of the current control syllable. This allows additional storage references to be made after 256 references to a maximum of 266 references.

Edit Control Syllables

The first byte of an Edit Control syllable is a function byte and may be followed by a character string. There are two classes of function bytes with the following formats:



Format 1



Format 2

The following notations are used to describe the edit control characters:

Zero – the source field zero condition exists if all the decimal digits specified by the digit count field are 0 or if the number of A-registers required is 0.

Overflow – source field overflow condition exists if any digit skipped by the skip count is not 0.

During the execution of the Edit instruction, the source field is examined for the 0 and overflow conditions, and the corresponding flags are set or cleared accordingly. Prior to termination of the instruction, Carry designator (D0) is written with the value of the 0 flag and Overflow designator (D1) is written with the value of the overflow flag.

- Format 1 is used for the following function bytes:

Stop Editing 0,0

This control character terminates the Edit instruction.

Set Significance 0,1

If the insertion flag is set and the significance flag is not already set, the insertion character is transferred to the current destination. The significance flag is unconditionally set.

Clear Significance 0,2

The significance flag is unconditionally cleared.

Define Fill Character 0,3 C1

The next character in the control string is used to establish the fill character. If the fill character is used before being established, the default case is an ASCII blank.

Enable Fill if Source Zero 0,4

If the entire source field is 0, set the enable fill flag; otherwise, go to the next control character.

Enable Insertion 0,5 C1

The insertion flag is set after the insertion value is established. If the fill flag is set, the fill character is used for the insertion character. If the fill flag is not set, the next character in the control string is used for the insertion character.

Move First FORTRAN Digit 0,7 C1 C2

This control syllable inspects a single source character after checking overflow and sign. Even if all characters skipped in the A-registers were 0, an overflow could exist for negative source fields that have a nonzero first character.

If overflow is detected, the overflow flag is set, control string character "C2" is used to establish the fill character and "C2" is stored at the destination character address. The enable fill flag is then set. The source character is not transferred and the source character address is advanced by 1. The destination address is advanced by 1, and the control character address is advanced by 3.

If overflow does not exist, the source field has a negative sign, and the source character is not zero, control string character "C2" is used to establish the fill character. This same character is stored at the current destination and the enable fill flag is set. The source character is not transferred and the source character address is advanced by 1. The destination address is advanced by 1, and the control syllable is advanced by 3.

If no overflow exists and the sign is negative, and the source character is 0 the following steps are taken:

If the significance flag is set, transfer the control string character "C1" to the destination character address. The source character is not transferred, and the source character address is advanced 1. The destination address is advanced 1, and the control syllable is advanced by 3.

If the significance flag is clear, the control string character "C1" is used to establish an insertion character and the insertion flag is set. The source character is not transferred, and the source character address is advanced 1. The control syllable address is advanced by 3, but the destination address in not advanced.

If no overflow existed and the sign is positive, the following steps are taken:

If the significance flag is set, the source character is transferred to the destination character address and step 4 is taken.

If the significance flag is clear and the source character is not 0, the significance flag is set and the source character is transferred to the destination character address and step 4 is taken.

If the significance flag is clear and the source character is 0, the fill character is transferred to the destination character address and step 4 is taken.

The destination address is advanced by 1. The source address is advanced by 1. The control syllable is advanced by 3.

Move Character Unconditionally	0,8

The next character in the control string is unconditionally transferred to the current destination.

Move Digit Include Zone Sign	0,9n
Move Digit Include Zone Sign if Positive	0,A
Move Digit Include Zone Sign if Negative	0,B

These commands allow conversion of the Binary Coded Decimal (BCD) source into unsigned ASCII or zoned ASCII that can represent an overpunched positive or negative sign. A zoned sign is included with the ASCII value if the appropriate condition is true. A source character is transferred to a destination character with or without sign included.

Move Character Conditioned by Significance	0,C C1 C2
Move Character Conditioned by Sign	0,D C1 C2
Move Character Conditioned by Zero	0,E C1 C2
Move Character Conditioned by Overflow	0,F C1 C2

If the enable fill flag is set, transfer the fill character to the current destination. If the enable fill flag is not set and the "selected flag" is set, transfer the second character in the control string after the current character to the current destination.

Skip if Significance	1,0 R

If the significance flag is set, skip the next R control characters. The next control character is the current character location plus R plus 2. The 2 accounts for the current character and the character used for R.

If the significance flag is not set, go to the next control character after the character R.

Skip if Negative	1,1 R

If the sign flag is set, skip the next R control characters. The next control character is the current character location plus R plus 2. The 2 accounts for the current character and the character used for R.

If the sign flag is not set, go to the next control character after the character R.

Skip if Zero	1,2 R

If the zero flag is set, skip the next R control characters. The next control character is the current character location plus R plus 2. The 2 accounts for the current character and the character used for R.

If the zero flag is not set, go to the next control character after the character R.

Skip if Overflow	1,3 R

If the overflow flag is set, skip the next R control characters. The next control character is the current character location plus R plus 2. The 2 accounts for the current character and the character used for R.

If the overflow flag is not set, go to the next control character after the character R.

Enable Insertion (Function of Significance)	1,4 C1 C2
Enable Insertion (Function of Sign)	1,5 C1 C2
Enable Insertion (Function of Zero)	1,6 C1 C2
Enable Insertion (function of Overflow)	1,7 C1 C2

The insertion flag is set after the insertion character value is established. If the fill flag is set, the fill character is used to establish the new insertion character. If the fill flag is not set and the "selected flag" is not set, the next character in the control string is used to establish the insertion character. If the fill character is not set and the "selected flag" is set, the second character is used to establish the insertion character.

- Format 2 is used for the following function bytes.

Move Digits	2,R

If the fill flag is not set, transfer the next R source characters, starting with the current source address, to the next R destination locations, starting with the current destination address. If the fill flag is set, use the fill character to transfer into each of the next R destination locations.

Move Digits and Check for Significance	3,R

This control syllable can be used to transfer R source characters to destinations while under flag control. After each source character is inspected or transferred, the source character address is advanced by 1, the destination character address is advanced by 1, and the source count R is internally decremented by 1. The steps below continue until R=0; when R becomes 0, the control syllable address is advanced by 1.

If the enable fill flag is set, each of the R destinations receives the fill character instead of the source characters.

If the enable fill flag is clear and the significance flag is set, the R source characters are transferred to the R destinations.

If the enable fill flag is clear and the significance flag is not set, each source character is inspected sequentially as follows until R is 0.

If the source character is 0, the destination receives a fill character and the source character is not transferred.

If a nonzero source character is detected, the following steps are taken.

If the insertion flag is set, the insertion character is moved to the destination, the destination character address is advanced 1 and the next step is taken.

The significance flag is set.

The source character is moved to the destination address. Source and destination address are now advanced 1.

Once the significance flag is set, all remaining source characters are transferred to the destination addresses.

Move Characters 4,R C1 C2 - - C_R

If the enable fill flag is not set, move the next R characters from the control character string into each of the R locations, starting with the current destination. If the fill flag is set, transfer the fill character R times, not the characters from the control string.

Move Characters Conditioned by Significance 5,R C1 C2 - - C_R

If the significance flag is set, not the fill flag, move the next R characters from the control character string into each of the R locations, starting with the current destination. If the fill flag is set or the significance flag is clear, transfer the fill character R times, not the characters from the control string.

Move Character Repetitively 6,R C1

If the fill flag is not set, move the character following the current control character into each of the R locations, starting with the current destination. If the fill flag is set, transfer the fill character R times, not the next character in the control string.

9.14.7. Decimal Instructions

Four decimal arithmetic instructions, Add DEcimal (ADE), Double Add DEcimal (DADE), Subtract DEcimal (SDE), and Double Subtract DEcimal (DSDE); and four binary/decimal conversion instructions, DEcimal to Integer (DEI), Double DEcimal to Integer (DDEI), Integer to DEcimal (IDE), and Double Integer to DEcimal (DIDE); are provided.

The decimal data format is binary coded decimal and signed magnitude. The decimal digits 0 through 9 are represented by the values 0000_2 through 1001_2. All eight decimal arithmetic instructions accept any value in the sign digit of an input operand. A positive sign is represented by the values 00_8, 02_8, 04_8, 06_8, 10_8, 12_8, 14_8, 16_8, and 17_8. A negative sign is represented by the values 01_8, 03_8, 05_8, 07_8, 11_8, 13_8, and 15_8.

For all instructions requiring results in a decimal format, 14_8 is stored in the sign digit of the result if the result is positive, and 15_8 is stored in the sign digit of the result if the result is negative.

The decimal arithmetic instructions (ADE, DADE, DSDE, SDE) convention for determining the correct sign that is stored in the specified A–register is as follows. When the result is not 0, the correct result is stored. When the result is equal to ±0 i.e., no overflow has occurred, +0 is stored. When overflow occurs and the truncated result is equal to ±0, then a positive or negative sign is stored, respectively.

Single and double word operands have the following format:

Single Word Operand Decimal Data Format

D	D	D	D	D	D	D	D	S
35 32	31 28	27 24	23 20	19 16	15 12	11 8	7 4	3 0

Double Word Operand Decimal Data Format

D	D	D	D	D	D	D	D	S
71 68	67 64	63 60	59 56	55 52	51 48	47 44	43 40	39 36

D	D	D	D	D	D	D	D	S
35 32	31 28	27 24	23 20	19 16	15 12	11 8	7 4	3 0

D = Decimal Digit S = Sign Digit

For the four add and subtract decimal instructions, the carry and overflow designators are handled as follows:

- The Carry designator (DO) is always cleared.
- The Overflow designator (D1) is set if the size of the result is greater than the field size provided for the result.

9.14.7.1. Add Decimal - ADE 07,00

The Add DEcimal (ADE) instruction adds an operand (U) to an A-register and stores the result in the A-register. Both operands have a signed magnitude BCD format. Results are stored even if overflow occurs.

9.14.7.2. Double Add Decimal - DADE 07,01

The Double Add DEcimal (DADE) instruction adds a 2-word operand (U, U+1) to two A-registers (A_a, A_{a+1}) and stores the result in A_a and A_{a+1}. Both operands have a signed magnitude BCD format. Results are stored even if overflow occurs.

9.14.7.3. Subtract Decimal - SDE 07,02

The Subtract DEcimal (SDE) instruction subtracts an operand (U) from an A-register and stores the result in the A-register. Both operands have a signed magnitude BCD format. Results are stored even if overflow occurs.

9.14.7.4. Double Subtract Decimal - DSDE 07,03

The Double Subtract DEcimal (DSDE) instruction subtracts a 2-word operand (U, U+1) from the two A-registers (A_a, A_{a+1}) and stores the result in A_a and A_{a+1}. Both operands have a signed magnitude BCD format. Results are stored even if overflow occurs.

9.14.7.5. Decimal to Integer - DEI 07,06

The DEcimal to Integer (DEI) instruction converts a 1-word operand (U) from a signed magnitude format to a ones complement binary format and stores the result in the specified A-register.

If the binary source is -0, the binary result is +0.

9.14.7.6. Double Decimal to Integer - DDEI 07,07

The Double DEcimal to Integer (DDEI) instruction converts a 2-word operand (U and U+1) from a decimal format (BCD signed magnitude) to a ones complement binary format and stores the result in the specified A-registers (A_a and A_{a+1}).

If the decimal source is -0, the binary result is +0.

9.14.7.7. Integer to Decimal - IDE 07,10

The Integer to DEcimal (IDE) instruction converts a 1-word operand (U) in a ones complement binary format to a decimal format (BCD signed magnitude) and stores the result in a pair of A-registers. The sign bit is retrieved from bit 35 of U. If bit 35 is 0, a positive sign is specified, and if bit 35 is 1, a negative sign is specified.

If the binary source is –0, the decimal result is +0.

9.14.7.8. Double Integer to Decimal - DIDE 07,11

The Double Integer to Decimal instruction converts a 2-word operand (U, U+1) in ones complement binary format to a decimal format (BCD signed magnitude and stores the result in three A-registers (A_a, A_{a+1}, A_{a+2}). The sign bit is retrieved from the bit 35 of U. If bit 35 is 0, a positive sign is specified and if bit 35 is 1, a negative sign is specified.

If the binary source is –0, the decimal result is +0.

9.14.8. Normalize Instructions

The normalize instructions reduce an index register to a standard normalized form to be utilized by the BMTC, BICL, BIML, DEB, BDE, and EDDE instructions.

9.14.8.1. Bit Normalize - BN 72,12

$(36_{10} * X_{a\ 17-0} + X_{a\ 35-30} + \text{signed } U_{35-0})/36_{10} \rightarrow$ result and remainder

Remainder	→	$X_{a\ 35-30}$
Zeros	→	$X_{a\ 29-18}$
Result	→	$X_{a\ 17-0}$

The Bit Normalize (BN) instruction adds the signed bit offset of an operand and the bit and word offset of an X-register and then stores the result in bit normalized form in the X-register. The resultant bit offset and word offset are modified so that the resulting bit offset is less than 36_{10}. The operation is executed as follows:

If the result overflows, the results of this instruction are unpredictable and the designator register will remain unchanged.

9.14.8.2. Byte to Bit Normalize - BBN 72,14

$(36_{10} * X_{a\ 17-0} + 9 * (X_{a\ 31-30} + \text{signed } U_{35-0}))/36_{10} \rightarrow$ result and remainder

Remainder	→	$X_{a\ 35-30}$
Zeros	→	$X_{a\ 29-18}$
Result	→	$X_{a\ 17-0}$

The Byte to Bit Normalize (BBN) instruction adds the signed byte (nine bits offset of an operand and the byte and word offset of an X-register and then stores the result in bit normalized form in the X-register. The resultant bit offset and word offset are modified so that the resulting bit offset is less than 36_{10}. The operation is executed as follows:

If the result overflows, the results of this instruction are unpredictable and the designator register will remain unchanged.

9.14.9. Single Character Instructions

9.14.9.1. Load A Quarter Word - LAQW 07,04

(Selected quarter word) → $A_{a\ 8-0}$;
zeros → $A_{a\ 35-9}$

The Load A-register Quarter Word (LAQW) instruction transfers a single 9-bit character to the selected A_a-register.

Source Character Index - X_x

Ignored	X_x	Reserved (must be zero)	Source Index
35 32	31 30	29 18	17 0

where:

Bits 31-30	Selected quarter word $(X_x)_{31-30}$ specifies the following:
	= 0: character is in bits 35-27
	= 1: character is in bits 26-18
	= 2: character is in bits 17-9
	= 3: character is in bits 8-0
Bits 17-0	Source index U_{17-0} specifies a word address.

Destination A_a-Register

Zero	Quarter Word
35 9	8 0

The quarter word selected by U and X_x bits 31 and 30 is transferred to $A_{a\ 8-0}$. Bits 35-09 of A_a are zero filled.

zeros → $A_{a\ 35-9}$

9.14.9.2. Store A Quarter Word - SAQW 07,05

$(A_a)_{8-0}$ → Selected quarter word.
$(A_a)_{8-0}$ is transferred to the quarter word selected by U and X_x bits 31 and 30.

The Store A-register Quarter Word (SAQW) instruction transfers the $A_{a\ 8-0}$ character to the location specified by U and $X_{x\ 31-30}$.

Destination Character Index - X_x

Ignored	X_x	Reserved (must be zero)	Destination Index
35 32	31 30	29 18	17 0

where:

Bits 31-30	Selected quarter word $(X_x)_{31-30}$ specifies the following: = 0: character is in bits 35-27 = 1: character is in bits 26-18 = 2: character is in bits 17-9 = 3: character is in bits 8-0
Bits 17-0	Destination index (U_{17-0}) specifies a word address.

Source A_a-Register

Zero	Quarter Word
35 9	8 0

9.15. Executive Instructions

The instructions in this group are intended for use by the Executive System. When the Privileged Instruction GRS Protection designator (D2) is 0, the Executive repertoire is selected. This allows execution of all Executive (privileged) instructions in addition to those of the user repertoire. The Executive repertoire includes instructions for control of the processor state, interrupts, input/output, and instrumentation.

The Executive control instructions defined for the Central Processing Unit (CPU) are described in the following paragraphs.

9.15.1. Initiate Maintenance Interrupt - IMI 72,00

The IMI instruction generates an attention interrupt to the System Support Processor (SSP) and provides the operand as a 36-bit function or status word. The format of the 36-bit operand is specified by software. If the system support interrupt mechanism is available, the IMI instruction is executed and the next instruction is skipped. If the SSP has not accepted a previous system support interrupt, the next instruction is executed.

9.15.2. Prevent All Interrupts and Jump - PAIJ 72,13

Prevents all interrupts and jumps to U

The CPU will not recognize certain interrupt requests received after the completion of the PAIJ instruction nor will it react to interrupt requests received after the start of the instruction execution.

The following interrupts may be prevented by the PAIJ instruction:

- All I/O interrupts, including those for Normal Status, Table Status, and Machine Check interrupts.
- Jump History Stack interrupts.
- Interprocessor interrupts.
- Dayclock and Real-Time Clock interrupts.
- Delayed Storage Check interrupts reporting faults detected in a Storage Interface Unit (SIU) or a Main Storage Unit (MSU).
- System Support Processor interrupts.

9.15.3. Load Dayclock - LDC 73,14,10

The LDC instruction causes the internal dayclock register value of the CPU to be replaced with the value in the dayclock location in fixed storage. (See 9.2.2 for a description of the dayclock.)

9.15.4. Diagnostics

9.15.4.1. Microdiagnostic C - MDC 73,14, 16

Used for CPU Fault Injection

The MDC instruction provides a way to inject internal checks during normal program execution to verify the proper functioning of the check injection, check detection, retry hardware, and the fault analysis and retry microcode. The instruction injects the check specified by the MDC control word loaded by software into X_x. When the check is detected, the instruction is retried. During the retry the check is removed, and the delayed check status word describing the failure is loaded into X_x. No delayed check interrupt is generated as a result of the retry. Any internal delayed check interrupt request that is outstanding at the time the instruction is executed, is cleared so that the instruction is executed with external interrupts enabled. (Delayed Check interrupt requests from storage are unaffected.) Any check injection bits set in the hardware state register at the time of execution are cleared. (The instruction can be used with a NOP control word to clear the HSR check injection bits.)

The MDC control word stored in X_x has a 4-bit command code in bits 5-2. Bits 35-6 and 1-0 are not used and may be set to any state. No X incrementation will take place. The following table lists the 4-bit command codes, the check that is injected, and the expected status word returned to X_x.

Command Code	Check injected	Status Word in X_x
0000	NOP	X_x unchanged
0001	NOP	X_x unchanged
0010	NOP	X_x unchanged
0011	LFC Miscompare	512001000004
0100	D-Bus Miscompare	513000000004

0101	Shift PROM Parity	512000400004
0110	Dispatcher Miscompare	512002000004
0111	Shifter Miscompare	512004000004
1000	Shifter Input SLR Parity Upper	512000200004
1001	Shifter Input SLR Parity Lower	512000200004
1010	GRS Parity Upper	512400000004
1011	GRS Parity Lower	512400000004
1100	P1 Local Store Upper	512040000004
1101	P1 Local Store Upper	512020000004
1110	P2 Local Store Upper	512200000004
1111	P2 Local Store Lower	512100000004

If the injected fault is not detected X_x is unchanged. If the wrong fault is injected, the status word in X_x will not match the expected word in the table.

9.15.4.2. Reserved Operation Codes - 73,14, 14, 15, 17

Operation codes 73, 14, a = 14, 15, 17 are reserved for future use. They are executed as NOP instructions.

9.15.5. Select Interrupt Locations - SIL 73,15,00

(U) specifies control parameter and modes of operation to SIU or MSU

The contents of U specifies a maintenance operation action that is recognized either by the SIU or MSU. The SIU passes a maintenance reference to an MSU if an SIU action is not specified. If an MSU action is specified, the address presented by the maintenance determines the MSU that receives the reference. The address presented by the maintenance operation is the operand address, except for maintenance operations that enable/disable single bit errors in the MSU where $(U)_{13-11}$ equals 0_8 or 7_8.

9.15.6. Load Breakpoint Register - LBRX 73,15,02

Transfers operand to Breakpoint Register

The operand specified by the operand address is transferred to the breakpoint register. This establishes the modes of operation for the breakpoint mechanism, and activates and establishes the modes of operation for the jump history stack.

H	S	B	P	R	W	C	Reserved	Absolute Breakpoint Address
35	34	33	32	31	30	29	28 24	23 0

where:

Bit 35 The H-bit specifies that the CPU will stop on a breakpoint match condition; an interrupt request will not be generated. If H is 0, a Breakpoint interrupt will occur on a breakpoint match condition. The H-bit is ignored in real-time mode and does not affect interrupts caused by the jump history stack.

Bit 34	The S-bit specifies that when the jump history stack is full (an entry is made in the last location), entry stacking is disabled, and a Jump History Stack interrupt is generated. If the S-bit is 0, entry stacking wraps around from the last to first location in the stack without causing an interrupt. The S-bit should not be set unless the C-bit is also set.
Bit 33	The B-bit specifies that entry stacking is disabled when any interrupt occurs. If the B-bit is 0, entry stacking is not affected by interrupts, except as provided by the S-bit.
Bit 32	The P-bit allows a breakpoint match to occur on an instruction address produced from a program address register or a jump or an Execute operand instruction.
Bit 31	The R-bit allows a breakpoint match to occur on an operand address during a read operation.
Bit 30	The W-bit allows a breakpoint match to occur on an operand address during a write operation.
Bit 29	The C-bit specifies that the General Register Set (GRS) locations 070–077 be cleared to 0 and sequential jump history stacking begin at 070.
Bits 28–24	Reserved for future use.
Bits 23–0	The absolute breakpoint address is the value compared to the 24-bit absolute instruction or operand address.

9.15.7. Load Quantum Timer – LQT 73,15,03

The full-word operand specified by the operand address is placed in the quantum timer.

9.15.8. Initiate Interprocessor Interrupt – IIIX 73,15,04

The operand address value specifies the number of the CPU to be interrupted. The next instruction is skipped only if an IPI request is made to the designated processor. The request will not be made if the CPUs are in different applications or if there is a previous IPI request from this CPU to the one specified by the IIIX instruction.

9.15.9. Store Processor ID – SPID 73,15,05

The binary serial number is stored in the first third of the operand; the 4-bit binary firmware branch and the 8-bit binary hardware level are stored in the second third of operand; the CPU features provided are stored in the fifth sixth of the operand (bit 11 with the performance monitoring feature and bit 10 with extended instruction set feature); and the binary CPU number is stored in the last sixth of the operand.

Serial Number	Firmware Branch	REV LEV	PM	EI	0	0	0	0	CPU Number
35 24	23 20	19 12	11	10	9	8	7	6	5 0

where:

Bits 35–24	Serial Number specifies the binary serial number.
Bits 23–20	Firmware Branch specifies the binary firmware branch.
Bits 19–12	Revision level (REV LEV) specifies the binary revision level.
Bit 11	Set if Performance Monitor (PM) feature is installed.
Bit 10	Set if Extended Instruction (EI) set feature is installed.
Bits 9–6	Are zeros.
Bits 5–0	The CPU number.

9.15.10. Clear Support Controller – Clear SC 73,15,06

If $U_0 = 1$, clear any outstanding IMI requests to SC;
if $U_0 = 0$, clear SC except for SSP select field

If bit 0 of U is 0, the instruction clears the Support Controller (SC) except for the SSP select field in the SC address register.

If bit 0 of U is 1, the instruction clears any outstanding Initiate Maintenance Interrupt (IMI) requests to the SC.

If a Clear SC command is received by the SC while one of the SSP interfaces is active, a unit check status is presented to the active SSP interface. Bit 1 of the first sense byte is also set.

9.15.11. Load Base – LB 73,15,10

Places operand bits 17–0 in base value field of BDR specified by $X_{x\ 34-33}$

Bits 17 through 0 of the operand specified by the operand address are placed in the base–value field of the bank descriptor register specified by bits 34 and 33 of X_x. If the x–field of the instruction is 0, Bank Descriptor Register 0 (BDRO) is implicitly specified.

NOTE: Execution of an LB instruction invalidates the bank descriptor specification in GRS location 046 or 047 and the Bank Descriptor Table Pointer (BDTP) in GRS location 040 or 045 for the specified base.

9.15.12. Load Limits – LL 73,15,11

Places operand bits 34–24 23–15 in BDR limits specified by $X_{x\ 34-33}$

Bits 35 through 24 and 23 through 15 of the operand specified by the operand address are placed in the upper and lower limits fields, respectively, of the bank descriptor register specified by bits 34–33 of X_x. If the x–field of the instruction is 0, BDRO is implicitly specified.

NOTE: Execution of an LL instruction invalidates the bank descriptor specification in GRS location 046 and 047 and the Bank Descriptor Table Pointer (BDTP) in GRS location 040 or 045 for the specified base.

9.15.13. Load Addressing Environment - LAE 73,15,12

The double-word operand specified by the operand address contains four bank descriptor specifications in the following format:

E 0	XX	Zeros	BDI0	E 2	XX	Zeros	BDI2
E 1	XX	Zeros	BDI1	E 3	XX	Zeros	BDI3
35	34 33	32 30	29 18	17	16 15	14 12	11 0

This operand is placed in GRS locations 046 and 047, and the limits and base values of the four bank descriptors specified by this operand are placed in the respective bank descriptor registers. The bank descriptor table length check is not performed on the bank descriptor index supplied by the instruction. Bank descriptor flags and use counts are neither interpreted nor altered by LAE.

9.15.14. Store Quantum Timer - SQT 73,15,13

The current value of the quantum timer is stored at the operand address, which may be in GRS or storage. Execution of this instruction has no effect on Quantum Timer Enable designator (D29).

9.15.15. Load Designator Register - LD 73,15,14

The full-word operand specified by the operand address is placed in the designator register. All designator register specifications are in effect at the completion of this instruction.

9.15.16. Store Designator Register - SD 73,15,15

The contents of the designator register is stored at the location by the operand address. Reserved bits of the designator register are zero filled.

9.15.17. User Return - UR 73,15,16

The UR instruction provides an orderly mechanism for returning to a user program. This instruction effectively combines LD and jump, except that the component operations are performed with the correct repertoire, addressing, and register set.

The double-word operand specified by the operand address contains the relative program address and designator register value that establish the user operating state.

The second word of the operand is placed in the designator register, and all specifications are put in effect. The lower 24 bits of the first word of the operand then become the relative program address. If the relative program address is subsequently found out of limits, the interrupt will capture the new P-value.

Bit positions 23 through 18 of the relative address and the A-flag (bit position 35 of the same word) should be 0, unless base register suppression (D35 = 0, D7 = i = 1) is intended or was in effect when the address was stored as the result of an interrupt.

9.15.18. Input/Output Instructions

The I/O instructions are described in detail in Section 4.

For each I/O instruction, the operand address specifies the IOU and subchannel, if applicable. For certain instructions, the index register specified by the a-field of the instruction (X_a) contains a parameter associated with the operation, generally an address. Each I/O instruction skips the next instruction if the I/O instruction is successfully executed. If the I/O instruction is not successfully executed, a 3-bit code is stored in X_a bits 35-33; bits 32-30 of X_a are set to 0, and the next instruction is executed. The 3-bit condition code identifies the fault.

The Input/Output instructions are:

- Start I/O Fast Release (SIOF 75, 01), see 4.4.1.
- Test SubChannel (TSC 75, 03), see 4.4.2.
- Halt DeVice (HDV 75, 04), see 4.4.3.
- Halt CHannel (HCH 75, 05), see 4.4.4.
- Load Channel Register (LCR 75, 10), see 4.4.5.

9.16. Invalid Function Codes

The following operation codes are invalid:

00,00–17	07,16	33,00–17	37,00–07
73,14,00–07	73,14,11–13	73,15,01	73,15,07
73,15,17	73,16	73,17,03–17	74,14,04–17
74,15,04–17	75,(00,02,06,07,11–17)	77,00–17	

Codes 07,00–11; 37,10–17; 72,12; and 72,14 are invalid if the extended instruction set feature is not installed.

Appendix A. Abbreviations, Definitions, and Symbols

A	An arithmetic register. GRS addresses $14–33_8$ and $154–173_8$. Registers at addresses 34, 35, 174, and 175_8 can be used either as general purpose registers or as extensions of the sets of A-registers. In some cases A is used to mean A_a.
A_a	The A-register specified explicitly by the a-field of an instruction word.
A_{a+1}	An A-register having an address one greater than the address of the A-register specified by the a-field of an instruction word.
A_{a+2}	An A-register having an address of two greater than the address of the A-register specified by the a-field of an instruction word.
Absolute Address	A pattern of characters that identifies a specific location in main storage, as opposed to the relative address.
a-field	A-register designator (bits 25–22) of an instruction word. The a-field is interpreted in one of several ways, depending on the instruction word function code. The a-field may specify an A-register, an R-register, or an X-register. For the function code 70_8 (JGD instruction), the j-field and a-field are combined to specify a GRS address. The a-field also is used to specify the I/O channels, a jump key, stop keys, or as an extension of the function code of the instruction.
ANA	Address not available
$\boxed{\text{AND}}$	Logical product AND
ASCII	American Standard Code for Information Interchange (seven bits)
Bank	A set of main storage locations having consecutive addresses. Defined by a bank descriptor word (BDW). Bank addressing is achieved by loading a base value in a designator register to be added to each bank relative address to produce the corresponding absolute address.
BD	Bank descriptor is a two-word set of data defining storage allocation for a program segment.
BDI	Bank descriptor index. An integer value used as an index into a BDT.

BDI Registers	The two locations in the GRS that contain the BDIs (total of 4) for the banks currently addressable by the CPU. GRS locations 046 and 047.
BDR	Bank descriptor register.
BDT	Bank descriptor table.
BDTP	Bank descriptor table pointer
Block Multiplexer	A block multiplexer channel has multiple subchannels and always forces the I/O device to transfer data in multibyte mode.
BT	Block transfer
Byte	A unit of information that consists of eight bits of data.
CAW	The channel address word contains the instruction, IOU and CPU number, channel address, device address, and the address of the first CCW.
CC	Chain command
CCW	Channel command word. A control word located anywhere in storage (location specified by the CAW) used for channel operations. The CCW specifies the device command, data address, CCW flags, format flag, and data count.
CD	Chain data
Channel	An I/O channel provides the hardware control and data paths required to direct the flow of data between a peripheral device and storage.
Channel Base Register	The contents of the channel base register are used to address control words in upper storage.
Characteristic	Biased exponent portion of a floating-point number.
Condition Code	Indicates the channel's response during the execution of an instruction.
Control Word	Refer to CAW and CCW.
Control Module	The control module handles all I/O instructions and resolves storage request and interrupt conflicts for up to five channel modules.
Command Chaining	Allows execution of a new channel command word whenever the present operation is complete at the device level. This will result in the specification of a new operation with the same device without program intervention.
CPU	Central processing unit.
CSW	Channel status word. The channel stores status detected or received during execution of an I/O instruction and ending status associated with noncommunication subchannels and the status table subchannel in the CSW.

DAD	Data address decrement
DAL	Data address lock
Data Chaining	Specifies a new buffer area in storage and permits continuous operation of the device without program intervention.
D-Bank	A bank based on BD.
Designator Bits D-bits	These bits are used to establish and provide control of the CPU operations and to report status. (See 7.2.1)

D35-D30	reserved
D29	Quantum Timer Enable designator
D28	reserved
D27	Fault Interrupt Pointer designator
D26-24	Software Performance Monitor designators
D23	Divide Check designator
D22	Characteristic Overflow designator
D21	Characteristic Underflow designator
D20	Arithmetic Exception Interrupt designator
D19	EXEC BDTP Enable designator
D18	reserved
D17	Floating-Point Residue Store Enable designator Instruction designator
D16	Bank Descriptor Register (BDR3) Write Protection designator
D15	BDR1 Write Protection designator
D14	BDR2 Write Protection designator
D13	BDR0 Write Protection designator
D12	BDR Selector designator
D11	reserved
D10	Quarter-Word Mode designator
D9	reserved
D8	Floating-Point Zero Format Selection designator

	D7	Relocation and Storage Suppression designator
	D6	GRS Selection designator
	D5	Double-Precision Underflow designator
	D4	not used
	D3	Allow Interrupts designator
	D2	Privileged Instruction and GRS Protection designator
	D1	Overflow designator
	D0	Carry designator

Device	A basic peripheral unit from or to which data is transferred in a system.
Device Address	An address generated in the CPU during an I/O instruction and by the control unit to indicate the address of the currently selected device. This is used to associate a particular device with a subchannel operation.
Double Word Boundary	Any even-numbered storage address.
E-bit	Bit 35 of the word in X_a for an LIJ/LDJ instruction and bits 35 and 17 of the BDI registers.
ECC	Error correction code.
ECL	Emitter coupled logic
EF	External function. A control signal sent by an IOU to a peripheral control unit to identify the word on the output data lines as a function word rather than an output data word.
EI	External interrupt. A control signal sent to an IOU by a peripheral control unit that identifies the word on the input word lines as a status word rather than an input data word.
EOT	End of transmission
ESI	Externally specified index.
ESI Interface	Word channel interface capable of addressing up to 64 communications devices on one I/O interface.
f-field	Function code designator (bits 35-30) of an instruction word. The f-field specifies the particular type of operation or function to be performed. The j- and a-fields serve as minor function codes on certain instructions.
GM	Guard mode

Granule	Any group of 64 contiguous words in main storage having addresses in the range $XXXXX000_8$ through $XXXXX777_8$.
GRS	General Register Stack. A group of 128 addressable 36-bit control registers. The CPU uses these high-speed registers for holding intermediate results, indexing, and a variety of special functions such as repeat counting and holding status words.
h-field	Index register incrementation designator (bit 17) of an instruction word. The h-field controls index register modification.
I-bank	A bank based on the I-bank base value of the bank descriptor register.
i-field	Indirect addressing designator (bit 16) of an instruction word. The i-field normally controls indirect addressing. It may be used instead to specify base register suppression/24-bit indexing or use of the utility base for operands, depending on the values of D7.
IGR	Integrated general register
Immediate Command	An operation that results in the subsystem generating an immediate status condition upon receipt of the command code.
Increment	The leftmost 18 bits (12 bits if D9 = D7 = i = 1) of an index register. Symbolized by X_i.
Instruction Word	A statement that specifies an operation and the values or locations of its operands.
I/O	Input/output
IOU	Input/output unit
IPI	Interprocessor interrupt
IPL	Initial program load
ISI	Internally specified index
ISI Interface	A word channel interface that communicates with one peripheral control unit.
j-field	Operand qualifier, partial GRS location, or minor function code designator (bits 29-26) of an instruction word.
K	Uppercase K is used for notational convenience to replace the low order digits of an integral power of 2 or an integral multiple thereof. Thus, 262K is used to represent 262,144 (2^{18}).
k	Represents 1000 in decimal notation.
LJ0 LJ1 LJ2	Indicates the number of bytes in strings SJ0, SJ1, and SJ2, respectively. Stored in $SR3_{35-27}$, $SR3_{26-18}$, and $SR3_{17-8}$, respectively. Maximum value is 511.

LRU	Least recently used
m	Modifier portion of the x-register (See Modifier).
Main Storage Unit (MSU)	Main storage consists of 524K word expandable to 1048K words. The MSU resides in the CPU/IOU cabinet.
Major Function Code	The f-field of an instruction word.
Mantissa	The fractional part of a floating-point number.
MCSW	Machine check status word
Minor Function Code	A portion of an instruction word used with the f-field to specify the operation to be performed. For all instructions that f = 07_8, 33_8, or 37_8, or that f is greater than 70_8, the j-field contains a minor function code. For some instructions, when f is greater than 70_8, the a-field also contains a minor function code.
Modifier	The rightmost 18 bits (24 bits if D7 = i = 1) of an index register. Symbolized by X_m. It is added to the 16-bit address in the u-field of an instruction to produce a relative address (X_m is 18 bits) or absolute address (X_m is 24 bits).
MON	Monitor
MOP	Maintenance operation
MSR	Module select register
MUE	Multiple bit uncorrectable error.
NI	Next instruction
Nonresident Subchannel	A set of control words held in reserve storage.
Nonshared Subchannel	A subchannel intended to operate with communications type peripheral devices. These subchannels allow concurrent access in an interleaving manner by a multiple number of devices through a multiplexing control unit to main storage.
Normalize	To normalize a number in floating point format, the mantissa is shifted left or right until the leftmost bit of the mantissa is not identical to the sign bit.
$\boxed{OR}$	Logical inclusive OR
P	The program address or P-register
PLRU	Paired least recently used
P-Register	The P-register contains the address of the instruction that is currently being executed.

Parity Bit	A binary digit appended to a group of bits to make the number of one bits always odd or always even.
PCI	Program controlled interrupt.
Program Controlled Interrupt	A program controlled bit in a CCW. When set, an interrupt and/or a table entry in the status table is made for that subchannel.
R	A special purpose control register specified explicitly or implicitly by an instruction word. GRS locations 0100 - 0117 and 0120 - 0137.
R_a	The R-register specified by the a-field of an instruction word.
Relative Address	Normally, the address (U) formed by the addition of u, the address field of an instruction, and X_m, the modifier portion of the index register specified by the instruction ($U = u + X_m$).
Relative P+1	An 18-bit relative address captured by certain jump instructions, formed by subtracting the active PSRs BI or BD value that corresponds to the value used to develop the absolute jump to address for the most recent previous jump instruction from the address of the instruction following the current jump instruction.
Resident Subchannel	A set of control words held in the central control module.
Residue	The least significant result word produced by a single-precision Floating Add or Floating Add Negative instruction.
RTC	Real-Time Clock
R0	Real-Time clock register at GRS location 0100, or the control register at GRS location 0120.
R1	Repeat count control registers at GRS location 0101 and 0121. They are used during Block Transfer, search, and masked search instructions.
R2	Masked control registers at GRS location 0102 and 0122. They are used during masked search instructions and the Masked Load Upper instruction.
S	Sign bit or bit position
SBE	Single bit error
SCISR	Storage check interrupt status request
SD	The 24-bit D-bank absolute address developed through addition: $SD = (u + BD) + X_m$ or $SD = (u + BD) + X_m + O_w$.

Shared Subchannel	A subchannel is shared if two or more devices use the same subchannel for I/O operations. On a shared subchannel only one device at a time can transfer data.
SI	The 24-bit I-bank absolute address developed through addition: SI = $(u + BI) + X_m$ or SI = $(u + BI) + X_m + O_w$.
SIL	Select interrupt locations
SIOF Queue	Used for storing the device address for SIOF instructions presented by the CPU but not yet executed by the IOU.
SK	Skip data
SLI	Suppress length indication
STCW	Status table control word
SIU	Storage interface unit, a free-standing cabinet made up of one to four 4K word segments of high-speed buffer storage.
SIU Half	The segments in an SIU that are associated with the lower address range or with the upper address range. There are one or two SIU segments in a half.
SSP	System support processor
Subchannel	A subchannel is an organization of uniquely addressable access paths that are capable of independently sustaining a single I/O operation concurrent with other I/O operations.
Subsystem Clear	An I/O CLEAR signal originating at the IOU goes out on all 24 channels of that IOU.
System Reset	Clears all IOU registers and control designators, resets all peripheral subsystems, and initializes all resident subchannels to idle mode.
TIC	Transfer in channel. A command stored as part of the CCW to perform a branch between noncontiguous CCWs.
TS	Truncated search
TSC	Test subchannel
TSW	Table status word
U	The 18-bit value produced in the index subsection by adding the rightmost 18 bits (X_m-field) of the index register specified by the x-field of the instruction (or by adding 0 if X = 0) to the 16-bit value in the u-field of the instruction (u-field is extended to 18 bits). $U = u + X_m$
U_c	Relative control string address
U_d	Relative destination address

U_s	Relative source address
u-field	The contents of bit positions 15-0 of an instruction word.
W-field	The count field of an access control word. For ISI operations, the W-field is bits 33-18. For half-word ESI operations, the W-field is bits 32-18. For quarter-word ESI operations, the W-field is bits 29-18.
Word Interface	A set of cable drivers and receivers for communicating with one peripheral control unit. A word channel consists of four word interfaces.
X	Index register. GRS location 01 - 017 and 0141 - 0157.
X_{+1}	An index register having an address one greater than the address of the index register specified by the x-field of an instruction word.
X_{+2}	An index register having an address two greater than the address of the index register specified by the x-field of an instruction word.
X_a	The X-register specified by the a-field of an instruction word.
X_c	Control string index register
X_d	Destination index register
X_i	Normally, bits 35-18 of an index register (bits 35-24 when D7 = i = 1). Used to increment or decrement X_m (the modifier) when specified by the instruction word.
x-field	Index register designator (bits 21-18) of an instruction word.
X_m	Normally, bits 17-0 of an index register (bits 23-0 when D9 = D7 = i = 1).
$\boxed{\text{XOR}}$	Logical exclusive OR
X_s	Source index register
X_t	Translation base
X_x	The X-register specified by the x-field of instruction word. In some cases X is used to mean X_x.
+0	Two words, one word, or a field consisting of all 0's.
-0	Two words, one word, or a field consisting of all 1's.
( )	The contents of the register or location identified by the symbol within the parentheses.
( )'	The ones complement of the register or location identified by the symbol within the parentheses.
$( )_n$	The contents of bit position n of the register or location identified by the symbol within the parentheses. For example, $(A)_{35}$ means the contents of bit position 35 of A.

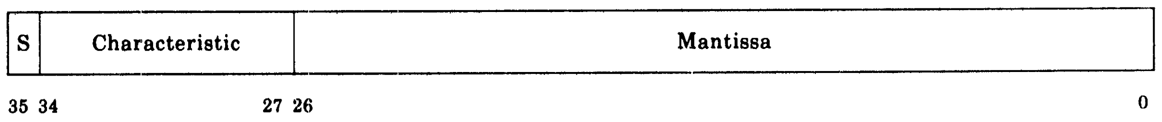
$(\ )_{n-m}$	The contents of bit positions n through m of the register or location identified by the symbol within the parentheses. For example, $(X)_{17-0}$ means the contents of bit positions, 17 through 0 of X.
$\|(\ )\|$	Absolute value or magnitude
$\rightarrow$	Direction of data flow

Appendix B. Summary of Word Formats

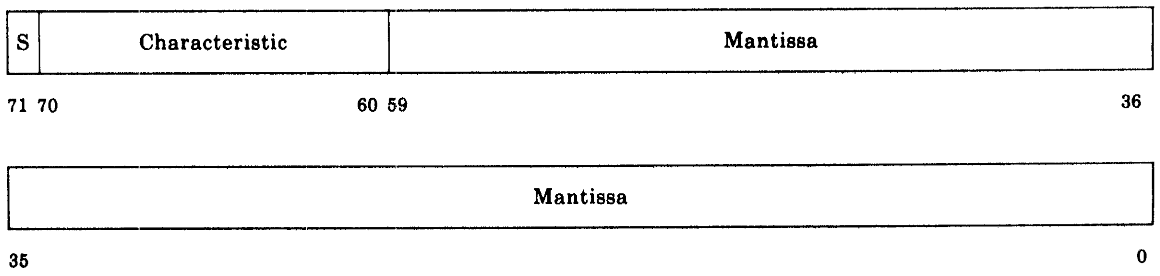
This section list the word formats used on the 1100/70 System in the order as they appeared in the subsections.

See 3.2.8. for the following

Single-Precision Floating-Point Format

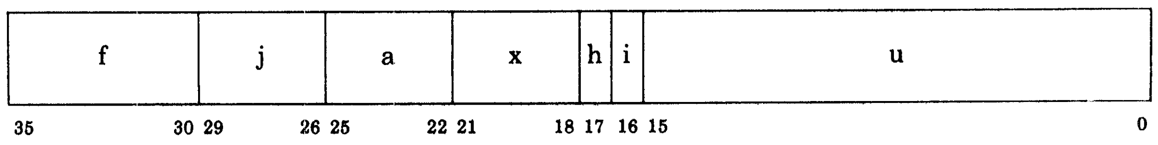


Double-Precision Floating-Point Format



See 3.3.1 for the following:

Instruction Word Format



See 4.3.4.1 for the following:

Channel Address Word 1 Format

Not Used	j	Not Used	IOU No.	Subchannel Address
35–30	29–26	25–12	11 10	9–0

Channel Address Word 2 Format for SIOF Instruction

Not Used	Address of first CCW
35–24	23–0

Channel Address Word 2 Format for TSC Instruction

Not Used	Address for Subchannel Snapshot
35–24	23–0

Channel Address Word 2 Format for LCR Instructions

Not Used	Mask Register Data
35–6	5–0

See 4.3.4.2.1 for the following:

ISI Word Interface CCW Format

Reserved for Software	Command Code	Data Address
35–28	27–24	23–0

Reserved for Software	CCW Flags									Reserved for Software	Word Count
	CD	CC	XXX	SK	XXX	MON	DAD	DAL	XXX		
71–68	67	66	65	64	63	62	61	60	59	58–52	51–36

See 4.3.4.2.2 for the following:

ESI Word Interface CCW Format

Reserved for Software	FCF	Command Code	Data Address
35 30	29 28	27 24	23 0

Reserved for Software	CCW Flags									Reserved for Software	Data Count
	CD	CC	EIC	SK	XXX	MON	DAD	DAL	EOT		
71 68	67	66	65	64	63	62	61	60	59	58 52	51 36

See 4.3.4.2.3 for the following:

Block Multiplexer Channel CCW Format

Reserved for Software	Command Code	Data Address
35 32	31 24	23 0

Reserved for Software	CCW Flags												Reserved for Software	Data Count
	CD	CC	SLI	SK	XXX	TS	DAD	DAL	XXX	FORMAT A	FORMAT B	FORMAT C		
71 68	67	66	65	64	63	62	61	60	59	58	57	56	55 52	51 36

See 4.3.4.2.4 for the following:

Status Table Subchannel CCW Format

Reserved for Software	Command Code	Table Address
35 28	27 24	23 0

Reserved for Software	C D	Reserved for Software	M O N	Reserved for Software	Table Entry Count
71 68	67	66 63	62	61 52	51 36

See 4.9 for the following:

CSW or TSW Format

IOU No.	Subchannel Address	Next CCW Address
35 34	33 24	23 0

Not Used	Device Status	Subchannel Status	Residual Data Count
71 68	67 60	59 52	51 36

External Interrupt Status Word (Word Channel Only)
107 72

See 6.2 for the following:

Storage Address Format

Block Address	Set Address	WS
23 11	10 2	1 0

See 6.2.3.1 for the following:

Internal SIU Storage Check Interrupt Status Format

0	0	0	SSO	BDD	BCD	BBD	BAD	DCU	DCL	TBC	AGE	Age Bits	Requested Address
35	34	33	32	31	30	29	28	27	26	25	24	23 20	19 0

SIU/MSU Interface Storage Check Interrupt Status Format

0	0	1	SSO	SCD	IAC	RDR	MDM	ANA	DPC	ADC	WCC	Requested Address
35	34	33	32	31	30	29	28	27	26	25	24	23 0

See 6.3.2.2 for the following:

Storage Boundary

MS	Not Used	Exp. Adrs	16K Adrs	Column Address	Row Address	FWB
23	22 20	19 18	17 16	15 9	8 2	1 0

See 6.3.2.6 for the following:

MSU Bank Status Word

0	1	0	SSO	MUE	ECC Syndrome	Absolute Address
35	34	33	32	31	30 24	23 0

See 6.3.2.7 for the following:

MSU Maintenance Operation Format

Not Used	MR	Not Used	MR	MRFC	MR
35 32	31 30	29 26	25 14	13 11	10 0

See 7.2.1 for the following:

Designator Register Format

Reserved	D29	R	D27	D26 … D19	R	D17 … D12	R	D10	R	D8 … D0
35 30	29	28	27	26 19	18	17 12	11	10	9	8 0

Designator Register User Mode Format

0 ---- 0	1	0	R	D25	D24	D23	D22	D21	D20	D19	0	D17	D16 … D13	D12	0	D10	0	D8	0	0	D5	0	1	1	D1	D0
35 30	29	28	27 26	25	24	23	22	21	20	19	18	17	16 13	12	11	10	9	8	7	6	5	4	3	2	1	0

Designator Register Interrupt Mode Format

0 ---- 0	0 ---- 0	0 ---- 0	D12	0	0	0	0	1	1	0 ----- 0
35 29	28 24	23 13	12	11	10	9	8	7	6	5 0

Designator Register Executive Mode Format

0 ---- 0	D29	0	R	D25 D20	1	0	D17	D16	D15	D14	D13	D12	0	D10	0	D8	D7	D6	D5	0	D3	0	D1	D0
35 30	29	28	27 26	25 20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

See 7.3.4 for the following:

Bank Descriptor Table Pointer Format

Table Length	Table Address
35 24	23 0

Bank Descriptor Format

Reserved for Software	Base Value
35 18	17 0

Upper Limit	Lower Limit	R	W	P	V	*	C	Use Count
35 24	23 15	14	13	12	11	10	9	8 0

See 8.2.1 for the following:

Program Return Address Word Format

A	Not Used	Program Return Address
35	34 24	23 0

See 8.2.2 for the following:

Address Status Format

E0	0 0	0 - 0	BDI 0	E2	1 0	0 - 0	BDI 2
E1	0 1	0 - 0	BDI 1	E3	1 1	0 - 0	BDI 3
35	34 33	32 30	29 18	17	16 15	14 12	11 0

See 8.3.1 for the following:

Guard Mode Interrupt Status Format

W P	BDR	S L	I F	S O	I T	C R	P I	Zeros	Effective U
35	34 33	32	31	30	29	28	27	26 24	23 0

See 8.3.1 for the following:

Addressing Exception Interrupt Status Format

E N	V	E	R	C O	T	New BDI	E O	BDR	O	C U	C Z	Old BDI
35	34	33	32	31	30	29 18	17	16 15	14	13	12	11 0

See 8.3.3 for the following:

Breakpoint Interrupt Status Format

0 - 0	P	R	W	0 ——— 0	Absolute Breakpoint Address
35 33	32	31	30	29 24	23 0

See 8.3.4 for the following:

Interprocessor Interrupt Status Format

Zeros	CPU
35 2	1 0

See 8.3.6 for the following:

Immediate Check Interrupt Status for Storage Interface Checks Format

0	1	1	MR	OP	RF	MUE	ANA	RDPC	WDPC	CPC	APC	Absolute Address
35	34	33	32	31	30	29	28	27	26	25	24	23 0

Format of Immediate Check Interrupt Status Internal Check 1

1	0	0	MO	RF	Zeros	CS13	CS12	CS11	CS10	CS9	CS8	CS7	CS6	CS5	CS4	CS3	CS2	CS1	CS0	Micro Address
35	34	33	32 30	29	28 25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10 0

Format of Immediate Check Interrupt Status Internal Check 2

1	0	1	MO	RF	MP	MBM	GPC	LM2U	LM2L	LM1U	LM1L	IDS	SM	DM	LFM	SPP	SIS	Zeros	Micro Address
35	34	33	32 30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15 11	10 0

See 8.3.7 for the following:

Delayed Check Interrupt Status for Storage Interface Checks Format

0	1	1	MR	OP	0	MUE	ANA	RDPC	WDPC	CPC	APC	Absolute Address
35	34	33	32	31	30	29	28	27	26	25	24	23 0

Delayed Check Interrupt Status Internal Check 1 Format

1	0	0	MO	Zeros	CS13	CS12	CS11	CS10	CS9	CS8	CS7	CS6	CS5	CS4	CS3	CS2	CS1	CS0	Micro Address
35	34	33	32 30	29 25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10 0

Delayed Check Interrupt Status Internal Check 2 Format

1	0	1	MO	0	MP	MBM	GPC	LM2U	LM2L	LM1U	LM1L	IDS	SM	DM	LFM	SPP	SIS	Zeros	Micro Address
35	34	33	32 30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15 11	10 0

See 8.4.1 for the following:

IOU Machine Check Status Word Format

Zeros	SSPE	CDPE	0	CSPE	CFPE	CPC	DAPE	CAPE	ARC	MUE	MSNA	MSRD	ANA	MSWD	MSWC	MSAC	IOU No.	Subchannel Address
35 28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11 10	9 0

See 8.4.2.1 for the following:

ISI Word Interface CSW Format

IOU	Subchannel Address	Next CCW Address
35 34	33 24	23 0

Not Used	ATN	Zeros	Subchannel Status	Residual Word Count
71 68	67	66 60	59 52	51 36

External Interrupt Status Word
107 72

See 8.4.2.2 for the following:

Block Multiplexer CSW Format

IOU	Subchannel Address	Next CCW Address
35 34	33 24	23 0

Residual Byte Count	Device Status	Subchannel Status	Residual Word Count
71 68	67 60	59 52	51 36

See 8.4.2.3 for the following:

Status Table Subchannel CSW Format

IOU	Subchannel Address	Next CCW Address
35 34	33 24	23 0

Zeros	Subchannel Status	Residual Entry Count
71 60	59 52	51 36

See 8.4.2.4 for the following:

Test Subchannel CSW Format

IOU No.	Subchannel Address	Integrated General Register
35 34	33 24	23 0

Residual Word Count	P	Data Address
71 61	60	59 36

Not Used	Sub. Mode	Subchannel Status	P	P	CCW Flag	P	FCF	Residual Word Count
107 106	105 103	102 91	90	89	88 80	79	78 77	76 72

See 8.4.3 for the following:

Channel Status Word For Table Interrupt

IOU	Status Table Subchannel Address	Next CCW Address
35 34	33 24	23 0

Zeros	Residual Entry Count
71 52	51 36

ESI Word Interface Table Status Word (TSW)

IOU	Subchannel Address	Next CCW Address
35 34	33 24	23 0

Not Used	A T N	Zeros	Subchannel Status	Residual Character Count
71 68	67	66 60	59 52	51 36

External Interrupt Status Word
107 72

See 9.10.1 for the following:

X_a Before Execution of LDJ, LIJ, or LBJ Instruction

E	BDR	0 — 0	New BDI	Not Used
35	34 33	32 30	29 18	17 0

X_a After Execution of LDJ, LIJ, or LBJ Instruction

E	BDR	0 — 0	Old BDI	Relative Program Address
35	34 33	32 30	29 18	17 0

See 9.13.9 for the following:

A_a Format for the SRS Instruction

0 0	Area 2 Count	0 0	Area 2 Address	0 0	Area 1 Count	0 0	Area 1 Address
35 34	33 27	26 25	24 18	17 16	15 9	8 7	6 0

See 9.13.11 for the following:

Test Relative Address Format

Word 0	Designator Register					
Word 1	Bank Descriptor Table Pointer					
Word 2	E 0	ignored	BDI 0	E 2	ignored	BDI 2
Word 3	E 1	ignored	BDI 1	E 3	ignored	BDI 3
	35	34 30	29 18	17	16 12	11 0

Appendix C. Instruction Repertoire

This section contains three tables of the instruction repertoire. Table C-1 lists the instruction repertoire by mnemonic code in alphabetical order. Table C-2 list the instruction repertoire in order of function code. Table C-3 provides cross-reference to octal and mnemonic.

Table C-1. Mnemonic/Function Code Cross-Reference

Mnemonic	Function Code (Octal)		
	f	j	Paragraph Reference
A,AA	14		9.4.1
A,AX	24		9.4.7
AAIJ	74	07	9.9.3
ADE	07	00	9.14.7.1
AH	72	04	9.4.17
AM,AMA	16		9.4.3
AN,ANA	15		9.4.2
AN,ANX	25		9.4.8
AND	42		9.12.3
ANH	72	05	9.4.18
ANM,ANMA	17		9.4.4
ANT	72	07	9.4.20
ANU	21		9.4.6
AT	72	06	9.4.19
AU	20		9.4.5
BBN	72	14	9.14.8.2.
BDE	37	15	9.14.4.
BIC	37	11	9.14.2.
BICL	37	13	9.14.3.2
BIM	37	10	9.14.1
BIML	37	14	9.14.3.3
BMTC	37	12	9.14.3.1
BN	72	12	9.14.8.1
BT	22		9.3.8
CDU	76	07	9.5.16
Clear SC	73	15	9.15.10
	a = 06		
DA	71	10	9.4.15
DADE	07	01	9.14.7.2
DAN	71	11	9.4.16
DDEI	07	07	9.14.7.6
DEB	37	16	9.14.5
DEC	05	00-17	9.13.12
	a = 11		
DEC2	05	00-17	9.13.12
	a = 13		
DEI	07	06	9.14.7.5
DF	36		9.4.14
DFA	76	10	9.5.3
DFAN	76	11	9.5.4
DFD	76	13	9.5.8
DFM	76	12	9.5.6
DFP,DLCF	76	15	9.5.12
DFU	76	14	9.5.10
DI	34		9.4.12
DIDE	07	11	9.14.7.8
DJZ	71	16	9.11.2
DL	71	13	9.2.9
DLM	71	15	9.2.11
DLN	71	14	9.2.10
DLSC	73	07	9.8.8
DS	71	12	9.3.7

Mnemonic	Function Code (Octal)		Paragraph Reference
	f	j	
DSA	73	05	9.8.6
DSC	73	01	9.8.2
DSDE	07	03	9.14.7.4
DSF	35		9.4.13
DSL	73	03	9.8.4
DTE	71	17	9.7.14
EDDE	37	17	9.14.6
ENZ	05	00–17	9.13.12
	a = 14–17		
ER	72	11	9.13.4
EX	72	10	9.13.3
FA	76	00	9.5.1
FAN	76	01	9.5.2
FCL	76	17	9.5.14
FD	76	03	9.5.7
FEL	76	16	9.5.13
FM	76	02	9.5.5
HCH	75	05	4.4.4
HDV	75	04	4.4.3
HJ,HKJ	74	05	9.11.10
IDE	07	10	9.14.7.7
IIIX	73	15	9.15.8
	a = 04		
IMI	72	00	9.15.1
INC	05	00–17	9.13.12
	a = 10		
INC2	05	00–17	9.13.12
	a = 12		
J,JK	74	04	9.11.9
JB	74	11	9.11.12
JC	74	16	9.11.22
JDF	74	14	9.11.17
	a = 03		
JFO	74	14	9.11.16
	a = 02		
JFU	74	14	9.11.15
	a = 01		
JGD	70		9.11.1
JMGI	74	12	9.11.13
JN	74	03	9.11.8
JNB	74	10	9.11.11
JNC	74	17	9.11.23
JNDF	74	15	9.11.21
	a = 03		
JNFO	74	15	9.11.20
	a = 02		
JNFU	74	15	9.11.19
	a = 01		
JNO	74	15	9.11.18
	a = 00		
JNS	72	03	9.11.4

Mnemonic	Function Code (Octal)		Paragraph Reference
	f	j	
JNZ	74	01	9.11.6
JO	74	14	9.11.14
	a = 00		
JP	74	02	9.11.7
JPS	72	02	9.11.3
JZ	74	00	9.11.5
L,LA	10		9.2.1
L,LR	23		9.2.5
L,LX	27		9.2.7
LAE	73	15	9.15.13
	a = 12		
LAQW	07	04	9.14.9.1
LB	73	15	9.15.11
	a = 10		
LBJ	07	17	9.10.1
LBRX	73	15	9.15.6
	a = 02		
LCF	76	05	9.5.11
LCR	75	10	4.4.5
LD	73	15	9.15.15
	a = 14		
LDC	73	14	9.15.3
	a = 10		
LDJ	07	12	9.10.3
LDSC	73	11	9.8.10
LDSL	73	13	9.8.12
LIJ	07	13	9.10.2
LL	73	15	9.15.12
	a = 11		
LM,LMA	12		9.2.3
LMJ	74	13	9.9.2
LN,LNA	11		9.2.2
LNMA	13		9.2.4
LPD	07	14	9.13.1
LQT	73	15	9.15.7
	a = 03		
LRS	72	17	9.13.10
LSC	73	06	9.8.7
LSSC	73	10	9.8.9
LSSL	73	12	9.8.11
LUF	76	04	9.5.9
LXI	46		9.2.8
LXM	26		9.2.6
MASG	71	07	9.6.14
MASL	71	06	9.6.13
MCDU	76	06	9.5.15
MF	32		9.4.11
MI	30		9.4.9
MLU	43		9.12.4
MSE	71	00	9.6.7
MSG	71	03	9.6.10

Mnemonic	Function Code (Octal) f	j	Paragraph Reference
MSI	31		9.4.10
MSLE,MSNG	71	02	9.6.9
MSNE	71	01	9.6.8
MSNW	71	05	9.6.12
MSW	71	04	9.6.11
NOP	74	06	9.13.8
0R	40		9.12.1
PAIJ	72	13	9.15.2
S,SA	01		9.3.1
S,SR	04		9.3.4
S,SX	06		9.3.6
SAQW	07	05	9.14.9.2
SAS	05	00–17	9.3.5
	a = 06		
SAZ	05	00–17	9.3.5
	a = 07		
SD	73	15	9.15.16
	a = 15		
SDE	07	02	9.14.7.3
SE	62		9.6.1
SFS	05	00–17	9.3.5
	a = 04		
SFZ	05	00–17	9.3.5
	a = 05		
SG	65		9.6.4
SIL	73	15	9.15.5
	a = 00		
SIOF	75	01	4.4.1
SLE,SNG	64		9.6.3
SLJ	72	01	9.9.1
SM,SMA	03		9.3.3
SN,SNA	02		9.3.2
SNE	63		9.6.2
SNW	67		9.6.6
SNZ	05	00–17	9.3.5
	a = 01		
SN1	05	00–17	9.3.5
	a = 03		

Mnemonic	Function Code (Octal) f	j	Paragraph Reference
SPD	07	15	9.13.2
SPID	73	15	9.15.9
	a = 05		
SP1	05	00–17	9.3.5
	a = 02		
SQT	73	15	9.15.14
	a = 13		
SRS	72	16	9.13.9
SSA	73	04	9.8.5
SSC	73	00	9.8.1
SSL	73	02	9.8.3
SW	66		9.6.5
SZ	05	00–17	9.3.5
	a = 00		
TCS	73	17	9.13.7
	a = 02		
TE	52		9.7.6
TEP	44		9.7.1
TG	55		9.7.9
TLE,TNG	54		9.7.8
TLEM,TNGM	47		9.7.3
TN	61		9.7.13
TNE	53		9.7.7
TNW	57		9.7.11
TNZ	51		9.7.5
TOP	45		9.7.2
TP	60		9.7.12
TRA	72	15	9.13.11
TS	73	17	9.13.5
	a = 00		
TSC	75	03	4.4.2
TSS	73	17	9.13.6
	a = 01		
TW	56		9.7.10
TZ	50		9.7.4
UR	73	15	9.15.17
	a = 16		
XOR	41		9.12.2

Table C-2. Instruction Repertoire

f	j	a	Mnemonic	Instruction	Description
00	0–17		–	Invalid Code	Causes Invalid Instruction interrupt to MSR + 221_8
01	0–15		S,SA	Store A	$(A_a) \rightarrow U$
02	0–15		SN,SNA	Store Negative A	$-(A_a) \rightarrow U$
03	0–15		SM,SMA	Store Magnitude A	$\lvert (A_a) \rvert \rightarrow U$
04	0–15		S,SR	Store R	$(R_a) \rightarrow U$
05	0–17	00	SZ	Store Zero	Stores constant 000000 000000, zeros, in location specified by operand address.
05	0–17	01	SNZ	Store Negative Zero	Stores constant 777777 777777, negative zero, in location specified by operand address.
05	0–17	02	SP1	Store Positive One	Stores constant 000000 000001, positive one, in location specified by operand address.
05	0–17	03	SN1	Store Negative One	Stores constant 777777 777776, negative one, in location specified by operand address.
05	0–17	04	SFS	Store Fieldata Spaces	Stores constant 050505 050505, Fieldata spaces, in location specified by operand address.
05	0–17	05	SFZ	Store Fieldata Zeros	Stores constant 606060 606060, Fieldata zeros, in location specified by operand address.
05	0–17	06	SAS	Store ASCII Spaces	Stores constant 040040 040040, ASCII spaces, in location specified by operand address.
05	0–17	07	SAZ	Store ASCII Zeros	Stores constant 060060 060060, ASCII zeros, in location specified by operand address.
05	0–17	10	INC	Increase Operand by One	Increases operand by one. If initial operand or result is zero, execute NI; if not zero, skip NI.
05	0–17	11	DEC	Decrease Operand by One	Decreases operand by one. If initial operand or result is zero, execute NI; if not zero, skip NI.
05	0–17	12	INC2	Increase Operand by Two	Increases operand by two. If initial operand or result is zero, execute NI; if not zero, skip

Table C-2. Instruction Repertoire (continued)

f	j	a	Mnemonic	Instruction	Description
					NI.
05	0–17	13	DEC2	Decrease Operand by Two	Decreases operand by two. If initial operand or result is zero, execute NI; if not zero, skip NI.
05	0–17	14–17	ENZ	Eliminate Negative Zero	Increases operand by zero. If initial operand or result is zero execute NI; if not zero, skip NI.
06	0–15		S,SX	Store X	$(X_a) \rightarrow U$
07	00		ADE	Add Decimal	Adds U to an A–register and stores result in A–register.
07	01		DADE	Double Add Decimal	Adds two–word operand(U, U+1) to two A–registers (A_a, A_{a+1}) and stores results in A_a and A_{a+1}.
07	02		SDE	Subtract Decimal	Subtracts U from an A–register and stores result in A–register.
07	03		DSDE	Double Subtract Decimal	Subtracts two–word operand from A–register (A_a, A_{a+1}) and stores result in A_a and A_{a+1}.
07	04		LAQW	Load a Quarter Word	(Selected quarter word) $\rightarrow (A_a)_{8-0}$
07	05		SAQW	Store a Quarter Word	$(A_a)_{8-0} \rightarrow$ Selected quarter word
07	06		DEI	Decimal to Integer	Converts (U) from signed magnitude format to a ones complement binary format and stores in A–register.
07	07		DDEI	Double Decimal to Integer	U, U+1 converts from decimal to ones complement binary format and stores in A_a and A_{a+1}.
07	10		IDE	Integer to Decimal	Converts U in ones complement to decimal format, stores in a pair of A–registers.
07	11		DIDE	Double Integer to Decimal	Converts U, U+1 from ones complements to decimal, stores in A_a, A_{a+1}, A_{a+2}.
07	12		LDJ	Load D–Bank Base and Jump	Ignores X_a bit positions 34–33; if D12 = 0, load BDR2; if D12 = 1, select BDR3; jump to U.
07	13		LIJ	Load I–Bank Base and Jump	Ignores X_a bit positions 34–33; if D12 = 0, load BDR0; if D12 =1, select BDR1; jump to

Table C-2. Instruction Repertoire (continued)

f	j	a	Mnemonic	Instruction	Description
					U.
07	14		LPD	Load DR Designators	$U_{6,5,3-1}$ → Designator Register Bit 6 → D20 Bit 2 → D8 Bit 5 → D17 Bit 1 → D5 Bit 3 → D10
07	15		SPD	Store DR Designators	Designator Register bits (D-bits) → U_{6-1} D20 → Bit 6 D10 → Bit 3 D17 → Bit 5 D8 → Bit 2 D12 → Bit 4 D5 → Bit 1
07	16		–	Invalid Code	Causes Invalid Instruction interrupt to MSR + 221_8.
07	17		LBJ	Load Bank and Jump	Loads BDR selected by bit positions 34–33; jump to U.
10	0–17		L,LA	Load A	$(U) \rightarrow A_a$
11	0–17		LN,LNA	Load Negative A	$-(U) \rightarrow A_a$
12	0–17		LM,LMA	Load Magnitude A	$\lvert(U)\rvert \rightarrow A_a$
13	0–17		LNMA	Load Negative Magnitude A	$-\lvert(U)\rvert \rightarrow A_a$
14	0–17		A,AA	Add to A	$(A_a) + (U) \rightarrow A_a$
15	0–17		AN,ANA	Add Negative to A	$(A_a) - (U) \rightarrow A_a$
16	0–17		AM, AMA	Add Magnitude to A	$(A_a) + \lvert(U)\rvert \rightarrow A_a$
17	0–17		ANM, ANMA	Add Negative Magnitude to A	$(A_a) - \lvert(U)\rvert \rightarrow A_a$
20	0–17		AU	Add Upper	$(A_a) + (U) \rightarrow A_{a+1}$
21	0–17		ANU	Add Negative Upper	$A_a - (U) \rightarrow A_{a+1}$
22	0–15		BT	Block Transfer	$(X_x + u) \rightarrow X_a + u$; repeat k times
23	0–17		L,LR	Load R	$(U) \rightarrow R_a$
24	0–17		A,AX	Add to X	$(X_a) + (U) \rightarrow X_a$

Table C-2. Instruction Repertoire (continued)

f	j	a	Mnemonic	Instruction	Description
25	0–17		AN,ANX	Add Negative to X	$(X_a) - (U) \rightarrow X_a$
26	0–17		LXM	Load X Modifier	$(U) \rightarrow X_{a\ 17-0}$; $X_{a\ 35-18}$ unchanged
27	0–17		L,LX	Load X	$(U) \rightarrow X_a$
30	0–17		MI	Multiply Integer	$(A_a) * (U) \rightarrow A_a, A_{a+1}$
31	0–17		MSI	Multiply Single Integer	$(A_a) * (U) \rightarrow A_a$
32	0–17		MF	Multiply Fractional	$(A_a) * (U) \rightarrow A_a, A_{a+1}$, left circular one bit
33	0–17		–	Invalid Code	Causes Invalid Instruction interrupt to MSR $+ 221_8$
34	0–17		DI	Divide Integer	$(A_a, A_{a+1}) \div (U) \rightarrow A_a$; REMAINDER $\rightarrow A_{a+1}$
35	0–17		DSF	Divide Single Fractional	$[(A_a$, 36 sign bits) right algebraic shift 1 place$] \div (U) \rightarrow A_{a+1}$
36	0–17		DF	Divide Fractional	$[(A_a, A_{a+1}$ right algebraic shift 1 place$] \div (U) \rightarrow A_a$; REMAINDER $\rightarrow A_{a+1}$
37	0–7		–	Invalid Code	Causes Invalid Instruction interrupt to MSR $+ 221_8$.
37	10		BIM	Bit Move	Moves a source string of bits that starts on any bit boundary to a destination string that also starts on any bit boundary.
37	11		BIC	Bit Compare	Compares a source string of bits to a destination string of bits.
37	12		BMTC	Bit Move with Translation and Control	Moves a source string of characters to a destination string of characters.
37	13		BICL	Bit Compare Long	Compares a source string of characters to a destination string.
37	14		BIML	Bit Move Long	Moves a source string of bits that starts on any bit boundary to a destination that also starts on any bit boundary.
37	15		BDE	Byte to Decimal	Converts a string of either ASCII or External Comp 3 characters to a signed magnitude format, results stored in A_a,

Table C-2. Instruction Repertoire (continued)

f	j	a	Mnemonic	Instruction	Description
					A_{a+1}, A_{a+2}.
37	16		DEB	Decimal to Byte	Converts a string of decimal characters in BCD signed magnitude format to either ASCII or External Comp 3 format.
37	17		EDDE	Edit Decimal	Moves a source string to a destination string.
40	0–17		OR	Logical OR	(A_a) $\boxed{\text{OR}}$ $(U) \rightarrow A_{a+1}$
41	0–17		XOR	Logical Exclusive OR	(A_a) $\boxed{\text{XOR}}$ $(U) \rightarrow A_{a+1}$
42	0–17		AND	Logical AND	(A_a) $\boxed{\text{AND}}$ $(U) \rightarrow A_{a+1}$
43	0–17		MLU	Masked Load Upper	$[(U)$ $\boxed{\text{AND}}$ $(R2)]$ $\boxed{\text{OR}}$ $[(A_a)$ $\boxed{\text{AND}}$ $\text{NOT } (R2)] \rightarrow A_{a+1}$
44	0–17		TEP	Test Even Parity	Skips NI if (U) $\boxed{\text{AND}}$ (A_a) has even parity
45	0–17		TOP	Test Odd Parity	Skips NI if (U) $\boxed{\text{AND}}$ (A_a) has odd parity
46	0–17		LXI	Load X Increment	$(U) \rightarrow (X_a)_{35-18}$; $(X_a)_{17-0}$ unchanged
47	0–17		TLEM TNGM	Test Less Than or Equal to Modifier Test Not Greater Than Modifier	Skips NI if $(U)_{17-0} \leq (X_a)_{17-0}$; always $(X_a)_{17-0} + (X_a)_{35-18} \rightarrow X_{a\ 17-0}$
50	0–17		TZ	Test Zero	Skips NI if $(U) = \pm 0$
51	0–17		TNZ	Test Nonzero	Skips NI if $(U) \neq \pm 0$
52	0–17		TE	Test Equal	Skips NI if $(U) = (A_a)$
53	0–17		TNE	Test Not Equal	Skips NI if $(U) \neq (A_a)$
54	0–17		TLE TNG	Test Less Than or Equal Test Not Greater	Skips NI if $(U) \leq (A_a)$
55	0–17		TG	Test Greater	Skips NI if $(U) > (A_a)$
56	0–17		TW	Test Within Range	Skips NI if $(A_a) < (U) \leq (A_{a+1})$

Table C-2. Instruction Repertoire (continued)

f	j	a	Mnemonic	Instruction	Description
57	0–17		TNW	Test Not Within Range	Skips NI if $(U) \leq (A_a)$ or $(U) > (A_{a+1})$
60	0–17		TP	Test Positive	Skips NI if $(U)_{35} = 0$
61	0–17		TN	Test Negative	Skips NI if $(U)_{35} = 1$
62	0–17		SE	Search Equal	Skips NI if $(U) = (A_a)$, else repeat
63	0–17		SNE	Search Not Equal	Skips NI if $(U) \neq (A_a)$, else repeat
64	0–17		SLE	Search Less Than or Equal	Skips NI if $(U) \leq (A_a)$, else repeat
			SNG	Search Not Greater	
65	0–17		SG	Search Greater	Skips NI if $(U) > (A_a)$, else repeat
66	0–17		SW	Search Within Range	Skips NI if $(A_a) < (U) \leq (A_{a+1})$, else repeat
67	0–17		SNW	Search Not Within Range	Skips NI if $(U) \leq (A_a)$ or $(U) > (A_{a+1})$, else repeat
70	0–17		JGD	Jump Greater and Decrement	Jumps to U if $(\text{Control Register})_{ja} > 0$; goes to NI if $(\text{Control Register})_{ja} \leq 0$; always $(\text{Control Register})_{ja} - 1 \rightarrow \text{Control Register}_{ja}$
71	00		MSE	Masked Search Equal	Skips NI if $(U)\ \boxed{\text{AND}}\ (R2) = (A_a)\ \boxed{\text{AND}}\ (R2)$, else repeat
71	01		MSNE	Masked Search Not Equal	Skips NI if $(U)\ \boxed{\text{AND}}\ (R2) \neq (A_a)\ \boxed{\text{AND}}\ (R2)$, else repeat
71	02		MSLE	Masked Search Less Than or Equal	Skips NI if $(U)\ \boxed{\text{AND}}\ (R2) \leq (A_a)\ \boxed{\text{AND}}\ (R2)$, else repeat
			MSNG	Masked Search Not Greater	
71	03		MSG	Masked Search Greater	Skips NI if $(U)\ \boxed{\text{AND}}\ (R2) > (A_a)\ \boxed{\text{AND}}\ (R2)$, else repeat
71	04		MSW	Masked Search Within Range	Skips NI if $(A_a)\ \boxed{\text{AND}}\ (R2) < (U)\ \boxed{\text{AND}}\ (R2) \leq (A_{a+1})\ \boxed{\text{AND}}\ (R2)$, else repeat
71	05		MSNW	Masked Search Not Within Range	Skips NI if $(U)\ \boxed{\text{AND}}\ (R2) \leq (A_a)\ \boxed{\text{AND}}\ (R2)$ or $(U)\ \boxed{\text{AND}}\ (R2) > (A_{a+1})\ \boxed{\text{AND}}\ (R2)$, else repeat

Table C-2. Instruction Repertoire (continued)

f	j	a	Mnemonic	Instruction	Description
71	06		MASL	Masked Alphanumeric Search Less Than or Equal	Skips NI if (U) $\boxed{\text{AND}}$ (R2) $\leq$ (A_a) $\boxed{\text{AND}}$ (R2), else repeat
71	07		MASG	Masked Alphanumeric Search Greater	Skips NI if (U) $\boxed{\text{AND}}$ (R2) $>$ (A_a) $\boxed{\text{AND}}$ (R2), else repeat
71	10		DA	Double-Precision Fixed-Point Add	$(A_a, A_{a+1}) + (U, U+1) \rightarrow A_a, A_{a+1}$
71	11		DAN	Double-Precision Fixed-Point Add Negative	$(A_a, A_{a+1}) - (U, U+1) \rightarrow A_a, A_{a+1}$
71	12		DS	Double Store A	$(A_a, A_{a+1}) \rightarrow U, U+1$
71	13		DL	Double Load A	$(U, U+1) \rightarrow A_a, A_{a+1}$
71	14		DLN	Double Load Negative A	$-(U, U+1) \rightarrow A_a, A_{a+1}$
71	15		DLM	Double Load Magnitude A	$\lvert (U, U+1) \rvert \rightarrow A_a, A_{a+1}$
71	16		DJZ	Double-Precision Jump Zero	Jumps to U if $(A_a, A_{a+1}) = \pm 0$; goes to NI if $(A_a, A_{a+1}) \neq \pm 0$
71	17		DTE	Double-Precision Test Equal	Skips NI if $(U, U+1) = (A_a, A_{a+1})$
72	00		IMI	Initiate Maintenance Interrupt	Sends Attention interrupt to the SSP specified by A_0 and skips NI if there are no outstanding Attention interrupts to that SSP; otherwise, NOP.
72	01		SLJ	Store Location and Jump	Relative P+1 $\rightarrow U_{17-0}$; jump to U+1
72	02		JPS	Jump Positive and Shift	Jumps to U if $(A_a)_{35} = 0$; goes to NI if $(A_a)_{35} = 1$; always shift (A_a) left circularly one bit position
72	03		JNS	Jump Negative and Shift	Jumps to U if $(A_a)_{35} = 1$; goes to NI if $(A_a)_{35} = 0$; always shift (A_a) left circularly one bit position
72	04		AH	Add Halves	$(A_a)_{35-18} + (U)_{35-18} \rightarrow A_{a\ 35-18}$; $(A_a)_{17-0} + (U)_{17-0} \rightarrow A_{a\ 17-0}$

Table C-2. Instruction Repertoire (continued)

f	j	a	Mnemonic	Instruction	Description
72	05		ANH	Add Negative Halves	$(A_a)_{35-18}$ - $(U)_{35-18}$ → $A_{a\ 35-18}$; $(A_a)_{17-0}$ - $(U)_{17-0}$ → $A_{a\ 17-0}$
72	06		AT	Add Thirds	$(A_a)_{35-24}$ + $(U)_{35-24}$ → $A_{a\ 35-24}$; $(A_a)_{23-12}$ + $(U)_{23-12}$ → $A_{a\ 23-12}$; $(A_a)_{11-0}$ + $(U)_{11-0}$ → A_{a} 11-0
72	07		ANT	Add Negative Thirds	$(A_a)_{35-24}$ - $(U)_{35-24}$ → $A_{a\ 35-24}$; $(A_a)_{23-12}$ - $(U)_{23-12}$ → $A_{a\ 23-12}$; $(A_a)_{11-0}$ - $(U)_{11-0}$ → A_{a} 11-0
72	10		EX	Execute	Executes the instruction at U
72	11		ER	Executive Request	Generates Executive Request interrupt to MSR + 222_8.
72	12		BN	Bit Normalize	$(36_{10}$ * $X_{a\ 17-0}$ + $X_{a\ 35-30}$ + signed $U_{35-0})/36_{10}$ → result and remainder
72	13		PAIJ	Prevent All Interrupts and Jump	Prevents all interrupts and jumps to U
72	14		BBN	Byte to Bit Normalize	$(36_{10}$ * $X_{a\ 17-0}$ + 9 * $(X_{a\ 31-30}$ + signed $U_{35-0}))/36_{10}$ → result and remainder
72	15		TRA	Test Relative Address	Determines if a relative address specified in X_a is within a given relative addressing range
72	16		SRS	Store Register Set	Transfers GRS areas defined in A_a to consecutive storage starting at address U
72	17		LRS	Load Register Set	Transfers from consecutive storage, starting at location U, to GRS areas defined in A_a
73	00		SSC	Single Shift Circular	Shifts (A_a) right circularly U places
73	01		DSC	Double Shift Circular	Shifts (A_a, A_{a+1}) right circularly U places
73	02		SSL	Single Shift Logical	Shifts (A_a) right U places, zero fill
73	03		DSL	Double Shift Logical	Shifts (A_a, A_{a+1}) right U places, zero fill
73	04		SSA	Single Shift Algebraic	Shifts (A_a) right U places, sign fill

Table C-2. Instruction Repertoire (continued)

f	j	a	Mnemonic	Instruction	Description
73	05		DSA	Double Shift Algebraic	Shifts (A_a, A_{a+1}) right U places, sign fill
73	06		LSC	Load Shift and Count	(U) $\rightarrow$ A_a; shift (A_a) left circularly until $(A_a)_{35} \neq (A_a)_{34}$; number of shifts $\rightarrow$ A_{a+1}
73	07		DLSC	Double Load Shift and Count	(U, U+1) $\rightarrow$ A_a, A_{a+1}; shift (A_a, A_{a+1}) left circularly until $(A_a, A_{a+1})_{71} \neq (A_a, A_{a+1})_{70}$; number of shifts $\rightarrow$ A_{a+2}
73	10		LSSC	Left Single Shift Circular	Shifts (A_a) left circularly U places
73	11		LDSC	Left Double Shift Circular	Shifts (A_a, A_{a+1}) left circularly U places
73	12		LSSL	Left Single Shift Logical	Shifts (A_a) left U places, zero fill
73	13		LDSL	Left Double Shift Logical	Shifts (A_a, A_{a+1}) left U places, zero fill
73	14	00–07	–	Invalid Code	Causes Invalid Instruction interrupt to MSR $+221_8$
73	14	10	LDC	Load Dayclock	Replaces dayclock register value with fixed storage value at start of next update cycle.
73	14	11–13	–	Invalid Code	Causes Invalid Instruction interrupt to MSR $+ 221_8$
73	14	16	MDC	MicroDiagnostic C	Used for CPU Fault Injection
73	14	14, 15, 17		Operation codes	Reserved for future use. Executed as NOP instruction.
73	15	00	SIL	Select Interrupt Locations	(U) specifies control parameter and modes of operation to SIU or MSU.
73	15	01		Invalid Code	Causes Invalid Instruction interrupt to MSR $+ 221_8$
73	15	02	LBRX	Load Breakpoint Register	Transfers operand to Breakpoint Register
73	15	03	LQT	Load Quantum Timer	Places full-word operand in Quantum Timer

Table C-2. Instruction Repertoire (continued)

f	j	a	Mnemonic	Instruction	Description
73	15	04	IIIX	Initiate Interprocessor Interrupt	Interrupt processor specified by operand address value; if interprocessor interrupt mechanism of that processor is available, skips NI; otherwise goes to NI.
73	15	05	SPID	Store Processor ID	Stores binary serial number in first third; 2-character Fieldata revision level in second third; processor features in the fifth sixth; processor number in last sixth of operand
73	15	06	Clear SC	Clear Support Controller	If $U_0 = 0$, clear SC except for SSP select field If $U_0 = 1$, clear any outstanding IMI requests to the SC
73	15	07	–	Invalid Code	Causes Invalid Instruction interrupt to MSR + 221_8
73	15	10	LB	Load Base	Places operand bits 17 through 0 in base value field of BDR specified by bits 34 and 33 of X_x
73	15	11	LL	Load Limits	Places operand bits 35 through 24 and 23 through 15 in BDR limits fields specified by X_x bits 34 and 33
73	15	12	LAE	Load Addressing Environment	Places the double-word operand in GRS location 046 and 047, and places the limits and base values of the four Bank Descriptors specified by this operand in the respective Bank Descriptor Registers
73	15	13	SQT	Store Quantum Timer	Stores Quantum Timer value at the operand address location. Executing this instruction has no effect on D29.
73	15	14	LD	Load Designator Register	Places full-word operand in Designator register
73	15	15	SD	Store Designator Register	Stores Designator register contents at location specified by operand address
73	15	16	UR	User Return	$(U + 1) \rightarrow$ Designator register; jumps to address specified by $(U)_{23-0}$ using new register set
73	15	17	–	Invalid Code	Causes Invalid Instruction interrupt to MSR + 221_8

Table C-2. Instruction Repertoire (continued)

f	j	a	Mnemonic	Instruction	Description
73	16		–	Invalid Code	Causes Invalid Instruction interrupt to MSR $+221_8$
73	17	00	TS	Test and Set	If $(U)_{30} = 1$, generates Test and Set interrupt; if $(U)_{30} = 0$, goes to NI; if $U \geq 200$, then $01_8 \rightarrow U_{35-30}$; $(U)_{29-0}$ unchanged
73	17	01	TSS	Test and Set and Skip	If $(U)_{30} = 1$, goes to NI; if $(U)_{30} = 0$, skips NI; and $01_8 \rightarrow U_{35-30}$; $(U)_{29-0}$ unchanged
73	17	02	TCS	Test and Clear and Skip	If $(U)_{30} = 0$, performs NI; if $(U)_{30} = 1$, skips NI; clears $(U)_{35-30}$; $(U)_{29-0}$ unchanged
73	17	3–17	–	Invalid Code	Causes Invalid Instruction interrupt to MSR $+ 221_8$
74	00		JZ	Jump Zero	Jumps to U if $(A_a) = \pm 0$; goes to NI if $(A_a) \neq \pm 0$
74	01		JNZ	Jump Nonzero	Jumps to U if $(A_a) \neq \pm 0$; goes to NI if $(A_a) = \pm 0$
74	02		JP	Jump Positive	Jumps to U if $(A_a)_{35} = 0$; goes to NI if $(A_a)_{35} = 1$
74	03		JN	Jump Negative	Jumps to U if $(A_a)_{35} = 1$; goes to NI if $(A_a)_{35} = 0$
74	04		J JK	Jump Jump Key	Jumps to U if a = 1 $\boxed{\text{AND}}$ (JUMP KEY selected by A_a) = 1; goes to NI if neither is true
74	05		HJ HKJ	Halt Jump Halt Keys and Jump	Stops if [a = 0 $\boxed{\text{OR}}$ if (a-field $\boxed{\text{AND}}$ set STOP SELECT control circuits) $\neq$ 0];$\boxed{\text{AND}}$ $D_2 = 0$; on restart or continuation jumps to U
74	06		NOP	No Operation	Proceeds to Next Instruction
74	07		AAIJ	Allow All Interrupts and Jump	Allows all interrupts and jump to U
74	10		JNB	Jump No Low Bit	Jumps to U if $(A_a)_0 = 0$; goes to NI if $(A_a)_0 = 1$
74	11		JB	Jump Low Bit	Jumps to U if $(A_a)_0 = 1$; goes to NI if $(A_a)_0 = 0$

Table C-2. Instruction Repertoire (continued)

f	j	a	Mnemonic	Instruction	Description
74	12		JMGI	Jump Modifier Greater and Increment	Jumps to U if $(X_a)_{17-0} > 0$; goes to NI if $(X_a)_{17-0} \leq 0$; always $(X_a)_{17-0} + (X_a)_{35-18} \rightarrow X_{a\ 17-0}$
74	13		LMJ	Load Modifier and Jump	Relative P + 1 $\rightarrow (X_a)_{17-0}$; jump to U
74	14	00	J0	Jump Overflow	Jumps to U if D1 = 1; goes to NI if D1 = 0
74	14	01	JFU	Jump Floating Underflow	Jumps to U if D21 = 1, clears D21; goes to NI if D21 = 0
74	14	02	JFO	Jump Floating Overflow	Jumps to U if D22 = 1, clears D22; goes to NI if D22 = 0
74	14	03	JDF	Jump Divide Fault	Jumps to U if D23 = 1, clears D23; goes to NI if D23 = 0
74	14	04–17	–	Invalid Code	Causes Invalid Instruction interrupt to MSR + 221_8
74	15	00	JNO	Jump No Overflow	Jumps to U if D1 = 0; goes to NI if D1 = 1
74	15	01	JNFU	Jump No Floating Underflow	Jumps to U if D21 = 0; goes to NI if D21 = 1; clears D21
74	15	02	JNFO	Jump No Floating Overflow	Jumps to U if D22 = 0; goes to NI if D22 = 1; clears D22
74	15	03	JNDF	Jump No Divide Fault	Jumps to U if D23 = 0; goes to NI if D23 = 1; clears D23
74	15	04–17	–	Invalid Code	Causes Invalid Instruction interrupt to MSR + 221_8
74	16		JC	Jump Carry	Jumps to U if D0 = 1; goes to NI if D0 = 0
74	17		JNC	Jump No Carry	Jumps to U if D0 = 0; goes to NI if D0 = 1
75	00		–	Invalid Code	Causes IOU to return a condition code of 3 to the CPU, indicating instruction not available
75	01		SIOF	Start I/O Fast Release	Initiates operation on subchannel specified by bit 00 through 15 of CAW

Table C-2. Instruction Repertoire (continued)

f	j	a	Mnemonic	Instruction	Description
75	02		–	Invalid Code	Causes IOU to return a condition code of 3 to the CPU, indicating instruction not available
75	03		TSC	Test Subchannel	Interrogates the channel and subchannel
75	04		HDV	Halt Device	Terminates current operation on channel and subchannel.
75	05		HCH	Halt Channel	Terminates current operation on channel.
75	06,07		–	Invalid Code	Causes Invalid Instruction interrupt to MSR + 221_8.
75	10		LCR	Load Channel Register	Loads the interrupt mask register or load the channel base register
75	11–17		–	Invalid Code	Causes Invalid Instruction interrupt to MSR + 221_8
76	00		FA	Floating Add	$(A_a) + (U) \rightarrow A_a$; RESIDUE $\rightarrow A_{a+1}$ if D17 = 1
76	01		FAN	Floating Add Negative	$(A_a) - (U) \rightarrow A_a$; RESIDUE $\rightarrow A_{a+1}$ if D17 = 1
76	02		FM	Floating Multiply	$(A_a) * (U) \rightarrow A_a$ (and A_{a+1} if D17 = 1)
76	03		FD	Floating Divide	$(A_a) \div (U) \rightarrow A_a$; REMAINDER $\rightarrow A_{a+1}$ if D17 = 1
76	04		LUF	Load and Unpack Floating	$\mid (U) \mid_{34-27} \rightarrow A_{a\ 7-0}$, zero fill; $(U)_{26-00} \rightarrow A_{a+1 \text{ bits } 26-00}$, sign fill
76	05		LCF	Load and Convert to Floating	$(U)_{35} \rightarrow A_{a+1 \text{ bit } 35}$, [NORMALIZED $(U)]_{26-0} \rightarrow A_{a+1 \text{ bits } 26-0}$; if $(U)_{35} = 0$, $(A_a)_{7-0} \pm$ NORMALIZING COUNT $\rightarrow A_{a+1 \text{ bits } 34-27}$; if $(U)_{35} = 1$, ones complement of [$(A_a)_{7-0} \pm$ NORMALIZING COUNT] $\rightarrow A_{a+1 \text{ bits } 34-27}$
76	06		MCDU	Magnitude of Characteristic Difference to Upper	$\mid \mid (A_a) \mid_{35-27} - \mid (U) \mid_{35-27} \mid \rightarrow A_{a+1 \text{ bits } 8-0}$; zeros $\rightarrow A_{a+1 \text{ bits } 35-9}$
76	07		CDU	Characteristic Difference to Upper	$\mid (A_a) \mid_{35-27} - \mid (U) \mid_{35-27} \rightarrow A_{a+1 \text{ bits } 8-0}$; sign bits $\rightarrow A_{a+1 \text{ bits } 35-9}$

Table C-2. Instruction Repertoire (continued)

f	j	a	Mnemonic	Instruction	Description
76	10		DFA	Double-Precision Floating Add	$(A_a, A_{a+1}) + (U, U+1) \rightarrow A_a, A_{a+1}$
76	11		DFAN	Double-Precision Floating Add Negative	$(A_a, A_{a+1}) - (U, U+1) \rightarrow A_a, A_{a+1}$
76	12		DFM	Double-Precision Floating Multiply	$(A_a, A_{a+1}) * (U, U+1) \rightarrow A_a, A_{a+1}$
76	13		DFD	Double-Precision Floating Divide	$(A_a, A_{a+1}) \div (U, U+1) \rightarrow A_a, A_{a+1}$
76	14		DFU	Double Load and Unpack Floating	$\vert (U, U+1) \vert_{70-60} \rightarrow A_{a10-0}$, zero fill; $(U, U+1)_{59-36} \rightarrow A_{a+1 \text{ bits } 23-0}$, sign fill; $(U, U+1)_{35-0} \rightarrow A_{a+2}$
76	15		DFP, DLCF	Double Load and Convert to Floating	$(U)_{35} \rightarrow A_{a+1 \text{ bit } 35}$; [NORMALIZED $(U, U+1)]_{59-0} \rightarrow A_{a+1 \text{ bits } 23-0}$ and A_{a+2}; if $(U)_{35} \neq 0$, $(A_a)_{10-0} \pm$ NORMALIZING COUNT $\rightarrow A_{a+1 \text{ bits } 34-24}$; if $(U)_{35} = 1$, ones complement of $[(A_a)_{10-0} \pm$ NORMALIZING COUNT] $\rightarrow A_{a+1 \text{ bits } 34-24}$
76	16		FEL	Floating Expand and Load	If $(U)_{35} = 0$; $(U)_{35-27} + 1600_8 \rightarrow A_{a\ 35-24}$; if $(U)_{35} = 1$; $(U)_{35-27} - 1600_8 \rightarrow A_{a\ 35-24}$, $(U)_{26-3} \rightarrow A_{a\ 23-0}$; $(U)_{2-0} \rightarrow A_{a+1 \text{ bits } 35-33}$; $(U)_{35} \rightarrow A_{a+1 \text{ bits } 32-0}$
76	17		FCL	Floating Compress and Load	If $(U)_{35} = 0$; $(U)_{35-24} - 1600_8 \rightarrow A_{a\ 35-27}$; if $(U)_{35} = 1$; $(U)_{35-24} + 1600_8 \rightarrow A_{a\ 35-27}$; $(U)_{23-0} \rightarrow A_{a\ 26-3}$; $(U+1)_{35-33} \rightarrow A_{a\ 2-0}$
77	0-17		-	Invalid Code	Causes Invalid Instruction interrupt to MSR + 221_8

Table C-3. Octal vs Mnemonic Instruction Code

First Digit	Function Code - Second Digit							
	0	1	2	3	4	5	6	7
0		S SA	SN SNA	SM SMA	S SR	(1)	S SX	(see below)
1	L LA	LN LNA	LM LMA	LNMA	A AA	AN ANA	AM AMA	ANM ANMA
2	AU	ANU	BT	L LR	A AX	AN ANX	LXM	L LX
3	MI	MSI	MF		DI	DSF	DF	(see below)
4	OR	XOR	AND	MLU	TEP	TOP	LXI	TLEM TNGM
5	TZ	TNZ	TE	TNE	TLE TNG	TG	TW	TNW
6	TP	TN	SE	SNE	SLE SNG	SG	SW	SNW
7	JGD	(see below)						

Funct. Code	First j Digit	Second j Digit							
		0	1	2	3	4	5	6	7
07	0	ADE	DADE	SDE	DSDE	LAQW	SAQW	DEI	DDEI
	1	IDE	DIDE	LDJ	LIJ	LPD	SPD		LBJ
37	0								
	1	BIM	BIC	BMTC	BICL	BIML	BDE	DEB	EDDE
71	0	MSE	MSNE	(2)	MSG	MSW	MSNW	MASL	MASG
	1	DA	DAN	DS	DL	DLN	DLM	DJZ	DTE
72	0	IMI	SLJ	JPS	JNS	AH	ANH	AT	ANT
	1	EX	ER	BN	PAIJ	BBN	TRA	SRS	LRS
73	0	SSC	DSC	SSL	DSL	SSA	DSA	LSC	DLSC
	1	LSSC	LDSC	LSSL	LDSL	(3)	(4)		(5)
74	0	JZ	JNZ	JP	JN	J,JK	HJ,HKJ	NOP	AAIJ
	1	JNB	JB	JMGI	LMJ	(6)	(7)	JC	JNC
75	0		SIOF		TSC	HDV	HCH		
	1	LCR							
76	0	FA	FAN	FM	FD	LUF	LCF	MCDU	CDU
	1	DFA	DFAN	DFM	DFD	DFU	(8)	FEL	FCL

NOTES:

1. *SZ, SNZ, SP1, SN1, SFS, SFZ, SAS, SAZ, INC, DEC, INC2, DEC2, ENZ*
2. *MSLE*, MSNG**
3. *LDC, MDC*
4. *SIL, LBRX, LQT, IIIX, SPID, Clear SC, LB, LL, LAE, SQT, LD, SD, UR*
5. *TS, TSS, TCS*
6. *JO, JFU, JFO, JDF*
7. *JNO, JNFU, JNFO, JNDF*
8. *DFP*, DLCF**

* *These are different mnemonics for the same instruction.*

Appendix D. Code Conversions

D.1. ASCII and Fieldata Code Conversion Tables

Codes, which also represent collating sequence, are given in octal in Tables D-1 and D-2.

ASCII codes from 00_8 to 37_8 are used for communications. They are format, separator, and control characters. These are not converted into Fieldata.

The ASCII symbols represented by codes 40_8 to 137_8 are converted into the identical Fieldata symbols, except that the quotation marks symbol (42_8) is converted into a lozenge (76_8), the circumflex (136_8) is converted into a delta (04_8), and the underscore (137_8) is converted into a not equal sign (77_8).

There are no remaining unique Fieldata symbols into which to convert the balance of the ASCII symbols, represented by codes 140_8 to 177_8, (these codes are shown boxed in Table D-2), so most of these codes are "folded" over codes 100_8 to 137_8 (by clearing bit 5, which amounts to subtracting 40_8). This means that ASCII codes 101_8 (A) and 141_8 (a), for example, are both translated as if they were code 101_8 (converted to Fieldata 06_8 for A). Two exceptions to this general rule are the ASCII opening brace (173_8) and closing brace (175_8) that are converted to Fieldata question mark (54_8) and exclamation point (55_8), respectively, to satisfy overpunch sign considerations. The Operating System folds all codes from 140_8 to 177_8.

Table D-1. Fieldata to ASCII Code Conversion

Fieldata Code (Octal)	Fieldata 80-Column Card Code	Symbol	ASCII	
			Octal Code	Symbol
00	7–8	@	100	@
01	12–5–8	[	133	[
02	11–5–8	]	135	]
03	12–7–8	#	43	#
04	11–7–8	Δ	136	^
05	(blank)	(space)	40	(space)
06	12–1	A	101	A
07	12–2	B	102	B
10	12–3	C	103	C
11	12–4	D	104	D
12	12–5	E	105	E
13	12–6	F	106	F
14	12–7	G	107	G
15	12–8	H	110	H
16	12–9	I	111	I
17	11–1	J	112	J
20	11–2	K	113	K
21	11–3	L	114	L
22	11–4	M	115	M
23	11–5	N	116	N
24	11–6	0	117	O
25	11–7	P	120	P
26	11–8	Q	121	Q
27	11–9	R	122	R
30	0–2	S	123	S
31	0–3	T	124	T
32	0–4	U	125	U
33	0–5	V	126	V
34	0–6	W	127	W
35	0–7	X	130	X
36	0–8	Y	131	Y
37	0–9	Z	132	Z
40	12–4–8	)	51	)
41	11	– (minus)	55	– (minus)
42	12	+	53	+
43	12–6–8	<	74	<
44	3–8	=	75	=
45	6–8	>	76	>

Table D-1. Fieldata to ASCII Code Conversion (continued)

Fieldata Code (Octal)	Fieldata 80-Column Card Code	Symbol	ASCII	
			Octal Code	Symbol
46	2-8	&	46	&
47	11-3-8	$	44	$
50	11-4-8	*	52	*
51	0-4-8	(	50	(
52	0-5-8	%	45	%
53	5-8	: (colon)	72	: (colon)
54	12-0	?	77	?
55	11-0	!	41	!
56	0-3-8	,(comma)	54	,(comma)
57	0-6-8	\	134	\
60	0	0	60	0
61	1	1	61	1
62	2	2	62	2
63	3	3	63	3
64	4	4	64	4
65	5	5	65	5
66	6	6	66	6
67	7	7	67	7
70	8	8	70	8
71	9	9	71	9
72	4-8	'(apostrophe)	47	'(apostrophe)
73	11-6-8	;	73	;
74	0-1	/	57	/
75	12-3-8	.(period)	56	.(period)
76	0-7-8	⌑	42	"
77	0-2-8	≠ or stop	137	_

Table D-2. ASCII to Fieldata Code Conversion

ASCII		System Console			Fieldata	
Octal Code	Symbol	Keyboard Symbol	CRT Symbol	Incremental Printer Symbol	Octal Code	Symbol
40	SP	(space bar)	(space)	(space)	05	(space)
41	!	!	!	!	55	!
42	"	"	"	"	76	¤
43	#	#	#	#	03	#
44	$	$	$	$	47	$
45	%	%	%	%	52	%
46	&	&	&	&	46	&
47	' (apos.)	' (apos.)	' (apos.)	' (apos.)	72	' (apos.)
50	(	(	(	(	51	(
51	)	)	)	)	40	)
52	*	*	*	*	50	*
53	+	+	+	+	42	+
54	, (comma)	, (comma)	, (comma)	, (comma)	56	, (comma)
55	– (minus)	– (minus)	– (minus)	– (minus)	41	– (minus)
56	. (period)	. (period)	. (period)	. (period)	75	. (period)
57	/	/	/	/	74	/
60	0	0	0	0	60	0
61	1	1	1	1	61	1
62	2	2	2	2	62	2
63	3	3	3	3	63	3
64	4	4	4	4	64	4
65	5	5	5	5	65	5
66	6	6	6	6	66	6
67	7	7	7	7	67	7
70	8	8	8	8	70	8
71	9	9	9	9	71	9
72	: (colon)	: (colon)	: (colon)	: (colon)	53	: (colon)
73	;	;	;	;	73	;
74	<	<	<	<	43	<
75	=	=	=	=	44	=
76	>	>	>	>	45	>
77	?	?	?	?	54	?
100	@	@	@	@	00	@
101	A	A	A	A	06	A
102	B	B	B	B	07	B
103	C	C	C	C	10	C
104	D	D	D	D	11	D
105	E	E	E	E	12	E

Table D-2. ASCII to Fieldata Code Conversion (continued)

ASCII		System Console			Fieldata	
Octal Code	Symbol	Keyboard Symbol	CRT Symbol	Incremental Printer Symbol	Octal Code	Symbol
106	F	F	F	F	13	F
107	G	G	G	G	14	G
110	H	H	H	H	15	H
111	I	I	I	I	16	I
112	J	J	J	J	17	J
113	K	K	K	K	20	K
114	L	L	L	L	21	L
115	M	M	M	M	22	M
116	N	N	N	N	23	N
117	O	O	O	O	24	O
120	P	P	P	P	25	P
121	Q	Q	Q	Q	26	Q
122	R	R	R	R	27	R
123	S	S	S	S	30	S
124	T	T	T	T	31	T
125	U	U	U	U	32	U
126	V	V	V	V	33	V
127	W	W	W	W	34	W
130	X	X	X	X	35	X
131	Y	Y	Y	Y	36	Y
132	Z	Z	Z	Z	37	Z
133	[	[	[	[	01	[
134	\	\	\	\	57	\
135	]	]	]	]	60	]
136	^	^	^	^	04	Δ
137	_	_	_	_	77	≠
140	‘	‘	‘	@	00	@
141	a	A	a	A	06	A
142	b	B	b	B	07	B
143	c	C	c	C	10	C
144	d	D	d	D	11	D
145	e	E	e	E	12	E
146	f	F	f	F	13	F
147	g	G	g	G	14	G
150	h	H	h	H	15	H
151	i	I	i	I	16	I
152	j	J	j	J	17	J
153	k	K	k	K	20	K

Table D-2. ASCII to Fieldata Code Conversion (continued)

ASCII		System Console			Fieldata	
Octal Code	Symbol	Keyboard Symbol	CRT Symbol	Incremental Printer Symbol	Octal Code	Symbol
154	l	L	l	L	21	L
155	m	M	m	M	22	M
156	n	N	n	N	23	N
157	o	O	o	O	24	O
160	p	P	p	P	25	P
161	q	Q	q	Q	26	Q
162	r	R	r	R	27	R
163	s	S	s	S	30	S
164	t	T	t	T	31	T
165	u	U	u	U	32	U
166	v	V	v	V	33	V
167	w	W	w	W	34	W
170	x	X	x	X	35	X
171	y	Y	y	Y	36	Y
172	z	Z	z	Z	37	Z
173	{	{	{	[	54	?
174	:	:	:	\	57	\
175	}	}	}	]	55	!
176	~	~	~	^	04	Δ
177	DEL	(no key)	'/,	_	77	≠

D.2. Special Characters in ASCII

The special characters in ASCII are:

SP	designates space, which is normally nonprinting.
DEL	designates delete, and has a code of all 1 bits. This code eliminates the previous character - even on paper tape or other nonerasable medium.

Definitions of the 32 ASCII control characters, codes 00_8 to 37_8:

00 NUL	Null - all zero character that serves as time fill
01 SOH	Start of heading
02 STX	Start of text
03 ETX	End of text
04 EOT	End of transmission
05 ENQ	Enquire - "Who Are You?"
06 ACK	Acknowledge - "Yes"

07	BEL	Bell - human attention required	
10	BS	Backspace	format effectors for printing or punching
11	HT	Horizontal tabulation	
12	LF	Line feed	
13	VT	Vertical tabulation	
14	FF	Form feed	
15	CR	Carriage return	
16	SO	Shift out - nonstandard code follows	
17	SI	Shift in - return to standard code	
20	DLE	Data link escape - change limited data communications control	
21	DC1	Device control for turning on or off auxiliary devices	
22	DC2		
23	DC3		
24	DC4		
25	NAK	Negative acknowledge - "No"	
26	SYN	Synchronous idle - from which to achieve synchronism	
27	EBT	End of transmission block - relates to physical communications block	
30	CAN	Cancel previous data	
31	EM	End of medium - end of used, or wanted, portion of information	
32	SUB	Substitute character for one in error	
33	ESC	Escape - for code extension - change some character interpretations	
34	FS	File separator	These information separators are ordered in descending hierarchy. They are followed by ASCII 40_8 (space), which can also be thought of as a word separator.
35	GS	Group separator	
36	RS	Record separator	
37	US	Unit separator	

Index

SPERRY

USER COMMENTS

We will use your comments to improve subsequent editions.

NOTE: Please do not use this form as an order blank.

(Document Title)

(Document No.)	*(Revision No.)*	*(Update Level)*

Comments:

From:

(Name of User)

(Business Address)

Fold on dotted lines, and mail. (No postage is necessary if mailed in the U.S.A.)
Thank you for your cooperation

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 21 BLUE BELL, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

SPERRY CORPORATION

ATTN: Documentation Quality Control Group
C/O SYSTEM PUBLICATIONS

P.O. BOX 500
BLUE BELL, PENNSYLVANIA 19422-9990

FOLD