Xerox 530 Computer

Reference Manual

90 19 60B

XEROX 530 INSTRUCTIONS (NAMES)

Instruction Name	Syntax† Command	Syntax† Argument	Format†† First Word	Format†† Second Word	Page
Acknowledge I/O Interrupt	AIO		0001 0000 0101 0000		58
Add	ADD	*a, x, b	1010 RIXS D		21
Add Word	AW, r	*a, x, b	0001 0000 10001 GR	1010 RIXS D	35
And, Logical (one-word instruction)	AND	*a, x, b	1001 RIXS D		21
And, Logical (two-word instruction)	AND, r	*a, x, b	0001 0000 10001 GR	1001 RIXS D	36
Branch	B	*a, x, b	0100 RIXS D		24
Branch if Accumulator Negative	BAN	a	0110 111S D		24
Branch if Accumulator Zero	BAZ	a	0110 010S D		25
Branch if Extended Accumulator Negative	BEN	a	0110 110S D		25
Branch if No Carry	BNC	a	0110 001S D		25
Branch if No Overflow	BNO	a	0110 000S D		25
Branch on Incrementing Index	BIX	a	0110 011S D		25
Branch on Incrementing Index and No Carry	BXNC	a	0110 101S D		25
Branch on Incrementing Index and No Overflow	BXNO	a	0110 100S D		25
Compare	CP	*a, x, b	1101 RIXS D		24
Compare Arithmetic Field (optional)	CAF, rx, sx	*a, x, b	0001 0000 10 RX SX	1101 RIXS D	51
Compare Double	CPD	*a, x, b	0001 0000 10 010 110	1101 RIXS D	38
Compare Logical Field (optional)	CLF, rx, sx	*a, x, b	0001 0000 10 RX SX	0101 RIXS D	50
Compare Word	CW, r	*a, x, b	0001 0000 10001 GR	1101 RIXS D	36
Divide	DIV	*a, x, b	0101 RIXS D		28
Double Add	DAD	*a, x, b	0001 0000 10 010 110	1010 RIXS D	38
Double Subtract	DSB	*a, x, b	0001 0000 10 010 110	1011 RIXS D	38
Floating Add	FAD	*a, x, b	1010 RIXS D		42
Floating Compare	FCP	*a, x, b	1101 RIXS D		43
Floating Divide	FDV	*a, x, b	0101 RIXS D		43
Floating Load optional	FLD	*a, x, b	1000 RIXS D		42
Floating Multiply	FMP	*a, x, b	0011 RIXS D		43
Floating Store	FST	*a, x, b	1110 RIXS D		42
Floating Subtract	FSB	*a, x, b	1011 RIXS D		42
Halt Input/Output	HIO		0001 0000 0100 1000		58
Increment Memory	IM	*a, x, b	1111 RIXS D		21
Load Arithmetic Field (optional)	LAF, rx, sx	*a, x, b	0001 0000 10 RX SX	1001 RIXS D	49
Load Double	LDD	*a, x, b	0001 0000 10 010 110	1000 RIXS D	37
Load Index	LDX	*a, x, b	1100 RIXS D		21
Load Logical Field (optional)	LLF, rx, sx	*a, x, b	0001 0000 10 RX SX	1000 RIXS D	49
Load Multiple	LDM	*a, x, b, fr, nr	0001 0000 10 XXX YYY	1000 RIXS D	37
Load Register A	LDA	*a, x, b	1000 RIXS D		20
Load Word	LW, r	*a, x, b	0001 0000 10001 GR	1000 RIXS D	34
Multiply	MUL	*a, x, b	0011 RIXS D		28
Read Direct	RD	*a, x, b	0001 RIXS D		32
Register Add	RADD	*s, d	0111 1100 0 DR I/S SR		26
Register Add and Carry	RADDC	*s, d	0111 1110 0 DR I/S SR		27
Register Add and Increment	RADDI	*s, d	0111 1101 0 DR I/S SR		27
Register AND	RAND	*s, d	0111 0000 0 DR I/S SR		27
Register AND and Carry	RANDC	*s, d	0111 0010 0 DR I/S SR		28
Register AND and Increment	RANDI	*s, d	0111 0001 0 DR I/S SR		27
Register Clear, Add and Carry	RCLAC	*s, d	0111 1110 1 DR I/S SR		28
Register Clear, Add and Increment	RCLAI	*s, d	0111 1101 1 DR I/S SR		28
Register Clear and Add	RCLA	*s, d	0111 1100 1 DR I/S SR		28
Register Copy	RCPY	*s, d	0111 0100 1 DR I/S SR		26
Register Copy and Carry	RCPYC	*s, d	0111 0110 1 DR I/S SR		27
Register Copy and Increment	RCPYI	*s, d	0111 0101 1 DR I/S SR		27
Register Exclusive OR	REOR	*s, d	0111 1000 0 DR I/S SR		27
Register Exclusive OR and Carry	REORC	*s, d	0111 1010 0 DR I/S SR		28
Register Exclusive OR and Increment	REORI	*s, d	0111 1001 0 DR I/S SR		27
Register OR	ROR	*s, d	0111 0100 0 DR I/S SR		27
Register OR and Carry	RORC	*s, d	0111 0110 0 DR I/S SR		28
Register OR and Increment	RORI	*s, d	0111 0101 0 DR I/S SR		27
Sense Left Bit of Field (optional)	SLF, rx, sx	*a, x, b	0001 0000 10 RX SX	1111 RIXS D	51
Set Floating Mode (optional)	SFM		0001 0000 1001 1110		41
Shift	S	*a, x, b	0010 RIXS D		22
Shift Arithmetic Left Double	SALD	c, x, b	0010 0000 101 count		23
Shift Arithmetic Left Single	SALS	c, x, b	0010 0000 001 count		23
Shift Arithmetic Right Double	SARD	c, x, b	0010 0000 100 count		22
Shift Arithmetic Right Single	SARS	c, x, b	0010 0000 000 count		22
Shift Circular Left Double	SCLD	c, x, b	0010 0000 111 count		24
Shift Circular Left Single	SCLS	c, x, b	0010 0000 011 count		24
Shift Circular Right Double	SCRD	c, x, b	0010 0000 110 count		23
Shift Circular Right Single	SCRS	c, x, b	0010 0000 010 count		23
Start Input/Output	SIO		0001 0000 0100 0001		57
Store Double	STD	*a, x, b	0001 0000 10 010 110	1110 RIXS D	37
Store Field (optional)	STF, rx, sx	*a, x, b	0001 0000 10 RX SX	1010 RIXS D	49
Store Multiple	STM	*a, x, b, fr, nr	0001 0000 10 XXX YYY	1110 RIXS D	37
Store Ones Field (optional)	SOF, rx, sx	*a, x, b	0001 0000 10 RX SX	1100 RIXS D	50
Store Register A	STA	*a, x, b	1110 RIXS D		21
Store Word	STW	*a, x, b	0001 0000 10001 GR	1110 RIXS D	35
Store Zero Field (optional)	STZ, rx, sx	*a, x, b	0001 0000 10 RX SX	1011 RIXS D	50
Subtract	SUB	*a, x, b	1011 RIXS D		21
Subtract Word	SW, r	*a, x, b	0001 0000 10001 GR	1011 RIXS D	35
Test Device	TDV		0001 0000 0100 0100		58
Test Input/Output	TIO		0001 0000 0100 0010		57
Write Direct	WD	*a, x, b	0000 RIXS D		29

†Refer to the Xerox Extended Symbol/LN, OPS Reference Manual, 90 10 52, for further information on symbolic notation.

††Except for using binary notation (rather than hexadecimal) to represent fixed fields, the format is the same as described in Chapters 3 and 4.

Xerox Corporation
701 South Aviation Boulevard
El Segundo, California 90245
213 679-4511

XEROX

Xerox 530 Computer

Reference Manual

90 19 60B

September 1973

Price: $3.75

Printed in U.S.A.

REVISION

This publication, 90 19 60B, is a revision of the Xerox 530 Computer Reference Manual, 90 19 60A. It incorporates Publication Revision Package, 90 19 60A-2(4/73). The major change to the manual is the addition of Appendix B, "Instruction Timing". Other changes are indicated by a vertical line in the margin of the affected page.

RELATED PUBLICATIONS

Title	Publication No.
Xerox Symbol/Reference Manual	90 10 51
Xerox Extended Symbol/Reference Manual	90 10 52
Xerox Computer Systems/Interface Design Manual	90 09 73

Manual Content Codes: BP – batch processing, LN – language, OPS – operations, RP – remote processing, RT – real-time, SM – system management, TS – time-sharing, UT – utilities.

CONTENTS

APPENDIXES

FIGURES

TABLES

Xerox 530 Computer System

1. XEROX 530 COMPUTER SYSTEM

INTRODUCTION

This high-speed low-cost system is an integrated combination of sophisticated hardware technology (i.e., large- and medium-scale integrated circuits), advanced microprogramming techniques, and field-proven existing software. The Xerox 530 has advantages that are usually found only in large computing systems. It is well suited for a multi-usage environment, both real-time and general-purpose applications.

A basic system includes a central processor, main memory, and an independent input/output processor. The basic system may be expanded easily to accommodate the user's requirements (see "Standard and Optional Features"). The CPU's basic instruction repertoire may be increased to include the optional floating-point and field-addressing instructions. Main memory may be expanded by adding more memory modules. Input/output capability may be increased by adding a second input/output processor and additional device controllers and peripheral devices. A large complement of peripheral devices is available for cost effective input/output.

Concurrent multiprogramming capability permits the user to operate one or more fully protected, real-time programs in the foreground while concurrently operating a general-purpose program in the background. Overhead in switching from one task to another is minimized because both hardware and software are specifically designed for rapid context switching. A hardware register permits the software to generate reentrant code efficiently. Therefore, routines common to several programs, whether in foreground or background, need to be stored in memory only once.

The comprehensive field-proven programming package (assemblers, compilers, mathematical and utility routines, and applications) utilizes advanced features in the hardware. These programming systems are easy-to-use programming tools that increase productivity and allow user programs to be written more quickly at lower cost.

Existing Sigma 2 or 3 computer programs may be run on a Xerox 530 computer system. The compatibility of the modular software eliminates reprogramming or requires only minimal upgrading.

Optional field addressing instructions enable efficient operation upon any group of from 1 to 16 contiguous bits in memory without regard to word boundaries. Effectively used, the system provides a bit and byte manipulating capability, a general pushdown stack facility, and the ability to effectively operate on logical structures such as tables and strings.

General register instruction capability puts the result of executing one of a prescribed set of arithmetic and logical instructions (see Chapter 3) into a designated general register rather than into the accumulator. This mode of operation permits efficiencies in both code generation and execution times.

To enhance computations using scientific notations, optional floating-point hardware is available.

Xerox 530 systems provide significant reliability, maintainability, and availability improvements over other small- or medium-size computer systems. A remote assistance terminal connection with special software permits remote assistance as an integral part of maintenance.

GENERAL CHARACTERISTICS

The Xerox 530 computer system functions efficiently in a variety of computing environments and applications. Its operating characteristics and features are outlined below:

- Both word and byte organization of memory.
- Memory expandable from 8K words to 64K words in increments of 8K words.
- General-purpose registers to control program operations (all are available to the program). They provide
  - Hardware index registers for preindexing (base address), postindexing, or both (double indexing).
  - Hardware register for subroutine linkages.
  - Double precision accumulator.
  - Program address register.
  - Temporary storage register.
- Rapid context switching to preserve computer environment when switching from one program to another, including automatic status preservation at interrupt.
- Up to two independent Input/Output Processors (IOPs) for high-volume data I/O operations.
  - Up to 28 fully automatic I/O channels operating concurrently with one another.
  - I/O data chaining, for scatter-read and gather-write operations.

- Up to two direct memory adapters (optional), each having a maximum information transfer rate of approximately 625,000 words per second.

- Direct input/output of a full word (in parallel) without the use of an I/O channel (optional).

- A real-time priority interrupt system that features
  - Ten internal interrupt levels and up to 30 external interrupt levels. All external and most internal levels can be individually armed, enabled, and triggered by program control.
  - Automatic identification and fast response time.
  - Machine fault register which collects fault status enabling program retrieval.
  - Power Monitor for automatic shutdown in event of power failure and resumption of processing when power returns.
  - System protection that includes both memory write protection and operation protection for foreground programs.
  - Two real-time clocks (one with a choice of resolution) for independent time bases.

- An extensive repertoire that includes these classes of instructions:
  - Memory reference.
  - Conditional branch.
  - Copy (register-to-register).
  - Direct control.
  - Multiply/Divide.
  - Double precision, operations on 32-bit operands.
  - General register capability, which places the result of executing one of a prescribed set of arithmetic and logical instructions into a designated general register rather than in the accumulator.
  - Floating-point (optional).
  - Field addressing (optional), which permits operation on any group of from 1 to 16 contiguous bits in memory without regard to word boundaries, providing bit and byte manipulation, general pushdown stack capability, and ability to operate on logical structures such as tables and strings.

- Instruction characteristics include
  - Only one word of storage required for most instructions.
  - Two levels of indexing and one level of indirect addressing may be invoked individually or simultaneously.
  - Relative addressing (forward and backward).
  - Use of index register 2(B) as a base address register.
  - Direct reference of up to 1024 addresses; 256 addresses beginning with location zero, 256 addresses beginning with the base address, 256 addresses beginning with the current instruction location (relative forward), and 256 addresses backward from the current instruction (relative backward).

- Comprehensive, modular software that expands in capability and speed as the system grows, with no reprogramming required. Existing, field-proven Sigma 2 or 3 computer programs may be run on a Xerox 530 system.
  - Basic Control Monitor (BCM) Operating System for smaller systems including Symbol and Basic FORTRAN.
  - Real-Time Batch Monitor (RBM) Operating System including Extended Symbol, Basic FORTRAN IV, ANS FORTRAN IV, RPG, and SORT.
  - General loading programs.
  - Utility routines.
  - Mathematical routines.
  - General Debug for symbolic program troubleshooting.
  - Concordance program for documentation.
  - System Generation program for creating installation master.

- Standard and special-purpose peripheral equipment including
  - Rapid Access Data (RAD) files: capacities of .75, 1.5, or 3.0 million bytes per storage unit; transfer rate of 188,000 bytes per second; average access time of 17 milliseconds.
  - Magnetic tape units: IBM compatible; 7-track units operating at 37.5 inches per second with transfer rates up to 20,800 bytes per second; 9-track units operating at 75 inches per second with transfer rates up to 60,000 bytes per second.

- Card equipment: reading speeds up to 200 or 400 cards per minute; punching speeds up to 100 cards per minute.

- Line printers: fully buffered with speeds from 310 to 1100 lines per minute; up to 132 print positions and up to 91 characters.

- Keyboard/printers: ten characters per second; also available with paper tape reader (20 characters per second) and punch (10 characters per second).

- Paper tape equipment: readers with speeds up to 300 characters per second; punches with speeds up to 120 characters per second.

- Graph plotters: digital incremental, providing drift-free plotting in two axes in up to 300 steps per second at speeds from 30 millimeters to three inches per second.

- Data communications equipment: complete line of character-oriented and message-oriented equipment to connect remote user terminals to the computer system via common carrier lines and local terminals directly.

- Removable disk storage: capacities from 24.5 million to 196.6 million bytes; transfer rate of 312,000 bytes per second; average access time of 87.5 milliseconds; one- or two-byte data paths; device pooling.

- Multiprocessing equipment: exchange of critical control and data signals between CPUs and between IOPs on a real-time basis; sharing of I/O devices attached to IOPs; concurrent control of external (DIO) devices.

REAL-TIME AND MULTIUSAGE FEATURES

Real-time applications are characterized by a need for (1) hardware that provides quick response to an external environment, (2) sufficient speed to keep up with the real-time process itself, and (3) input/output flexibility to handle a wide variety of data types at varying speeds.

Multiusage applications, in the context of this computer system, are defined as the combination of foreground (real-time) and background processing techniques into one system. One of the most difficult general computing problems is the real-time application with its severe requirements for extreme speed and capacity. Since the computer system design is on a real-time base, it is well qualified for a mixture of applications in a multiusage environment. Many hardware features that are valuable for real-time applications are equally useful in background processing, but in different ways. The major features that make this system suitable for multiusage applications are described in the following paragraphs.

Multilevel, Priority Interrupt System. In a multiusage environment, many elements operate simultaneously and asynchronously. Thus, an efficient priority interrupt system is essential. The source of each interrupt is automatically identified and responded to according to its priority. For further flexibility, each level can be individually disarmed (to discontinue input acceptance) and disabled (to defer responses). Use of the disarm/disable feature makes programmed dynamic reassignment of priorities quick and easy, even while a real-time process is in progress.

Programs that deal with interrupt signals from special equipment often require checkout before the equipment is actually available. To permit simulating this special equipment, any external interrupt level can be "triggered" by the CPU through execution of a single instruction. This capability is also useful in establishing a modified hierarchy of responses. For example, in responding to a high priority interrupt after the urgent processing is completed, it may be desirable to assign a lower priority to the remaining portion so that the interrupt system is free to respond to other critical stimuli. The interrupt routine can accomplish this by "triggering" a lower priority level, which processes the remaining data only after other interrupts have been handled.

READ DIRECT and WRITE DIRECT instructions (described in Chapter 3) allow the program to acknowledge an I/O interrupt condition, read the status of interrupts, and control the individual levels of the priority interrupt system.

Nonstop Operation. When connected to special devices (on a ready/resume basis), the computer may be excessively delayed if the specific device does not respond quickly. A built-in watchdog timer assures that the computer cannot be delayed for an excessive length of time.

Real-Time Clocks. Many real-time functions must be timed to occur at specific instants. Other timing information is also needed; for example, elapsed time since a given event, or the current time of day. The computer system provides two real-time clocks, one with varying degrees of resolution, to meet these needs. These clocks also facilitate handling separate time bases and relative time priorities.

Rapid Context Switching. When responding to a new set of interrupt-initiated circumstances, a computer system must preserve the current operating environment for continuance later, while setting up the new environment. This changing of environments must be done quickly, with a minimum of "overhead" time costs. In this computer system, relevant information about the current environment (instruction address, status indicators, etc.) is retained in a 32-bit program status doubleword (PSD). When an interrupt occurs, the current PSD is automatically stored at an arbitrary location in memory; and the interrupt-servicing routine begins, following the location into which the PSD is stored. At the end of the interrupt-servicing routine, the PSD is restored and the interrupt level cleared.

Memory Protection. Both foreground (real-time) and background programs can be run concurrently in this computer system, since a real-time program is protected against destruction or alteration by an unchecked background program. The protect feature prevents accessing protected areas of memory for specified combinations of reading, writing, and instruction acquisition. The feature guarantees that protected memory cannot be written into by a program residing in unprotected memory. This feature also prevents background programs from executing instructions that could change the I/O system or the protection system. The protection pattern can be changed very quickly.

Input/Output. Because of the wide range of capacities and speeds, the computer system simultaneously satisfies the needs of many different application areas economically, both in terms of equipment and programming.

STANDARD AND OPTIONAL FEATURES

The basic Xerox 530 system has the following standard features:

- A CPU that includes
  - Main memory of 8K words.
  - Extended arithmetic unit (including multiply/divide).
  - Processor control panel.
  - Two real-time clocks.
  - Memory protection feature.
  - Interrupt master including 16 levels of interrupt priority.
  - Power Monitor.
- Input/Output Processor (IOP) with 16 channels.
- Remote assistance terminal connection.

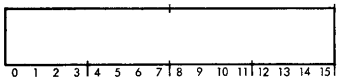
The system may also include the following optional features:

- Memory in 8K increments to a maximum of 64K.
- Floating-point arithmetic instructions.
- Field addressing instructions.
- Up to 24 external interrupt levels (two optional groups of 12).
- External interface (Direct I/O).
- An additional input/output processor with 12 channels.
- Four ASR, KSR Teletype models available (keyboard/printer required as operator's console).
- Up to two Direct Memory Attachments.

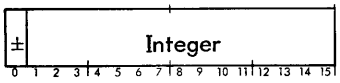
INFORMATION NOMENCLATURE AND FORMATS

The binary digit, or bit, is the most basic unit of digital information. Depending upon the context, a bit may be described by its binary value (0 or 1), status (off or on), condition (false or true), or other dichotomous attributes. A group of bits that are functionally related is commonly referred to as a "field". Except for fields that are used as operands for field addressing instructions, all fields have fixed formats and parameters (length, boundaries, and positional notations). The parameters of operands (containing from 1 to 16 bits) for field address instructions, described in Chapter 3, are defined by the software. Common "fixed" fields are bits, bytes, words, and doublewords.

The parameters of a fixed word, as illustrated, are 16 contiguous bits, a unique position (0 through 15) for each bit, and word boundaries that occur between bit 15 of one word and bit 0 of the next word.



The format of a data word for fixed point arithmetic operation is

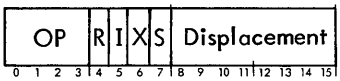


Bit position 0 contains a sign bit which is 0 if the integer is positive or a 1 if the integer is negative. Bit positions 1-15 represent the value of the integer. Bit position 1 is the most significant bit; bit position 15 is the least significant bit. The binary point is assumed to be on the right boundary.

Negative numbers are expressed in the two's complement.

For logical operations, a word is considered to be 16 bits without sign.

The format for a typical one-word instruction is

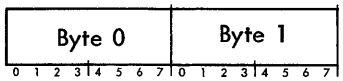


Bit positions 0-3 contain the operation code (OP). The operation code and format for each instruction are described in Chapters 3 or 4.

Bit positions 4-7 (R, I,X, S) comprise an address-control field. Refer to "Effective Address Computation" for further details.

Bit positions 8-15 contain a displacement value. Refer to "Effective Address Computation" and Chapters 3 or 4 for further details.

When the parameters of a byte (eight contiguous bits) are fixed, a byte is either the most significant half of a word (byte 0) or the least significant half of a word (byte 1). Bit positions within a byte are designated as 0 through 7. Byte boundaries occur between bit 7 of one byte and bit 0 of the next byte. Byte boundaries between bytes of different words coincide with the word boundaries.

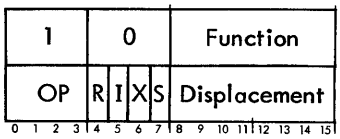


The parameters of a doubleword are always fixed. The individual bits are numbered 0 through 31.

The first 16 bits (0-15) comprise the most significant word and the second 16 bits (16-31) comprise the least significant word. Doubleword boundaries occur between bit 31 of one doubleword and bit 0 of the next doubleword. The general format of a doubleword is

A doubleword is always referred to by the address of the most significant word. As part of a Program Status Doubleword, an I/O Control Doubleword, a Field Descriptor required for field addressing instructions, or as part of doubleword operands, the most significant word of a doubleword may have either an even or odd address.

Field Addressing, General Register, and Multiple Register instructions, as described in Chapter 3, are two-word instruction sequences which have the following format:



For these instructions, the first word is always a READ DIRECT (Mode 0) instruction. As such, the recommended coding for the first word is as illustrated. The format and coding of the second word is similar to that described above for the typical one-word instruction.

A two-word instruction sequence is also used to exit from an interrupt-servicing routine. The format is similar to that illustrated above, except the first word is coded as a WRITE DIRECT (Mode 0).

The specific format (and recommended coding, when applicable) for each instruction is shown in Chapter 3.

Hexadecimal digits (each equivalent to four bits) are commonly used when referring to binary information. Thus, a four-bit field (e.g., operation code) may be expressed as one hexadecimal character, a byte may be expressed as a string of two hexadecimal digits, a word as a string of four hexadecimal digits, and a doubleword as a string of eight hexadecimal digits. The 16 configurations of four bits and corresponding decimal and hexadecimal digits are shown below.

Binary	Decimal	Hexadecimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

Note that within this manual (except for format diagrams) a hexadecimal number is displayed as a string of hexadecimal digits enclosed by single quotes and preceded by the letter "X". For example, the binary number 01011010 is expressed in hexadecimal notation as X'5A'.

Although hexadecimal notation is generally used to denote address and data values, decimal notation (where it is more meaningful or customary) may be used. Decimal/hexadecimal conversions are performed by assembler systems.

2. SYSTEM ORGANIZATION

A Xerox 530 computer system, as illustrated in Figure 1, may be comprised of standard and optional units. A functional description of the main memory, central processor unit (CPU), interrupt system, fault system, and interconnecting buses is given in this chapter. The input/output processors (IOPs) and related input/output instructions are described in Chapter 4. The processor control panel (PCP) and related operating procedures are described in Chapter 5.

BUSES

The various units of the computer system are interconnected by three buses. The "memory" bus connects the Memory Control to all memory modules. The "unit memory" bus is used by all units that require direct access to memory with the exception of the CPU, which connects directly to memory control logic. The "internal Direct I/O (DIO)" bus provides control interconnections between the CPU, interrupt system, external DIO Interface, IOPs, and Direct Memory Adaptors.

MAIN MEMORY

The memory system, which operates synchronously with the other central system components, is composed of magnetic core memory modules (CMM). The storage capacity of each memory module is 8K words, with each word consisting of 16 data bits plus two parity bits (one parity bit for each byte of the word). The memory capacity ranges from a minimum of 8K words to a maximum of 64K words, in 8K word increments.

When the memory system is 64K words, the memory is "wraparound" or circular, where the next location after 64K-1 is location 0. If a system has less than 64K words, any attempt to address a nonexistent location for either a fetch or store operation results in a machine fault interrupt.

The main memory may be accessed via one of four (maximum) access paths to the unit memory bus and by the CPU. Each access path of the unit memory bus is used by an IOP or a Direct Memory Adapter. Memory is addressed identically by all units (including the CPU) and only one memory access may take place during any instant of time. If two or more accesses to memory are attempted simultaneously, the conflict is resolved in accordance with the following priority: Direct Memory Adapter-2 (highest priority), IOP-2, IOP-1, Direct Memory Adapter-1, CPU (lowest priority).

CENTRAL PROCESSING UNIT

The central processing unit (CPU) is the primary controlling element for most system functions. Control intercommunications between the CPU, the interrupt system, external DIO adapter, IOPs, and Direct Memory Adapters are accomplished via the internal DIO bus.

Basically, the CPU consists of registers, an arithmetic logic unit, and a microprogrammed control unit (see Figure 2).

GENERAL REGISTERS

These eight registers are used for various purposes by a program. The address, designation, and function of each general register are as follows:

Address	Designation	Function
0	Z	Zero-Source
1	P	Program Address
2	L	Link Address
3	T	Temporary storage
4	X	Index 1 (post-index)
5	B	Index 2 (pre-index or base)
6	E	Extended accumulator
7	A	Accumulator

The general registers are addressable by General Register instructions, COPY instructions, READ DIRECT (Mode 0) instructions, and WRITE DIRECT (Mode 0) instructions, as described in Chapter 3.

PROTECTION SYSTEM REGISTERS

These 16 registers and a protect violation interrupt level comprise a protection system that guarantees the integrity of a master- or executive-mode (foreground) program while another (background) program is concurrently being executed. The protection system provides both operation protection and memory write protection. Each bit in these 16 registers (or words) is associated with a specific block of 256 consecutive locations in main memory. Bit 0 of protection register 0 is associated with main memory locations X'0000' through X'00FF', bit 1 of protection register 0 is

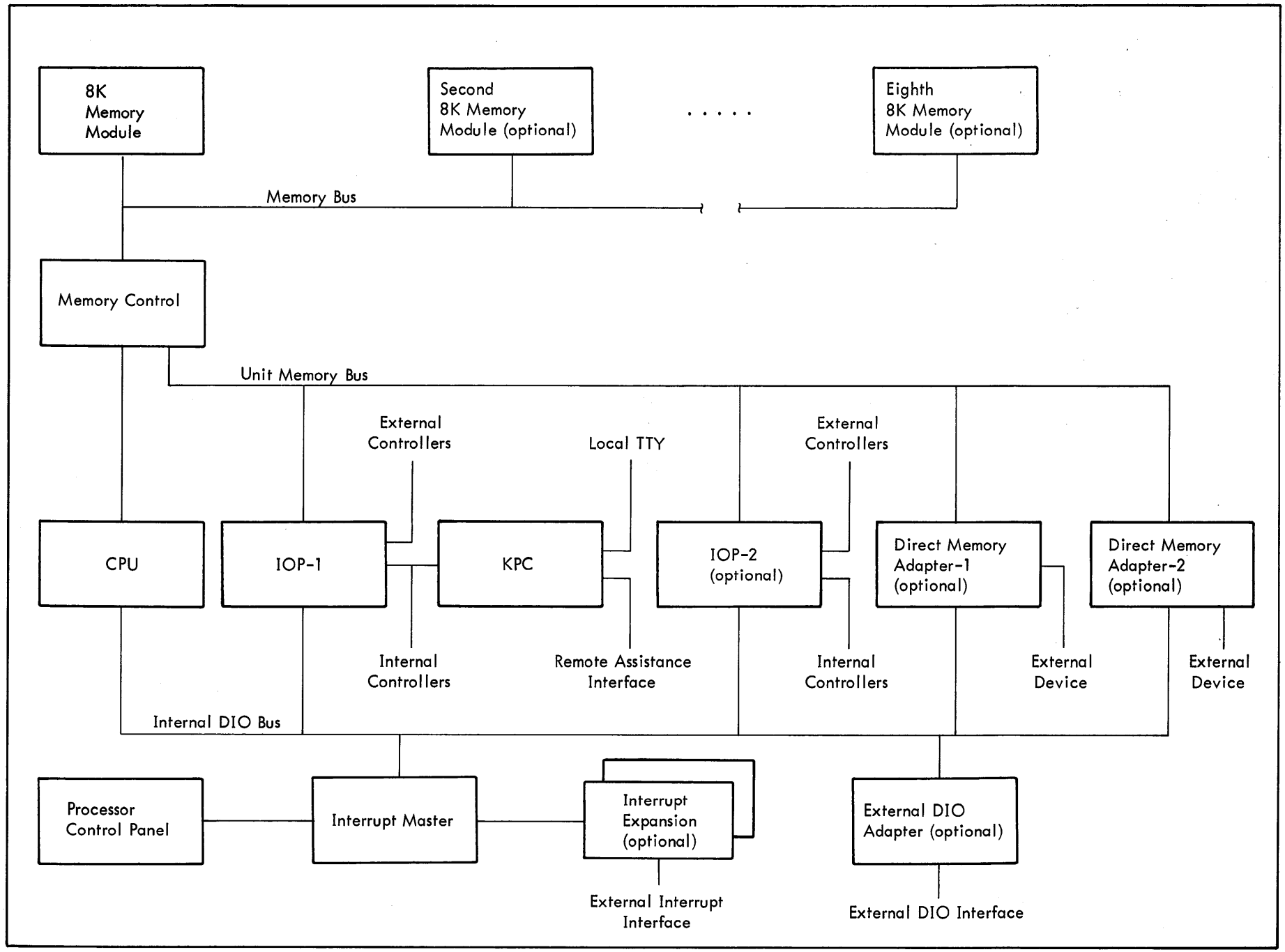


Figure 1. Xerox 530 Central System Block Diagram

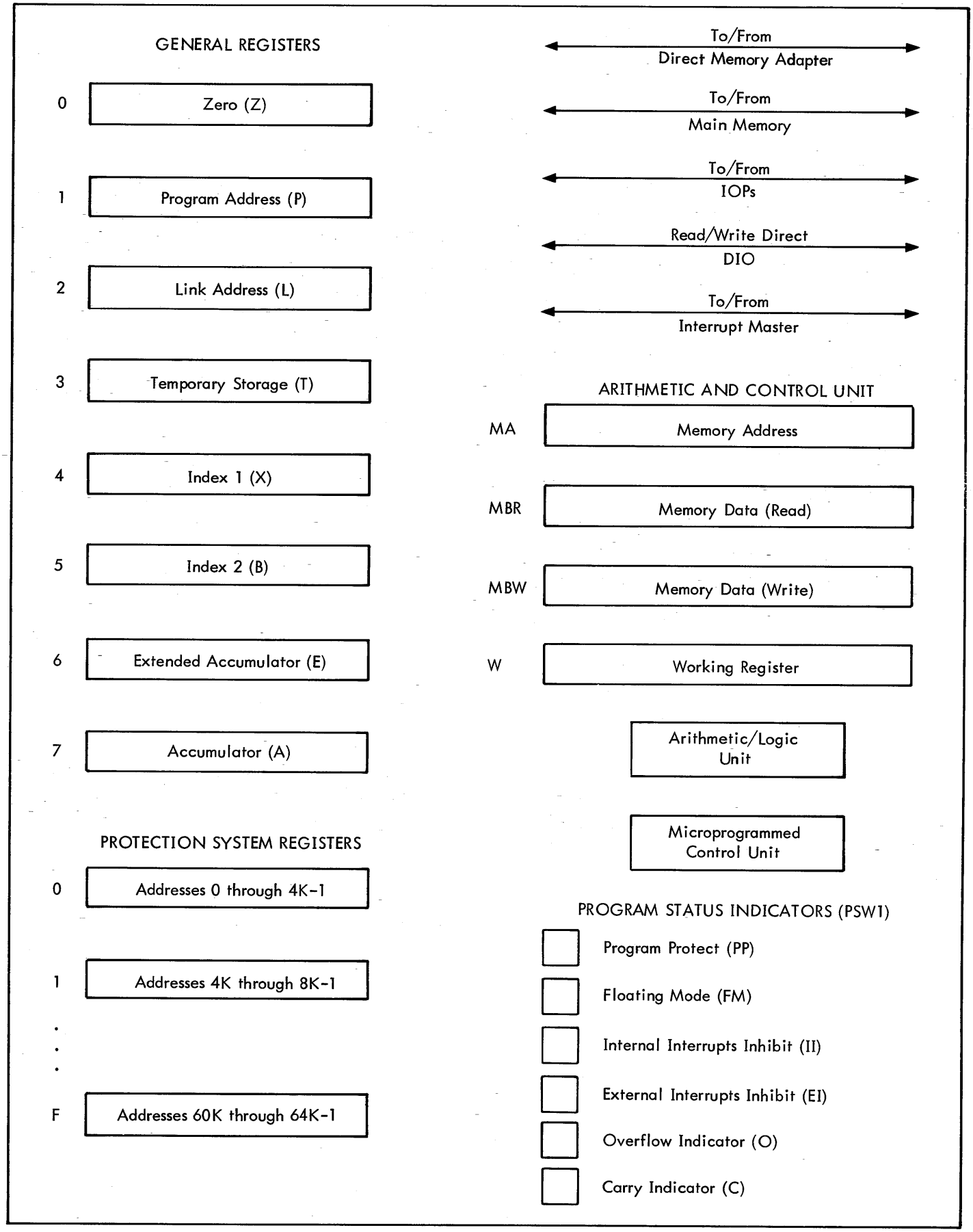


Figure 2. Central Processing Unit

associated with main memory locations X'0100' through X'01FF', and bit 15 of protection register X'F' is associated with main memory locations X'FF00' through X'FFFF'. A protect bit of "0" designates an unprotected memory block and a protect bit of "1" designates a protected block.

Each of the protection system registers can be loaded individually by executing a WRITE DIRECT (Mode 0) instruction having a function value of X'8r', where r is a hexadecimal digit that designates the protection register that is to be loaded with the contents of the accumulator (A register). Thus, the protect bits for 16 memory blocks (4096 words of main memory) can be set up by executing a single instruction.

Operation of the protection system is under control of the key-operated switch on the processor control panel (see Chapter 5). If the protection system is enabled, the following rules apply:

1. Privileged READ DIRECT and WRITE DIRECT instructions can be executed only if they are accessed from protected memory. If a privileged instruction is accessed from unprotected memory, the instruction is not executed; instead, the protect violation interrupt level is triggered. Note that two-word instruction sequences for Field Addressing, General Register, and Multiple Register instructions, as well as the SET FLOATING MODE instruction, are not privileged.

2. An instruction accessed from unprotected memory can be immediately followed by an instruction accessed from protected memory only in response to an interrupt condition. If an instruction is accessed from protected memory and the immediately preceding instruction was accessed from unprotected memory, the instruction is not executed (unless it is in response to an interrupt condition); instead, the protect violation interrupt level is triggered. This rule applies to branching from unprotected memory to protected memory as well as to executing an instruction in protected memory as the next instruction in normal sequence after an instruction in unprotected memory.

3. A STORE ACCUMULATOR (STA) or an INCREMENT MEMORY (IM) instruction can be used to alter protected memory only if the instruction is accessed from protected memory. If an attempt is made to alter protected memory with an instruction accessed from unprotected memory, the operation is not performed; instead, the protect violation interrupt level is triggered.

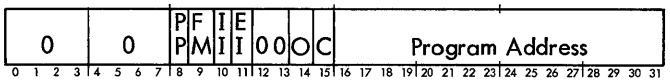
ARITHMETIC AND CONTROL UNIT

The arithmetic and control unit contains the necessary registers and control logic to access general registers or main memory, to modify instruction addresses, to perform arithmetic and logical operations, to provide indications of computational results, and to preserve interrupt status information. Basically, the arithmetic and control unit consists of registers, program status indicators, and a bussed arithmetic/logic unit with control provided by a microprogrammed control unit built around a read-only memory.

The MA (Memory Address), MBR (Memory Buffer Read), and MBW (Memory Buffer Write) registers are used to access main memory and provide temporary storage for information read out or written into main memory. The W (Working) register is an internal register and is not programmable; however, its contents may be displayed by the indicators on the processor control panel. The contents of the W register are used when generating effective addresses or effective instructions.

PROGRAM STATUS DOUBLEWORD

Critical control conditions of the CPU are defined by the Program Status Doubleword (PSD). When stored in main memory, the first (most significant) word of a PSD may occupy a location with an even address or an odd address and the second word must occupy the next contiguous location. The format of a PSD, as stored in memory, is as follows:



The first word of a PSD contains the following six status bits:

Bit Position	Status Bit Designation	Function
8	PP	Protected Program
9	FM	Floating Mode
10	II	Internal Interrupt Inhibit
11	EI	External Interrupt Inhibit
14	O	Overflow
15	C	Carry

Other bit positions are unassigned and must contain a zero. The second word of a PSD contains a program address. The first word of the current PSD is contained within an internal register (PSW); the second word of the current PSD is contained within general register 1 (P). The contents of the first word of the current PSD may be displayed by indicators on the Processor Control Panel.

If the Protected Program (PP) bit is a 1, the current program is located in an area of main memory that is protected; otherwise, the PP bit is a 0.

If the Floating Mode (FM) bit is a 1, the CPU is conditioned to perform Floating Point instructions (optional).

The Internal and External Interrupt inhibit bits determine whether a program interruption can occur. If an interrupt inhibit bit is a 0, the respective set of interrupt levels are allowed to interrupt the program being executed. Conversely; if an interrupt inhibit bit is a 1, the respective set of interrupt levels are inhibited from interrupting the program. Inhibiting interrupt levels also removes them from the interrupt system priority chain, allowing a lower-priority interrupt level to interrupt the program. Note, however, that the first six standard interrupt levels cannot be inhibited; and, that the six integral (external) interrupt levels within the standard group may be controlled with the Internal Interrupt inhibit bit.

The Overflow and Carry bits reflect the results of various operations. For arithmetic operations, the Overflow bit is set to a 1 if an overflow occurred and the Carry bit is set to a 1 if a carry occurred from the most significant (sign) position of the adder. Also, on a subtract operation, the Carry bit is set to a 1 if a "borrow" occurred from the sign position of the adder. For nonarithmetic operations (i.e., I/O operations), the Overflow and Carry bits may reflect status information relevant to the operation. When applicable, the significance of the Overflow and Carry bits are described under each instruction.

When an interrupt occurs, the current PSD is automatically stored in main memory and another PSD is loaded into the PSW and P registers to become the active PSD. The first PSD remains inactive in main memory until it is restored into the PSW and P registers (normally, when exiting from an interrupt-servicing routine).

INTERRUPT SYSTEM

Physically, the modular interrupt system is composed of one standard group of 16 interrupt levels and one or two optional groups, each with 12 external interrupt levels (see Table 1). Thus, a minimum interrupt system has 16 interrupt levels, an intermediate system has 28 levels, and a maximum system has 40 interrupt levels.

Each interrupt level has a unique priority, as listed in column 2, Table 1; a unique memory location, as listed in column 1; and, an assignment, as listed in column 6. Other operational and control characteristics are described in subsequent paragraphs.

STANDARD GROUP

The 16 interrupt levels that comprise the standard group are divided into two functional groups: internal and integral (or external). The functional assignments of each interrupt level is described below.

Power-On/Power-Off. These two interrupt levels (memory locations X'100' and X'101') are essential to the power monitor feature. Whenever an imminent power failure is sensed, the power-off interrupt level is triggered and a "power-off" routine is entered that typically stores (saves) volatile information (e.g., registers, program status doublewords, etc.) in main memory, halts all I/O operations, and ends with the CPU in a waiting state. When the power returns to a safe limit, the power-on interrupt level is triggered and a "power-on" routine is entered that typically restores information from main memory and prepares the system to resume processing.

These two interrupt levels operate automatically and continuously. They cannot be inhibited by the Internal Interrupt (II) bit of the Program Status Doubleword. Also, they are not addressable; hence, they cannot be controlled by a WRITE DIRECT instruction or interrogated with a READ DIRECT instruction.

If the power-on and power-off routines are both executed within two milliseconds, the power monitor feature is able to preserve the system, even under the most adverse condition (i.e., a power failure occurs immediately after the power assumed a normal value). This time constraint is imposed because of the following reasons:

1. The relative priority of the two interrupt levels precludes the power-off routine from becoming active until the power-on routine is completed (system is restored to a predictable state).

2. The primary power will remain at a safe limit for two milliseconds after an imminent power failure has been detected.

Note that when the power-on interrupt is active, the program protect feature is disabled (status of PSD 8, the PP bit, is ignored); and the program may execute instructions from any portion of memory.

Counter 2/Counter 1 (Real-Time Clocks). These two standard interrupt levels may be triggered by pulses from internal or external sources. Counter 1 has a constant frequency of 500 Hz; counter 2 may be set, at installation time, to any of four frequencies – the commerical line frequency, 2kHz, 8kHz, or a user-supplied external signal. When a clock pulse is received by one of the counter interrupt levels (and the level is armed and enabled), the value in the dedicated interrupt location is incremented by 1, and the level is cleared and re-armed. If the value in the dedicated interrupt location is zero after being incremented, the corresponding counter-equals-zero interrupt level is then triggered. All other interrupt levels (including the counter-equals-zero interrupt levels) are processed by interrupt-servicing routines and are designated as "normal" interrupt levels. The counter interrupt levels are addressable (group code X'0') and their status can be read with a READ DIRECT (Mode 1) instruction. They can be armed, disarmed, enabled, disabled, or triggered. Since a counter interrupt level never goes to the active state in normal operations,

Table 1. Interrupt System

Dedicated Interrupt Location		Priority Level	Read Status Register Bit	Set Active Register Bit	Write Direct Register Bit	Assignment	Availability	PSD Inhibit Bit	Read/Write Group Code
Dec.	Hex.								
256	100	1	None	None[t]	None[t]	Power on	Standard	None	None
257	101	2	None	None[t]	None[t]	Power off	Standard	None	None
254	FE	3	2	None	2	Counter 2	Standard	None	X'0'
255	FF	4	3	None	3	Counter 1	Standard	None	X'0'
258	102	5	0	None[t]	None[t]	Machine Fault	Standard	None	X'0'
259	103	6	1	None[t]	None[t]	Protection Violation	Standard	None	X'0'
264	108	7	8	8	8	Integral 5[tt]	Standard	II	X'0'
265	109	8	9	9	9	Integral 6[tt]	Standard	II	X'0'
262	106	9	6	6	6	Input/Output	Standard	II	X'0'
263	107	10	7	7	7	Control Panel	Standard	II	X'0'
266	10A	11	10	10	10	Counter 2 = 0	Standard	II	X'0'
267	10B	12	11	11	11	Counter 1 = 0	Standard	II	X'0'
268	10C	13	12	12	12	Integral 1[tt]	Standard	II	X'0'
269	10D	14	13	13	13	Integral 2[tt]	Standard	II	X'0'
270	10E	15	14	14	14	Integral 3[tt]	Standard	II	X'0'
271	10F	16	15	15	15	Integral 4[tt]	Standard	II	X'0'
272	110	17	0	0	0			EI	X'5'
273	111	18	1	1	1			EI	X'5'
274	112	19	2	2	2			EI	X'5'
275	113	20	3	3	3			EI	X'5'
276	114	21	4	4	4		First optional	EI	X'5'
277	115	22	5	5	5	Designated by	group of	EI	X'5'
278	116	23	6	6	6	Customer.	12 external	EI	X'5'
279	117	24	7	7	7		interrupts.	EI	X'5'
280	118	25	8	8	8			EI	X'5'
281	119	26	9	9	9			EI	X'5'
282	11A	27	10	10	10			EI	X'5'
283	11B	28	11	11	11			EI	X'5'
284	11C	29	12	12	12			EI	X'5'
285	11D	30	13	13	13			EI	X'5'
286	11E	31	14	14	14			EI	X'5'
287	11F	32	15	15	15			EI	X'5'
288	120	33	0	0	0		Second	EI	X'6'
289	121	34	1	1	1	Designated by	optional	EI	X'6'
290	122	35	2	2	2	Customer.	group of	EI	X'6'
291	123	36	3	3	3		12 external	EI	X'6'
292	124	37	4	4	4		interrupts.	EI	X'6'
293	125	38	5	5	5			EI	X'6'
294	126	39	6	6	6			EI	X'6'
295	127	40	7	7	7			EI	X'6'

[t]These bits need not be set to zero by program (they are ignored by hardware).

[tt]Connected and used as an external interrupt level.

the level must not be programmed into that state. These two levels can not be inhibited by the II bit of the Program Status Doubleword. Counter 2 has the third highest priority and counter 1 has the fourth highest priority within the interrupt system.

Machine Fault. This interrupt level is triggered whenever the Fault Register is nonzero, signifying that an abnormal condition has been detected and the correct PCP switches are enabled (see "Fault System"). The status of the machine fault interrupt level may be read into bit position 0 of the

A register by executing a READ DIRECT (Mode 1) instruction. This interrupt level cannot be controlled with a WRITE DIRECT (Mode 1) instruction nor can it be inhibited by the Internal Interrupt bit of the Program Status Doubleword.

Protection Violation. This sixth highest priority interrupt level is part of the protection system. This interrupt level is triggered if the protection system is enabled when a protection violation is encountered. The status of the interrupt level is read into bit position 1 of the A register whenever a READ DIRECT (Mode 1) instruction is executed. This interrupt level cannot be controlled with a WRITE DIRECT (Mode 1) instruction nor can it be inhibited by the Internal Interrupt control bit of the Program Status Doubleword.

Input/Output. The input/output interrupt level accepts interrupt signals from standard I/O systems. An I/O interrupt-servicing routine must contain an ACKNOWLEDGE I/O INTERRUPT (AIO) instruction, described in Chapter 4, that identifies the source and cause of an I/O interrupt. The I/O interrupt level is addressed with a group code of X'0'. The state of the interrupt level may be read into bit 6 of the A register by executing a READ DIRECT (Mode 1) instruction. The interrupt level may be controlled by a WRITE DIRECT (Mode 1) instruction and bit 6 of the A register. The interrupt level may be inhibited by the Internal Interrupt bit of the Program Status Doubleword.

Control Panel. The control panel interrupt level may be activated by either the INTERRUPT switch on the processor control panel or by a special control sequence on the operator's keyboard (see Chapter 5). The interrupt level can be triggered by the computer operator to initiate a specific routine. The control panel interrupt level is addressed with a group code of X'0'. The state of this interrupt level may be read into bit position 7 of the A register by executing a READ DIRECT (Mode 1) instruction. The interrupt level may be controlled with a WRITE DIRECT (Mode 1) instruction and bit 7 of the A register. The interrupt level may be inhibited by the Internal Interrupt bit of the Program Status Doubleword.

Counter-Equals-Zero. As described under "Counter 2/ Counter 1" above, a counter-equals-zero interrupt level is triggered whenever the corresponding counter (memory location) is incremented to the value of zero as a result of a count pulse. Triggering a counter-equals-zero interrupt level does not affect the counting process. The counter-equals-zero interrupt levels may be addressed with a group code of X'0'. The state of counter two-equals-zero may be read into bit position 10 of the A register and the state of counter one-equals-zero may be read into bit position 11 of the A register when a READ DIRECT (Mode 1) instruction is executed. Both interrupt levels may be controlled with a WRITE DIRECT (Mode 1) instruction and both interrupt levels may be inhibited by the Internal Interrupt bit of the Program Status Doubleword.

Integral Interrupt Levels. There are six integral interrupt levels within the standard group that may be used by the customer as external interrupts. The interrupt levels are addressed with a group code of X'0'. The state of each interrupt level may be read with a READ DIRECT instruction and controlled with a WRITE DIRECT instruction. The interrupt levels may be inhibited by the Internal Interrupt bit of the Program Status Doubleword.

Note: Care must be exercised in assigning integral five and integral six interrupt levels since foreground tasks connected to these interrupts can not utilize certain RBM or BCM monitor services (e.g., requests for monitor input/output services can not be made from these interrupt levels).

OPTIONAL INTERRUPT GROUPS

Each optional interrupt group consists of 12 external interrupt levels. All 24 optional interrupt levels may be inhibited by the External Interrupt bit of the Program Status Doubleword. The first 16 optional interrupt levels are addressed with a group code of X'5' while the last eight optional interrupt levels are addressed with a group code of X'6'. All optional interrupt levels may be read with a READ DIRECT (Mode 1) instruction and controlled with a WRITE DIRECT (Mode 1) instruction.

INTERRUPT LEVEL STATES

Each interrupt level is controlled on an individual basis by three flip-flops. Two of the flip-flops define four mutually exclusive states: disarmed, armed, waiting, and active. The third flip-flop enables or disables the level. The various states and the condition causing changes in state (see Figure 3) are described in the following paragraphs:

Disarmed. When an interrupt level is in the disarmed state, no signal to that interrupt level is admitted; that is, no record is retained of the signal nor is any program interrupt caused by it at any time.

Armed. When an interrupt level is in the armed state (IP flip-flop is on), it is capable of accepting and remembering an interrupt signal. The receipt of such a signal advances the interrupt level to the waiting state (IS flip-flop is turned on).

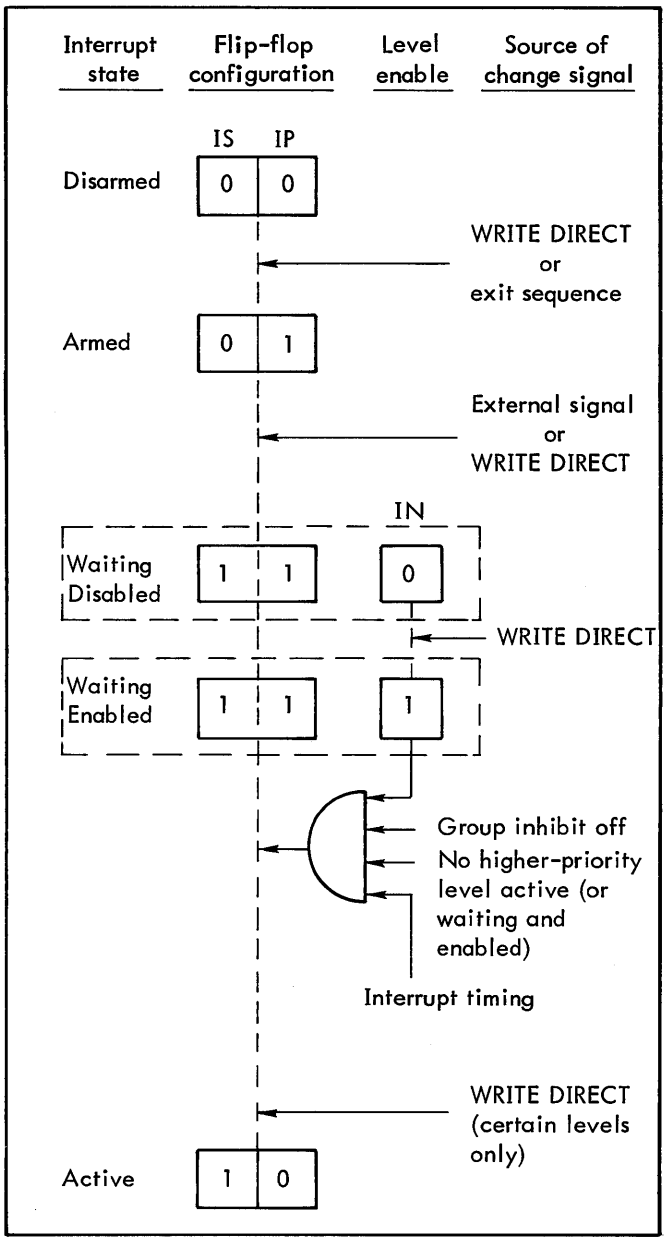


Figure 3. Interrupt Level Operation

Waiting. When an interrupt level in the armed state receives an interrupt signal, it advances to the waiting state, and remains in the waiting state until it is allowed to advance to the active state. (IP and IS flip-flops are both on in the waiting state.)

If the level-enable flip-flop (IN) is off, the interrupt level can undergo all state changes except that of moving from the waiting to the active state. Furthermore, if this flip-flop is off, the interrupt level is completely removed from the chain that determines the priority of access to the CPU. Thus, an interrupt level in the waiting state with its level-enable in the off condition does not prevent an enabled, uninhibited interrupt level of lower priority from moving to the active state.

When an interrupt level is in the waiting state, the following conditions must all exist simultaneously before the level advances to the active state:

1. The level is enabled (i.e., its level enable flip-flop, IN, is a 1).

2. The group inhibit (if applicable) is off (i.e., the appropriate inhibit is a 0).

3. No higher-priority interrupt level is in the active state (or is in the enabled, waiting state with its inhibit off).

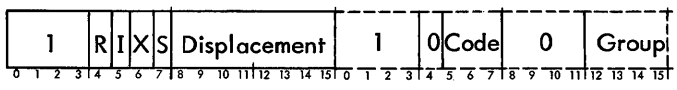
4. The CPU is in an interruptible phase of operation.

Active. When a normal interrupt level (any interrupt level except Counter 2 or Counter 1) meets all of the conditions necessary to permit it to move from the waiting state to the active state, it is permitted to do so by being acknowledged by the computer, which then automatically stores the current PSD at the location specified by the contents of the location associated with the level. The first instruction of the interrupt-servicing routine is then taken from the location following the stored PSD. A new interrupt cannot occur until after the execution of this first instruction.

A normal interrupt level remains in the active state until it is removed from the active state by a specific configuration of the WRITE DIRECT (WD) instruction, followed by an LDX instruction (an "exit sequence"), or if the same active level is armed or disarmed via an appropriate WD instruction. An interrupt-servicing routine can itself be interrupted (whenever a higher-priority interrupt level meets all of the conditions for becoming active) and then continued (after the higher-priority interrupt level is removed from the active state). However, an interrupt-servicing routine cannot be interrupted by a lower-priority interrupt level as long as its interrupt level remains in the active state. Normally, the interrupt-servicing routine returns its interrupt level to the armed state and transfers program control back to the point of interrupt by means of an interrupt routine exit sequence (see "Interrupt Routine Entry and Exit").

READING INTERRUPT LEVELS

Except for the first two interrupt levels (power on and power off), the state of all interrupt levels may be read into the A register by executing a READ DIRECT (Mode 1) instruction. The format of a typical READ DIRECT (Mode 1) instruction is as follows:



The effective instruction (shown enclosed within broken lines) is generated from the original instruction in the same manner as an effective address; i.e., the displacement value is modified as specified by the "RIXS" bits.

The recommended method for producing the appropriate configuration of the effective instruction is to indirectly address a memory location that contains the appropriate bit configuration.

Bits 0-3 of the effective instruction must be coded as X'1' to specify the interrupt control mode. (See READ DIRECT instruction, Chapter 3, for other control modes.)

Bits 4 and 8-11 must be coded as zeros.

Bits 5, 6, and 7 (the "code" field) may be coded to one of three values to specify a read operation to be performed (read one of the three flip-flops associated with each interrupt level).

Code Field Bits 5 6 7	Read Function
001	Set to a 1 the accumulator bits corresponding to each interrupt level that is in the Armed or Waiting state. Read IP flip-flops.)
010	Set to a 1 the accumulator bits corresponding to each interrupt level that is in the Waiting (or Active) state. (Read IS flip-flops.)
100	Set to a 1 the accumulator bits corresponding to each interrupt level that is Enabled. (Read IN flip-flops.)

The A register bit format for the READ DIRECT (Mode 1) instruction is shown in column 3 of Table 1. For READ DIRECT group code X'0', bit 5 is unused and is indeterminate, and bit 4 is set to 1 if the TRACE toggle switch is in the up (on) position or set to 0 if the TRACE switch is in the down (off) position. An uninstalled interrupt level will respond with a 0 to any of the three read operations.

Bits 12-15 specify which group of interrupt levels is to be interrogated.

Group Field Value	Interrupt Group Interrogated
X'0'	Standard group. Note that the first two interrupt levels (power on and power off) cannot be interrogated with a READ DIRECT instruction.
X'5'	The first 16 (optional) external interrupt levels.
X'6'	The last 8 (optional) external interrupt levels.

INTERRUPT SYSTEM CONTROL

The interrupt system may be controlled in the following ways:

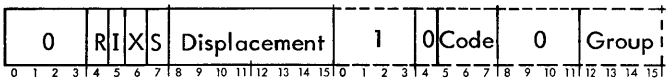
1. If the Internal Interrupt (II) bit of the Program Status Doubleword (PSD) is set to a 1, all interrupt levels within the standard group, except the first six – power on, power off, counter 2, counter 1, machine fault, and protection violation (see Table 1), are inhibited.

2. If the External Interrupt (EI) bit of the PSD is set to a 1, all optional interrupt levels are inhibited.

3. Except for the first six interrupt levels, each interrupt level (on an individual basis as specified by assigned bits of the A register) may be set to the Active state by executing a WRITE DIRECT (Mode 1) instruction, described below.

4. Except for the power on, power off, machine fault, and protection violation interrupt levels, each interrupt level (on an individual basis as specified by assigned bits of the A register) may be armed, disarmed, enabled, disabled, or triggered by executing a WRITE DIRECT (Mode 1) instruction, described below.

The format of a typical WRITE DIRECT (Mode 1) instruction for controlling the interrupt system is as follows:



The effective instruction (enclosed within broken lines) is generated in the same manner as an effective address; i.e., the displacement value of the original instruction is modified by the "RIXS" bits.

The recommended method for producing the appropriate configuration of the WRITE DIRECT effective instruction is to indirectly address a memory location that contains the appropriate bit configuration.

Bits 0-3 of the effective instruction must be coded as X'1' to specify the interrupt control mode. (See "WRITE DIRECT" instruction, Chapter 3, for other control modes.)

Bits 4 and 8-11 must be coded as zeros.

Bits 5, 6, and 7 (the "code" field) may be coded to any one of eight values to specify a function that is to be performed on a group of interrupt levels which is designated by the value of the "group" field (bits 12-15).

Individual interrupt levels within the designated group are further designated by the contents of the A register.

The value of the "code" field and associated functions are as follows:

Code Field Bits 5 6 7	Write Function
000	A code field (bits 5-7) of 000 will cause each interrupt level corresponding to the 1's in the accumulator to be set to the Active state if that interrupt level was previously in either the Armed or Waiting state. This operation is not affected by the state of the level enable flip-flops or the group inhibits. Any levels in the Disarmed state and those levels corresponding to the 0's in the accumulator are not affected. If the selected interrupt level is already Active, it will be set to the Disarmed state. The Set Active operation causes the selected level to enter the Active state, without going through the automatic interrupt entry sequence.
001	Disarm all levels corresponding to a 1 in the accumulator; no other levels are affected.
010	Arm and enable all levels corresponding to a 1 in the accumulator; no other levels are affected.
011	Arm and disable all levels corresponding to a 1 in the accumulator; no other levels are affected.
100	Enable all levels corresponding to a 1 in the accumulator; no other levels are affected.
101	Disable all levels corresponding to a 1 in the accumulator; no other levels are affected.
110	Enable all levels corresponding to a 1 in the accumulator and disable all other levels.
111	Trigger all levels corresponding to a 1 in the accumulator. All such levels that are currently armed advance to the waiting state. Those levels currently disarmed are not altered, and all levels corresponding to a 0 in the accumulator are not affected. The interrupt trigger is applied at the same input point as that used by the device connected to the interrupt level.

The required values for the "group" field and the designated interrupt groups are as follows:

Group Value	Selected Interrupt Group
X'0'	Standard group. Note, as shown in Table 1, that the power on, power off, machine fault, and protection violation interrupt levels cannot be controlled by a WRITE DIRECT instruction.
X'5'	The first 16 (optional) external interrupt levels.
X'6'	The last 8 (optional) external interrupt levels.

The relationship between the individual bits of the A register and the individual interrupt levels for the WRITE DIRECT instruction is shown in columns 4 and 5 of Table 1. Note that all bits of the A register are not assigned within groups X'0' and X'6'.

INTERRUPT ROUTINE ENTRY AND EXIT

When a normal interrupt level (not counter 2 or counter 1) becomes active, the computer automatically saves the Program Status Doubleword (which contains the protected program indicator, the floating mode bit, internal and external inhibit bits, overflow and carry bits, and the program address). The status information is stored in the location whose address is contained in the dedicated interrupt location. If the FM bit is set, the Scratchpad Floating Accumulator is stored into the Memory Floating Accumulator (three locations identified by the address in memory location 1).

The current value in the program address (P) register is stored in the location following the status information. The significance of the stored program address depends upon the particular interrupt level as follows:

1. For the machine fault error or the protect violation, the stored program address is the address of the instruction that was being executed at the time the interrupt condition occurred.

2. For all other normal interrupt levels, the stored program address is the address of the next instruction in sequence after the instruction whose execution was just completed at the time the interrupt condition occurred.

After the program address is stored, the next instruction to be executed is then taken from the location following the stored program address. The first instruction of the interrupt-servicing routine is always executed before another interrupt

can occur. Thus, the interrupt-servicing routine may inhibit all other normal interrupt levels so that the routine itself will not be interrupted while in process.

At the end of an interrupt-servicing routine, an exit sequence restores the program status and Scratchpad Floating Accumulator that existed when the interrupt level became active. An exit sequence is a two-word instruction sequence comprised of a WRITE DIRECT (Mode 0) instruction with an effective address of X'00D8' followed immediately by a LOAD INDEX (LDX) instruction with an effective address equal to the address value in the interrupt location for the routine (no interrupt is processed by the CPU between these two instructions). Execution of LDX in an interrupt routine exit sequence does not affect the contents of index register 1 (X).

COUNTER INTERRUPT PROCESSING

The counter interrupt levels are not associated with interrupt-servicing as such. Instead, an active counter interrupt level is serviced by accessing the contents of the memory location assigned to the interrupt level, incrementing the value in the memory location by 1, and restoring the new value in the same memory location. The processing of an active counter interrupt level does not affect the overflow indicator or the carry indicator. Thus, the on-going program is not affected by a counter clock pulse (other than by the time required for processing) unless the result in the assigned memory location is all 0's after being incremented; in that case, the corresponding counter-equals-zero interrupt level is triggered.

CPU INTERRUPT RECOGNITION

If all other conditions are met and an interrupt level is waiting and enabled, the CPU will recognize and process an interrupt following the completion of any instruction, except between the storing of the PSD and the execution of the next instruction upon entering a normal interrupt subroutine.

Note: Two-word instruction sequences, as required for field addressing, multiple register operations, and general register operations, as well as for exiting from an interrupt-servicing routine, are not completed until the second word of the sequence is executed. Hence, the CPU does not process any interrupts during a two-word instruction sequence.

FAULT SYSTEM

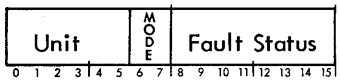
The fault system continuously monitors operations, especially data transfers, within the entire computer system in order to detect abnormal conditions.

By executing a READ DIRECT (Mode 1) instruction with an effective address of X'1040' (normally, as part of an MFI subroutine), 16 bits of fault status information (as described below) are copied into the A register and the fault register is reset. By evaluating the fault information, the CPU may perform either a fault logging operation or a fault recovery procedure.

The relatively high priority of the MFI level within the interrupt system permits a fault condition to be reported and processed expeditiously. Note that the MFI level may be interrogated with a READ DIRECT (Mode 1) instruction but cannot be controlled with a WRITE DIRECT instruction or inhibited by the Internal Interrupt inhibit bit of the Program Status Doubleword.

FAULT INFORMATION FORMATS

The general format for the 16 bits of fault information which may be copied into the A register or displayed on the processor control panel is as follows:



The "unit" field identifies the source(s) of fault information (see Table 2); the "mode" field permits the various faults which may be detected within each unit to be classified into one of four modes; and the "fault status" field indicates specific faults. In case a memory fault is detected, the fault status field also includes the three most significant bits of memory address associated with the attempted memory access. The mode and fault status information is dependent upon the unit (see Tables 3, 4, and 5).

Table 2. Unit Field (Fault Information)

Bit Position 0 1 2 3 4 5	Source of Fault Information
0 0 0 0 0 0	No faults.
1 - - - - -	CPU.
0 1 - - - -	Reserved for special systems.
0 - 1 - - -	IOP-2.
0 - - 1 - -	IOP-1.
0 - - - 1 -	Interrupt Master or External DIO system.
0 - - - - 1	Direct Memory Adapter.

Notes: 1. If the unit field indicates a CPU fault (bit 0 is a 1) in combination with one

Table 2. Unit Field (Fault Information) (cont.)

Notes: (cont.)	or more other unit indicators (bits 1-5), then the mode and fault status information (bits 6-15) is valid only for the CPU. The READ DIRECT instruction (effective address X'1040') that returned this word of fault information will reset the CPU unit fault indicator (bit 0) to 0, reset the CPU mode and fault status information (bits 6-15), and allow the mode and fault status information from the other units to occupy the fault register. A second READ DIRECT instruction should be issued within the Machine Fault Interrupt (MFI) subroutine in order to avoid a second MFI, and in order to process the remaining fault information. (See Note 2.)
2.	If the unit field indicates a combination of two or more unit indicators (bits 1-5) and the CPU indicator is zero (bit 0 is a 0), then the unit field is valid in pointing to the units that reported faults simultaneously, but the mode and fault status information (bits 6-15) is not valid (the information is merged). The READ DIRECT instruction which returned this word of fault information will reset the entire fault register.

Table 3. Mode and Fault Status Values (CPU Faults)

Bit Position 6 7 8 9 10 11 12 13 14 15	Type of Fault
	Mode 1
0 1 1 - - - - - - -	Instruction access fault.
0 1 - X - - - - - -	Most significant bit (MSB) of memory address.
0 1 - - X - - - - -	Second MSB of memory address.
0 1 - - - X - - - -	Third MSB of memory address.
0 1 - - - - 1 - - -	Reserved.
0 1 - - - - - 1 - -	Memory module absent (nonexistent address).

Table 3. Mode and Fault Status Values (CPU Faults) (cont.)

Bit Position 6 7 8 9 10 11 12 13 14 15	Type of Fault
0 1 - - - - - - 1 -	Memory address parity error.
0 1 - - - - - - - 1	Memory read data parity error.
	Mode 3
1 1 1 - - - - - - -	Instruction timer error.
1 1 - 1 - - - - - -	Address parity error in ROS.
1 1 - - 1 - - - - -	Data parity error in ROS.
1 1 - - - 1 - - - -	Memory control error.
1 1 - - - - 1 - - -	Parity error in arithmetic unit.
1 1 - - - - - 1 - -	Reserved.
1 1 - - - - - - 1 -	Reserved.
1 1 - - - - - - - 1	Reserved.
	Mode 2
1 0 1 - 0 - 0 0 0 0	Reserved.
1 0 - 1 0 - 0 0 0 0	Fan not normal.
1 0 - - 0 1 0 0 0 0	DIO data-in parity error.
1 0 0 0 1 0 0 0 0 0	Control module error.
1 0 0 0 - 0 1 - - -	No DFSA response.
1 0 0 0 - 0 - 1 - -	DFSA remains high constantly.
1 0 0 0 - 0 - - 1 -	Unimplemented instruction (class A).
1 0 0 0 - 0 - - - 1	Unimplemented instruction (class B).
	Mode 0
0 0 - - - - - - - -	Reserved.

Notes: 1. If more than one mode of faults exists simultaneously, the CPU Fault Register

Table 3. Mode and Fault Status Values (CPU Faults) (cont.)

Notes: (cont.)	will report only the highest priority fault mode. The fault modes have the following priority: Mode 1 (highest), Mode 3, Mode 2, Mode 0. In addition, Mode 2 faults are divided into two groups which are mutually exclusive for reporting purposes. The first group consists of two faults: fan not normal and DIO data-in parity error. The second group of Mode 2 faults consists of five faults that may be detected within the microprogrammed operations. The first group of Mode 2 faults have priority over the second group.
2.	Within a particular mode, more than one fault may be reported simultaneously.

Table 4. Mode and Fault Status Values (IOP Faults)

Bit Positions 6 7 8 9 10 11 12 13 14 15	Definition
	Memory Faults:
0 1 1 - - - - - - -	Reserved.
0 1 - X - - - - - -	First bit of memory address.
0 1 - - X - - - - -	Second bit of memory address.
0 1 - - - X - - - -	Third bit of memory address.
0 1 - - - - 1 - - -	Reserved.
0 1 - - - - - 1 - -	Memory address not here.
0 1 - - - - - - 1 -	Memory address parity error.
0 1 - - - - - - - 1	Memory data parity error.
	I/O Faults:
0 0 1 - - - - - - -	Reserved.
0 0 - 1 - - - - - -	IOP Adapter fault.
0 0 - - 1 - - - - -	IOP Fault.
0 0 - - - 1 - - - -	Watchdog timer interface.

Table 4. Mode and Fault Status Values (IOP Faults) (cont.)

Bit Positions 6 7 8 9 10 11 12 13 14 15	Definition
	I/O Faults (cont.):
0 0 - - - - 1 - - -	DIO address fault.
0 0 - - - - - 1 - -	DIO data parity error.
0 0 - - - - - - 1 -	NIO address parity error.
0 0 - - - - - - - 1	NIO data parity error (not reported for data-in parity error).

Table 5. Mode and Fault Status Values (IM and DIO Faults)

Bit Positions 6 7 8 9 10 11 12 13 14 15	Description
0 0 0 0 0 0 0 1 0 0	A pseudo-fault generated whenever the TRACE switch is on and the INTERRUPT switch is activated.
0 0 0 0 0 0 0 0 1 0	A DIO parity error occurred during a WRITE DIRECT instruction to the Interrupt Master. No attempt is made to abort the WRITE DIRECT.
0 0 0 0 0 0 0 0 0 1	Data parity error occurred during an external WRITE DIRECT instruction. No attempt is made to abort the WRITE DIRECT.
Note: Faults that are detected in the Interrupt Master (IM) or in DIO operations are as listed above. The first two faults are associated with IM operations and the last fault is associated with DIO operations.	

EFFECTIVE ADDRESS COMPUTATION

The CPU forms the effective address of a memory reference instruction in three basic steps as follows:

1. If the R bit (bit 4 of the instruction word) and the S bit (bit 7 of the instruction word) are both 0's, the reference address is equal to the value in the displacement field of the instruction. (This is referred to as "non-relative" addressing.)

2. If the R bit is a 0 and the S bit is a 1, the reference address is equal to the value in the displacement field

in the instruction plus the 16-bit value (base address) in index register 2. (This is referred to as "pre-indexing", or "base-relative" addressing.)

3. If the R bit is a 1, the reference address is equal to the 16-bit value in the internal W register (address of the current instruction before it is incremented to the address of the next instruction) plus the value in the low-order nine bits of the instruction, interpreted as a 9-bit two's complement integer. (This is referred to as "self-relative" addressing.)

Step 2 (determine direct address)

1. If the I bit (bit 5 of the instruction word) is 0, the direct address is equal to the value of the reference address (as determined in step 1).

2. If the I bit is a 1, the reference address is treated as an indirect address; the direct address is the 16-bit value in the location whose address is equal to the reference address. In effect, the indirect address is replaced by the direct address value.

Step 3 (determine effective address)

1. If the X bit (bit 6 of the instruction word) is a 0, the effective address is equal to the value of the direct address (as determined by step 2).

2. If the X bit is a 1, the effective address is equal to the value of the direct address plus the 16-bit value in index register 1. Note that indexing with X is applied after indirect addressing. (This is referred to as "post-indexing".)

The effective address for an instruction, therefore, is the final 16-bit address value developed for that instruction, starting with the displacement value in the instruction itself. The main memory location whose address equals the effective address value is referred to as the "effective location". Similarly, the contents of the effective location are referred to as the "effective word".

The process of effective address computation is summarized in Table 6. The symbols used in Table 6 are defined as follows:

R Bit 4 of the instruction.

I Bit 5 of the instruction.

X Bit 6 of the instruction.

S Bit 7 of the instruction.

D Bits 8-15 of the instruction (Displacement).

SD Sign extended displacement value.

(D) Contents of location D.

(X) Contents of index register 1 (general register 4).

(B) Contents of index register 2 (general register 5).

(W) Contents of the internal W register (the address of the current instruction).

Table 6. Effective Address Computation

R	I	X	S	Effective Address
0	0	0	0	D
0	0	0	1	D + (B)
0	0	1	0	D + (X)
0	0	1	1	D + (X) + (B)
0	1	0	0	(D)
0	1	0	1	(D + (B))
0	1	1	0	(D) + (X)
0	1	1	1	(D + (B)) + (X)
1	0	0		(W) + SD
1	0	1		(W) + SD + (X)
1	1	0		((W) + SD)
1	1	1		((W) + SD) + (X)

EFFECTIVE INSTRUCTIONS

An effective instruction is generated in the same manner as an effective address. In the case of effective instructions, the final 16-bit value (or portions thereof) is used to modify or augment the operation code of a given instruction, namely, a READ DIRECT or a WRITE DIRECT instruction that is used separately or as part of a two-word instruction sequence, as described in Chapter 3.

3. INSTRUCTION REPERTOIRE

This chapter describes all CPU instructions, except I/O instructions, which are described in Chapter 4. When applicable, the following information is provided for each instruction:

1. Mnemonic – the code used by assemblers to produce the instruction's basic operation code.

2. Instruction name – the instruction's descriptive title.

3. Parenthetical note – an indication of whether the instruction is optional and/or privileged.

4. Format – a one- or two-word diagram showing the various subfields of the instruction. The contents of subfields may be represented with descriptive words, symbols, abbreviations, or specific hexadecimal or binary values.

   When the configuration of an effective instruction can be ascertained readily from the configuration of the original instruction (i.e., when the "RIXS" bits of the original instruction are coded with zeros, the last eight bits of the effective instruction are equal to the last eight bits of the original instruction, and the leading eight bits of the effective instruction are zeros), the diagram (enclosed with solid border lines) illustrates the format of the instruction as stored in memory or as printed out during a listing.

   When the configuration of an effective instruction does not correlate directly with that stored in memory, that portion of the effective instruction which is generated by modifying the contents of the displacement field is enclosed within broken border lines and appears immediately to the right of the original instruction. These portions of the effective instruction are available to the computer via internal registers and are not stored in memory or available for printouts.

   Two-word instructions that specify Field Addressing, General Register, and Multiple Register operations must occupy two contiguous memory locations. The diagrams for these instructions show the first instruction word directly above the second instruction word. The address of the first word may be either even or odd.

5. Verbal description – an explanation of the function or operation performed by the instruction.

6. Affected – a symbolic listing of registers, storage areas, and indicators that can be affected by the instruction.

   Note: The instruction address portion of the program status doubleword (P register) is considered affected only if a branch condition can occur as a result of the instruction execution.

7. Timing – see Appendix B.

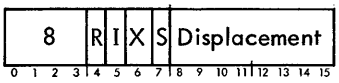
The following symbols are commonly used in the descriptions:

A	Accumulator (general register 7).
B	Index 2 (general register 5).
C	Carry indicator (PSD bit 15).
D	Displacement (bits 8-15 of instruction).
E	Extended accumulator (general register 6).
EI	External interrupt inhibit (PSD bit 11).
EL	Effective location.
EW	Effective word, or (EL).
FM	Floating Mode bit (PSD bit 9).
(W)	Contents of the internal W register (address of the current instruction).
II	Internal interrupt inhibit (PSD bit 10).
O	Overflow indicator (PSD bit 14).
P	Program address register, next instruction address, (general register 1).
PP	Protected program indicator (PSD bit 8).
SD	Sign extended displacement value.
T	Temporary storage (general register 3).
X	Index 1 (general register 4).
Z	General Register 0.

Other symbols and abbreviations, normally applicable to a limited number of instructions, are defined within the descriptions of those instructions.

MEMORY REFERENCE INSTRUCTIONS

LDA LOAD REGISTER A

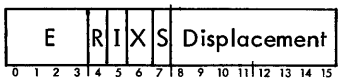


LOAD REGISTER A is a one-word nonfloating-point instruction that loads the effective word into the A register (general register 7).

This format is also used by FLOATING LOAD (see "Floating-Point Instructions").

Affected: (A)

STA STORE REGISTER A

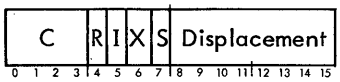


STORE REGISTER A is a one-word nonfloating-point instruction that stores the contents of the A register (general register 7) into the effective location.

This format is also used by FLOATING STORE (see "Floating-Point Instructions").

Affected: (EL)

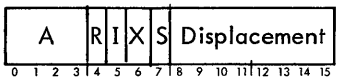
LDX LOAD INDEX



LOAD INDEX is a one-word instruction that loads the effective word into Index 1 (general register 4). This instruction is not affected by the Floating Mode bit (PSD 9).

Affected: (X)

ADD ADD



ADD is a one-word nonfloating-point instruction that adds the effective word to the contents of the A register and then loads the result into the A register.

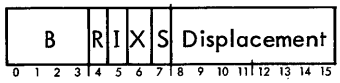
The Overflow and Carry indicators are set or reset to reflect the result of the addition as follows:

O	C	Significance
1	-	The signs of the two operands are equal but the sign of the result is different.
-	1	A carry occurred from the sign bit position of the adder.
0	0	Neither of the above conditions occurred.

This format is also used by FLOATING ADD (see "Floating-Point Instructions").

Affected: (A), O, C

SUB SUBTRACT



SUBTRACT is a one-word nonfloating-poing instruction that forms the one's complement of the effective word, increments by 1, adds this value to the contents of the A register, and then loads the result into the A register.

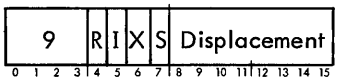
The Overflow and Carry indicators are set or reset to reflect the result of the subtraction as follows:

O	C	Significance
1	-	The sign of the result in the A register is equal equal to the sign of the effective word, but the sign of the original operand in the A register was different.
-	1	A carry occurred from the sign bit position of the adder, either during incrementing the one's complement or in adding the value to the A register: the magnitude of the 16-bit word in the effective location is equal to or less than the magnitude of the 16-bit word in the A register.
0	0	Neither of the above conditions occurred.

This format is also used by FLOATING SUBTRACT (see "Floating-Point Instructions").

Affected: (A), O, C

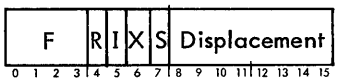
AND LOGICAL AND



LOGICAL AND is a one-word instruction (not affected by the FM bit) that forms the logical product of the effective word and the contents of the A register, and loads this product into the A register. The logical product contains a 1 in each bit position for which there is a corresponding 1 in both the A register and the effective word; the logical product contains a 0 in each bit position for which there is a 0 in the corresponding bit position of either operand.

Affected: (A)

IM INCREMENT MEMORY



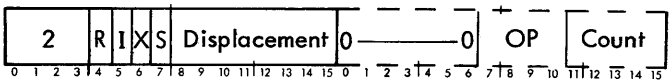
INCREMENT MEMORY is a one-word instruction (not affected by the FM bit) that adds 1 to the effective word and then stores the result in the effective location.

The Overflow and Carry indicators are set or reset to reflect the result of the incrementing as follows:

O	C	Significance
1	-	The resulting value of the effective word is X'8000' (32,768).
-	1	The resulting value of the effective word is X'0000'.
0	0	Neither of the above conditions occurred.

Affected: (EL), O, C

S SHIFT (General)



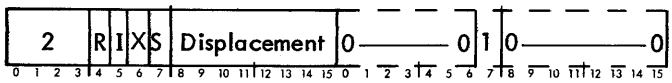
This general format may be used to specify any of nine one-word SHIFT instructions. The displacement value is modified as specified by the "RIXS" bits to generate the effective instruction. SHIFT instructions are not affected by the Floating Mode bit.

Bits 0 through 6 of the effective instruction must be coded as zeros. Bits 7 through 10 specify one of the following nine shift instructions which are described separately:

7 8 9 10	Shift instruction specified	Mnemonic
1 0 0 0	Normalize shift	
0 0 0 0	Shift Arithmetic Right Single	SARS
0 0 0 1	Shift Arithmetic Left Single	SALS
0 0 1 0	Shift Circular Right Single	SCRS
0 0 1 1	Shift Circular Left Single	SCLS
0 1 0 0	Shift Arithmetic Right Double	SARD
0 1 0 1	Shift Arithmetic Left Double	SALD
0 1 1 0	Shift Circular Right Double	SCRD
0 1 1 1	Shift Circular Left Double	SCLD

Bits 11 through 15 are used only by the nonnormalized shift instructions (when bit 7 is a 0). These bits contain the "count", a value of 0 through 31, which specifies the number of bit positions of the shift operation.

NORMALIZE SHIFT



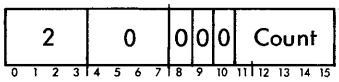
The NORMALIZE SHIFT instruction must be generated by coding the "RIXS" bits of the original shift instruction with a nonzero value so that bit 7 of the effective instruction is a 1. All other bits of the effective instruction must be 0.

If the initial contents of the Extended Accumulator (E) and the Accumulator (A) are both zero, the instruction exits without changing any register.

If the initial contents of either E or A are not zero, the instruction performs a double-register arithmetic left shift on E and A (bits shifted out of bit position 0 of A shift into bit position 15 of E). The double-register shifting continues until bit positions 0 and 1 of E are different. The contents of the temporary storage register, T (general register 3), are decremented by one for each left shift performed. At the completion of the NORMALIZE SHIFT instruction, the Carry indicator is reset and the Overflow indicator is set if the contents of the T register overflowed (i.e., was decremented past negative full scale during the normalize operation). If the T register has not overflowed, the Overflow indicator is reset.

Affected: (T), (E), (A), O, C

SARS SHIFT ARITHMETIC RIGHT SINGLE

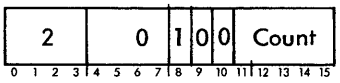


A SHIFT ARITHMETIC RIGHT SINGLE instruction causes the contents of the A register to be shifted right as many bit positions as specified by the "count" field. The sign position (bit 0) is copied into vacated bit positions on the left. Bits shifted out of bit position 15 are lost. Both Overflow and Carry indicators are reset to zero.

Note: Although the effective instruction may be generated by using other values in bit positions 4 through 15 of the original instruction, the recommended coding is as illustrated above.

Affected: (A), O, C

SARD SHIFT ARITHMETIC RIGHT DOUBLE



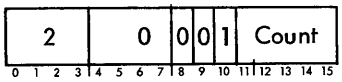
A SHIFT ARITHMETIC RIGHT DOUBLE instruction causes the contents of the E and A registers to be shifted right as many bit positions as specified by the "count" field. The two registers are treated as a single 32-bit register. The sign position (bit 0) of the E register is copied into vacated bit

positions on the left of the E register. Bits shifted out of bit position 15 of the E register are copied into bit position 0 of the A register. Bits shifted out of bit position 15 of the A register are lost. Both Overflow and Carry indicators are reset to zero.

Note: Although the effective instruction may be generated by using other values in bit positions 4 through 15 of the original instruction, the recommended coding is as illustrated above.

Affected: (E), (A), O, C

SALS SHIFT ARITHMETIC LEFT SINGLE



A SHIFT ARITHMETIC LEFT SINGLE instruction causes the contents of the A register to be shifted left as many bit positions as specified by the "count" field. Zeros are copied into the vacated bit positions on the right. Bits shifted out of the sign position (bit 0) are lost.

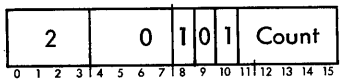
The Overflow and Carry indicators are set or reset to reflect the result of the shift as follows:

O	C	Significance
1	-	The sign bit was changed.
-	1	An odd number of "1" bits were shifted out of the sign bit position.
0	0	Neither of the above conditions occurred.

Note: Although the effective instruction may be generated by using other values in bit positions 4 through 15 of the original instruction, the recommended coding is as illustrated above.

Affected: (A), O, C

SALD SHIFT ARITHMETIC LEFT DOUBLE



A SHIFT ARITHMETIC LEFT DOUBLE instruction causes the contents of the E and A registers to be shifted left as many bit positions as specified by the "count" field. The two registers are treated as a single 32-bit register. Vacated bit positions on the right of A register are filled with zeros. Bits shifted out of bit position 0 of the A register are copied into bit position 15 of the E register. Bits shifted out of bit position 0 of the E register are lost.

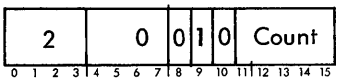
The Overflow and Carry indicators are set or reset to reflect the result of the shift as follows:

O	C	Significance
1	-	The sign bit was changed.
-	1	An odd number of "1" bits were shifted out of the sign bit position.
0	0	Neither of the above conditions occurred.

Note: Although the effective instruction may be generated by using other values in bit positions 4 through 15 of the original instruction, the recommended coding is as illustrated above.

Affected: (E), (A), O, C

SCRS SHIFT CIRCULAR RIGHT SINGLE

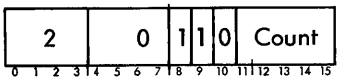


A SHIFT CIRCULAR RIGHT SINGLE instruction causes the contents of the A register to be shifted right as many bit positions as specified by the "count" field. Bits shifted out of bit position 15 are copied into bit position 0. Both Overflow and Carry indicators are reset to zero.

Note: Although the effective instruction may be generated by using other values in bit positions 4 through 15 of the original instruction, the recommended coding is as illustrated above.

Affected: (A), O, C

SCRD SHIFT CIRCULAR RIGHT DOUBLE

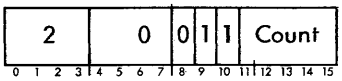


A SHIFT CIRCULAR RIGHT DOUBLE instruction causes the contents of the E and A registers to be shifted right as many bit positions as specified by the "count" field. The two registers are treated as a single 32-bit register. Bits are shifted out of bit position 15 of the E register into bit position 0 of the A register and from bit position 15 of the A register into bit position 0 of the E register. Both Overflow and Carry indicators are reset to zero.

Note: Although the effective instruction may be generated by using other values in bit positions 4 though 15 of the original instruction, the recommended coding is as illustrated above.

Affected: (E), (A), O, C,

SCLS SHIFT CIRCULAR LEFT SINGLE



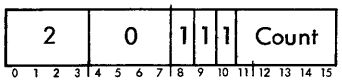
A SHIFT CIRCULAR LEFT SINGLE instruction causes the contents of the A register to be shifted left as many bit positions as specified by the "count" field. Bits shifted out of the sign position (bit 0) are copied into bit position 15.

O	C	Significance
0	0	An even number of "1" bits were shifted out of the sign bit position.
0	1	An odd number of "1" bits were shifted out of the sign bit position.

Note: Although the effective instruction may be generated by using other values in bit positions 4 through 15 of the original instruction, the recommended coding is as illustrated above.

Affected: (A), O, C

SCLD SHIFT CIRCULAR LEFT DOUBLE



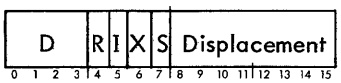
A SHIFT CIRCULAR LEFT DOUBLE instruction causes the contents of the E and A registers to be shifted left as many bit positions as specified by the "count" field. The two registers are treated as a single 32-bit register. Bits are shifted from bit position 0 of the A register into bit position 15 of the E register and from bit position 0 of the E register into bit position 15 of the A register.

O	C	Significance
0	0	An even number of "1" bits were shifted out of the sign bit position.
0	1	An odd number of "1" bits were shifted out of the sign bit position.

Note: Although the effective instruction may be generated by using other values in bit positions 4 through 15 of the original instruction, the recommended coding is as illustrated above.

Affected: (E), (A), O, C

CP COMPARE



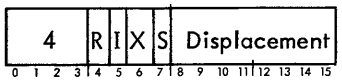
COMPARE is a one-word nonfloating-point instruction that algebraically compares the contents of the A register and the effective word, with both operands treated as signed quantities. The Overflow and Carry indicators are set or reset to reflect the result of the comparison as follows:

O	C	Result of Comparison
0	0	The operand in the A register is algebraically less than the effective word.
1	0	The operand in the A register is algebraically greater than the effective word.
1	1	The operand in the A register is equal to the effective word.

This format is also used by FLOATING COMPARE (see "Floating-Point Instructions").

Affected: O, C

B BRANCH



BRANCH (a one-word instruction, not affected by FM bit) loads the effective address into the Program Address register (general register 1). Thus, unconditionally, the next instruction is accessed from the location pointed to by the effective address of the BRANCH instruction. (Conditional branch instructions are described below.)

This instruction also resets the Floating Mode (FM) bit (PSD 9) to a zero.

Affected: (P), FM

CONDITIONAL BRANCH INSTRUCTIONS

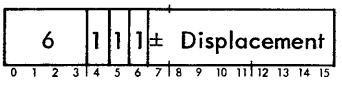
The eight conditional branch instructions specify conditional, relative branching. Each conditional branch instruction performs a test to determine whether the branch condition is "true".

If the branch condition is true, the instruction acts as a BRANCH instruction coded for self-relative addressing with neither indirect addressing nor indexing. (The conditional branch instructions automatically invoke self-relative addressing.) Thus, if the branch condition is true, the next instruction is accessed from the location pointed to by the effective address of the conditional branch instruction.

If the branch condition is not true, the instruction acts as a "no operation" instruction and the next instruction is accessed from the next location in ascending sequence after the conditional branch instruction.

Each conditional branch instruction is a one-word instruction not affected by the FM bit.

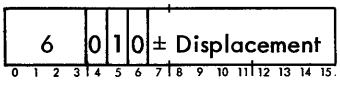
BAN BRANCH IF ACCUMULATOR NEGATIVE



The branch condition is true only if bit 0 of the accumulator (general register 7) is 1.

Affected: (P)

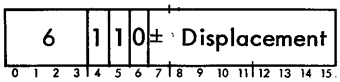
BAZ BRANCH IF ACCUMULATOR ZERO



The branch condition is true only if the accumulator (general register 7) contains the value X'0000'.

Affected: (P)

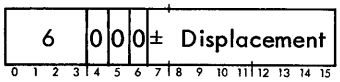
BEN BRANCH IF EXTENDED ACCUMULATOR NEGATIVE



The branch condition is true only if bit 0 of the extended accumulator (general register 6) is 1.

Affected: (P)

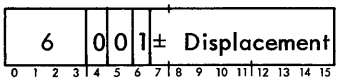
BNO BRANCH IF NO OVERFLOW



The branch condition is true only if the Overflow indicator is reset (0). The Overflow indicator is not affected.

Affected: (P)

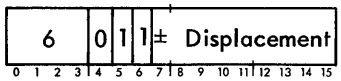
BNC BRANCH IF NO CARRY



The branch condition is true only if the Carry indicator is reset (0). The Carry indicator is not affected.

Affected: (P)

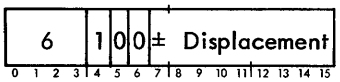
BIX BRANCH ON INCREMENTING INDEX



BIX adds 1 to the current value in Index 1 (general register 4) and loads the result into Index 1. The branch condition is true only if the result in Index 1 is a nonzero value.

Affected: (X), (P)

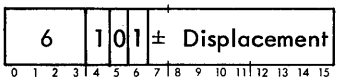
BXNO BRANCH ON INCREMENTING INDEX AND NO OVERFLOW



If the Overflow indicator is set (1), no operation is performed and the computer executes the next instruction in sequence. However, if the Overflow indicator is reset (0), BXNO adds 1 to the current value in Index 1 (general register 4) and loads the result into Index 1; the branch condition is true only if the result in Index 1 is a nonzero value. The Overflow indicator is not affected by this instruction.

Affected: (X), (P)

BXNC BRANCH ON INCREMENTING INDEX AND NO CARRY



If the Carry indicator is set (1), no operation is performed and the computer executes the next instruction in sequence. However, if the Carry indicator is reset (0), BXNC adds 1 to the current value in Index 1 and loads the result into Index 1; the branch condition is true only if the result in Index 1 is a nonzero value. The Carry indicator is not affected.

Affected: (X), (P)

COPY INSTRUCTIONS

A one-word copy instruction specifies operations between any two general registers. The format of a copy instruction is

Bit(s)	Function
0-3	Bit positions 0-3 are coded as X'7', to specify the copy instruction.
4-5 (OP)	Bit positions 4-5 specify which of four operations is to be performed. The operations are

4	5	Operation	
0	0	Logical AND	Overflow and Carry indicators not affected.
0	1	Logical inclusive OR	
1	0	Logical exclusive OR	
1	1	Arithmetic add (Overflow and Carry indicators set as described for the instruction ADD.)	

Bit(s)	Function
6 (AC)	Bit position 6 specifies whether the current value of the Carry indicator is to be added to the result. If this bit is a 1, the Carry indicator is added to the low-order bit position of the result. If this bit is a 0, the Carry indicator is ignored.
7 (AI)	Bit position 7 specifies whether the value X'0001' is to be added to the result. If this bit is a 1, a 1 is added to the low-order bit position of the result. If bits 6 and 7 are both 1's, the value X'0001' is added to the result (regardless of the current value of the Carry indicator).
8 (CD)	Bit position 8 specifies whether the destination register (specified by bits 9-11) is to be cleared before the operation called for by bits 4-7 is performed. If bit 8 is a 1, the destination register is initially cleared. If bit 8 is 0, the initial contents of the destination register remain unchanged until the result is loaded into the destination register.
9-11 (DR)	Bit positions 9-11 specify the general register that is to contain the result of the instruction. The Overflow and Carry indicators may be affected.
12 (IS)	Bit position 12 specifies whether the source register operand (the value in the register specified by bits 13-15) is to be used as it appears in the source register, or is to be inverted (one's complemented) before the operation is performed. If bit 12 is a 1, the inverse of the value in the source register is to be used as the source register operand; however, the value in the source register is not changed. If bit 12 is a 0, the value in the source register is used as the source register operand.
13-15 (SR)	Bit positions 13-15 specify the general register that contains the value to be used (normally or inverted) as the source register operand. A value of 0 in this field designates the value X'0000' as the contents of the source register.

The general registers are identified as follows:

Address	Designation	Function
0	Z	Zero
1	P	Program address
2	L	Link address
3	T	Temporary storage
4	X	Index 1
5	B	Index 2 (base address)
6	E	Extended accumulator
7	A	Accumulator

When execution of the copy instruction begins, the P register has already been incremented and contains the address of the next instruction.

Copy instructions are not affected by the Floating Mode bit.

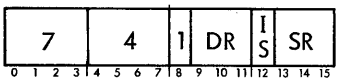
Affected: (DR), O, C

Examples:

Instruction	Effect
X'74F0'	Clear the accumulator (general register 7) to all zeros.
X'74FF'	Invert (form the one's complement of) the contents of the accumulator.
X'7DFF'	Negate (form the two's complement of) the contents of the accumulator.
X'7C78'	Subtract 1 from the contents of the accumulator.
X'7D7B'	Subtract the contents of the T register from the contents of the accumulator.
X'75A1'	Copy the contents of the P register plus 1 into the L register.

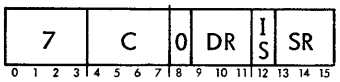
The basic assembly language recognizes the following command mnemonics and generates the appropriate settings for bit positions 0-8 of the copy instruction. The settings for bit positions 9-15 are derived from the argument field of the symbolic line in which the command mnemonic appears. The source register operand is the contents of the source register if the IS bit is 0, or is the inverse (one's complement) of the contents of the source register if the IS bit is 1.

RCPY REGISTER COPY



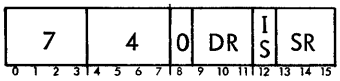
RCPY copies the source register operand into the destination register. The Overflow and Carry indicators are not affected.

RADD REGISTER ADD



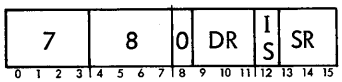
RADD adds the source register operand to the contents of the destination register and loads the result into the destination register. The Overflow and Carry indicators are set as described for the instruction ADD, based on the register operands and the final result.

ROR REGISTER OR



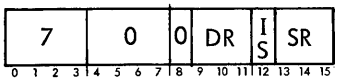
ROR logically inclusive ORs the source register operand with the contents of the destination register and loads the result into the destination register. If the corresponding bits in the source register operand and the destination register are both 0, a 0 remains in the corresponding bit position of the destination register; otherwise, the corresponding bit position of the destination register is set to 1. The Overflow and Carry indicators are not affected.

REOR REGISTER EXCLUSIVE OR



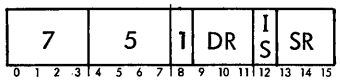
REOR logically exclusive ORs the source register operand with the contents of the destination register and loads the result into the destination register. If the corresponding bits of the source register operand and the destination register are different, the corresponding bit position of the destination register is set to 1; otherwise, the corresponding bit position of the destination register is reset to 0. The Overflow and Carry indicators are not affected.

RAND REGISTER AND



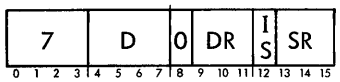
RAND logically ANDs the source register operand with the contents of the destination register and loads the result into the destination register. If the corresponding bits of the source register operand and the destination register are both 1, a 1 remains in the destination register; otherwise, the corresponding bit position of the destination register is reset to 0. The Overflow and Carry indicators are not affected.

RCPYI REGISTER COPY AND INCREMENT



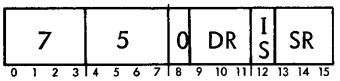
RCPYI copies the source register operand into the destination register and then adds 1 to the new contents of the destination register. The Overflow and Carry indicators are not affected.

RADDI REGISTER ADD AND INCREMENT



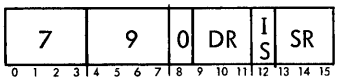
RADDI adds the source register operand to the contents of the destination register, increments the result by 1, and loads the final result into the destination register. The Overflow and Carry indicators are set, as described for the instruction ADD, based on the register operands and the final result.

RORI REGISTER OR AND INCREMENT



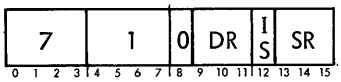
RORI logically ORs the source register operand with the contents of the destination register, increments the result by 1, and loads the final result into the destination register. The Overflow and Carry indicators are not affected.

REORI REGISTER EXCLUSIVE OR AND INCREMENT



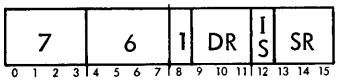
REORI logically exclusive ORs the source register operand with the contents of the destination register, increments the result by 1, and loads the final result into the destination register. The Overflow and Carry indicators are not affected.

RANDI REGISTER AND AND INCREMENT



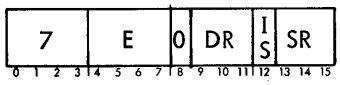
RANDI logically ANDs the source register operand with the contents of the destination register, increments the result by 1, and loads the final result into the destination register. The Overflow and Carry indicators are not affected.

RCPYC REGISTER COPY AND CARRY



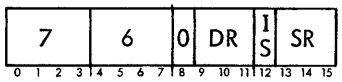
RCPYC copies the source register operand into the destination register and then adds the current value of the Carry indicator to the result in the destination register. The Overflow and Carry indicators are not affected.

RADDC REGISTER ADD AND CARRY



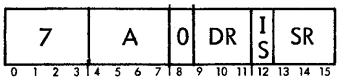
RADDC adds the source register operand to the contents of the destination register, adds the current value of the Carry indicator to the result and loads the final result into the destination register. The Overflow and Carry indicators are set, as described for the instruction ADD, based on the register operands and the final result.

RORC REGISTER OR AND CARRY



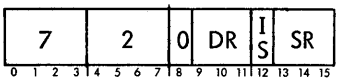
RORC logically inclusive ORs the source register operand with the contents of the destination register, adds the current value of the Carry indicator to the result, and loads the final result into the destination register. The Overflow and Carry indicators are not affected.

REORC REGISTER EXCLUSIVE OR AND CARRY



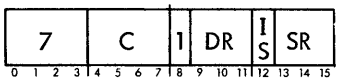
REORC logically exclusive ORs the source register operand with the contents of the destination register, adds the current value of the Carry indicator to the result, and loads the final result into the destination register. The Overflow and Carry indicators are not affected.

RANDC REGISTER AND AND CARRY



RANDC logically ANDs the source register operand with the contents of the destination register, adds the current value of the Carry indicator to the result and loads the final result into the destination register. The Overflow and Carry indicators are not affected.

RCLA REGISTER CLEAR AND ADD

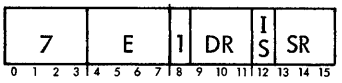


RCLA clears the destination register, adds the source register operand to the cleared destination register, and loads the final result into the destination register. The Overflow and Carry indicators are reset to 0.

RCLAI REGISTER CLEAR, ADD, AND INCREMENT

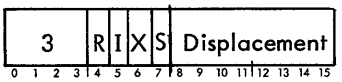
RCLAI clears the destination register, adds the source register operand to the cleared destination register, increments the result by 1, and loads the final result into the destination register. The Overflow and Carry indicators are set, as described for the instruction ADD, based on the contents of the source register operand and the final result.

RCLAC REGISTER CLEAR, ADD AND CARRY



RCLAC clears the destination register, adds the source register operand to the cleared destination register, adds the current value of the Carry indicator to the result, and loads the final result into the destination register. The Overflow and Carry indicators are set, as described for the instruction ADD, based on the contents of the source register operand and the final result.

MUL MULTIPLY

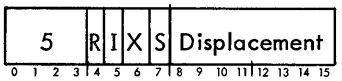


MULTIPLY is a one-word nonfloating-point instruction that multiplies the effective word by the contents of the A register (treating both words as signed integers), loads the 16 high-order bits of the doubleword product into the extended accumulator (general register 6), and loads the 16 low-order bits into the A register (general register 7). Neither overflow nor carry can occur; however, the Carry indicator is set equal to the sign of the doubleword product.

This format is also used by FLOATING MULTIPLY (see "Floating-Point Instructions").

Affected: (E), (A), C

DIV DIVIDE



DIVIDE is a one-word nonfloating-point instruction that divides the doubleword contained within the extended accumulator (general register 6) and the accumulator (general register 7) by the effective word (treating both words as a signed integer).

If the absolute value of the quotient is equal to or greater than 32,768 (2^{15}), the Overflow indicator is set to 1 and the instruction is terminated with the contents of the extended accumulator and the accumulator unchanged from their previous values, and the Carry indicator is set equal to the sign of the dividend.

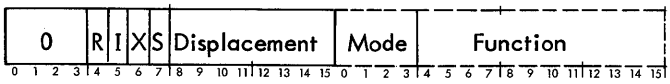
If the absolute value of the quotient is less than 32,768, the Overflow indicator is reset to 0, the integer quotient is loaded into the accumulator (general register 7), the integer remainder is loaded into the extended accumulator, and the Carry indicator is set equal to the sign of the remainder. (The sign of the remainder is the same as the sign of the dividend.)

This format is also used by FLOATING DIVIDE (see "Floating-Point Instructions").

Affected: (E), (A), O, C

WD WRITE DIRECT (privileged, partially optional)

The format of a typical WRITE DIRECT instruction is



The "mode" and "function" fields for the effective instruction are generated in the same manner as described under "Effective Address Computation" in Chapter 2. As listed in Table 7, each of the 16 different values of the mode field designates a different portion of the computer system that is to perform a control or nonarithmetic operation as specified by the value of the accompanying function field. The number of different operations that can be specified within each mode (or for each portion of the computer system) varies for each mode. The WRITE DIRECT instruction, as the READ DIRECT instruction, generates a set of control or nonarithmetic instructions that may be comprised of 16 subsets of instructions (one subset of instructions for each mode or for each portion of the computer system). The subset of instructions for WRITE DIRECT (Mode 0) operations is described within subsequent paragraphs. The subsets of instructions for other modes of operations (Mode 1 through Mode F) are appropriately referenced.

WRITE DIRECT (MODE 0) INSTRUCTIONS

The format and recommended coding for a typical WRITE DIRECT (Mode 0) instruction is as follows:

The WRITE DIRECT (Mode 0) instruction is unique in that if the "RIXS" bits are coded as zeros, the "function" may be encoded directly into the original instruction. Otherwise, the "RIXS" bits and the "displacement" field of the original instruction must be coded with appropriate values, which will result in an effective instruction as shown below.

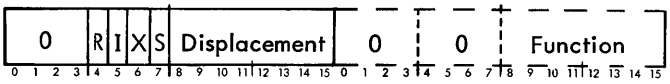


Table 7. WRITE DIRECT Mode Values and General Functions Performed

Mode		Portions of Computer System Designated	General Control or Nonarithmetic Operation Performed
Value	Title		
0	Internal Computer Control	CPU and/or IOP	1. Copy contents of the A register into a specified general register or I/O channel register. 2. Copy contents of bit position 0 of specified general register or I/O channel register into the Overflow indicator and then reset bit position 0 of the same register to 0. 3. Copy contents of the A register into one of the 16 protection registers. 4. Load program status information from the A register into the first word of the Program Status Doubleword. 5. Prepare IOP-1 or IOP-2 to operate as directed by a subsequent diagnostic program. 6. Set the Wait flip-flop to a 1 and cause the CPU to stop computations. 7. Prepare the CPU to exit from an interrupt-servicing routine. 8. Set or reset the External Interrupt (EI) and/or the Internal Interrupt (II) program status indicators.
1	Interrupt Control	Interrupt system	As described in Chapter 2, under "Interrupt System".
2 thru E	Direct Control	DIO system using standard Xerox computer products	Exchange control information and data as required to perform an I/O operation. Note that each mode value is assigned to a different DIO system. (Refer to the Xerox Computer Systems/Interface Design Manual, 90 09 73, for further details.)
F	Direct Control	Specially designed equipment	Exchange control information and data as required to perform an I/O operation. (Refer to the Xerox Computer Systems/Interface Design Manual, 90 09 73, for further details.)

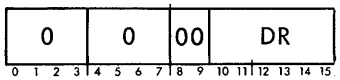
The function field may have any one of 256 different values; however, as described subsequently, not all values of the function field are assigned as part of an effective instruction (see Appendix C).

Note: Attempting to use a function value that is not assigned results in an "unimplemented instruction" fault.

For explanation purposes, the function values associated with WRITE DIRECT (Mode 0) instructions are divided into four functional groups, which are also readily differentiated by the values of the first two bits of the function field (bits 8 and 9), as described below.

Function field bit positions		Control or nonarithmetic operation performed
8	9	
0	0	All function values within this group (except X'00') are assigned. Each value designates a specific general register or an I/O channel register that is to be loaded with the contents of the A register.

The format and recommended coding for a WRITE DIRECT instruction to perform the above described operation is as follows:

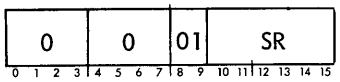


The "DR" field is coded with the address of the general or I/O channel register that is to be the destination register for the load operation.

Affected: (DR)

0	1	All function values within this group (except X'40') are assigned. Each value designates a specific general register or an I/O channel register whose bit position 0 content is to be copied into the Overflow indicator and then reset to a 0.

The format and recommended coding for a WRITE DIRECT instruction to perform the above described operation is as follows:

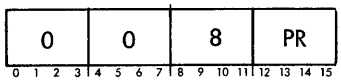


The "SR" field is coded with the address of the general or I/O channel register that is to be the source register from which bit position 0 is to be copied.

Affected: O and bit 0 of source register.

Function field bit positions		Control or nonarithmetic operation performed
8	9	
1	0	Within this group of function values (X'80'-X'BF') only the first 16 function values (X'80'-X'8F') are assigned. Each of the assigned values permits one of the 16 protection registers to be loaded with the contents of the A register.

The format and recommended coding for a WRITE DIRECT instruction to load a protection register is as follows:

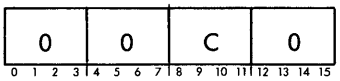


The protection register, as specified by the value of the "PR" field is loaded with the contents of the A register.

Affected: (protection register)

1	1	Within this group of function values (X'C0'-X'FF'), only the function values described below are assigned.

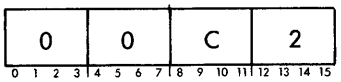
The format and recommended coding for a WRITE DIRECT instruction to load the program status indicators of the first word of the Program Status Doubleword from the A register is as follows:



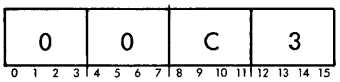
Bit positions 8-11, 14, and 15 of the PSD are loaded with the contents of corresponding bit positions of the A register. If this instruction causes the Protected Program bit (PSD 8) to change from a 1 to a 0, the next instruction is in protected memory, and the protection feature is operative. A protection violation interrupt occurs when the next instruction is accessed.

Affected: Program Status Indicators

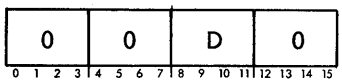
The format and recommended coding for a WRITE DIRECT instruction to prepare IOP-1 to operate as directed by a subsequent diagnostic program is as follows:



The format and recommended coding for a WRITE DIRECT instruction to prepare IOP-2 to operate as directed by a subsequent diagnostic program is as follows:



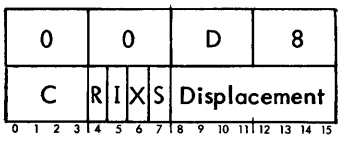
Function field bit positions		Control or nonarithmetic operation performed
8	9	
1	1	The format and recommended coding for a WRITE DIRECT instruction to set the Wait flip-flop and cause the CPU to stop communication is as follows:



The Wait flip-flop may be reset to a 0 by any interrupt activation (including counter interrupts) or by moving the COMPUTE switch to the IDLE position.

Affected: Wait flip-flop

A WRITE DIRECT instruction is also used as the first word of a two-word instruction sequence for exiting from an interrupt-servicing routine. The format and recommended coding for the two-word instruction sequence is as follows:

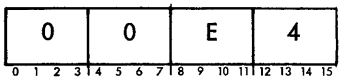


The first instruction word sets the exit condition by inhibiting all normal interrupt levels. The second instruction word, whose effective address must be equal to the address in the interrupt location for the desired interrupt-servicing routine, performs the following:

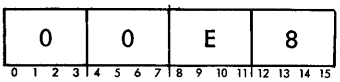
1. Loads the Program Status Doubleword from the first two words of the interrupt routine (bits 0-7, 12, and 13 of the effective doubleword are ignored).

2. If the FM bit of the new PSD is set, the SFA is loaded from the Memory Floating Accumulator.

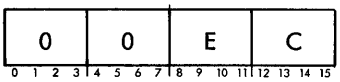
Function field bit positions		Control or nonarithmetic operation performed
8	9	
1 (cont.)	1	3. Arms the highest-priority active interrupt level. 4. Resets the exit condition. Affected: PP, FM, II, EI, O, C, (P), SFA, highest-priority interrupt level The format and recommended codings for setting and resetting the External and Internal interrupt control bits of the Program Status Doubleword and the significance of each are as follows:



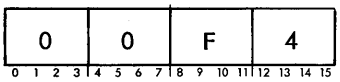
Reset the External Interrupt bit to 0.



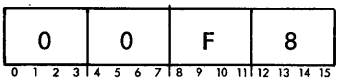
Reset the Internal Interrupt bit to 0.



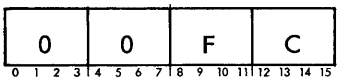
Reset both the External and Internal Interrupt bits to 0.



Set the External Interrupt bit to 1.



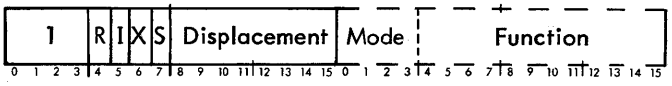
Set the Internal Interrupt bit to 1.



Set both the External and Internal Interrupt bits to 1.

RD READ DIRECT (partially privileged, partially optional)

The format of a typical READ DIRECT instruction is



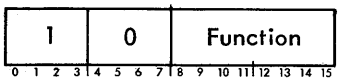
The "mode" and "function" fields for the effective instruction are generated in the same manner as described under "Effective Address Computation", in Chapter 2. As listed in Table 8, each of the 16 different values of the mode field designates a different portion of the computer system that is to perform a control or nonarithmetic operation as specified by the value of the accompanying function field. The number of different operations that can be specified within each mode (or for each portion of the computer system) varies for each mode. Thus, the READ DIRECT instruction generates a set of control or nonarithmetic instructions that may be comprised of 16 subsets of instructions (one subset of instructions for each mode or for each portion of the computer system). The subset of instructions for READ DIRECT (Mode 0) operations is described within subsequent paragraphs. The subsets of instructions for the other modes of operations (Mode 1 through Mode F) are appropriately referenced.

Table 8. READ DIRECT Mode Values and General Functions Performed

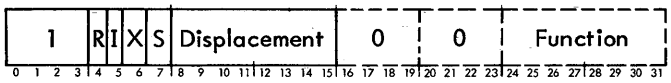
Mode		Portions of Computer System Designated	General Control or Nonarithmetic Operations Performed
Value	Title		
0	Internal Computer Control	CPU and/or IOP	1. Copy contents of specified general register or I/O channel register into the A register. 2. I/O operations via an IOP, as described in Chapter 4. 3. Copy contents of DATA switches into the A register. 4. Condition the CPU to perform a. Field Addressing instructions. b. General Register instructions. c. Doubleword instructions. d. Multiple-Register instructions. e. Floating-Point instructions. 5. Copy the first word of the Program Status Doubleword into the A register and then conditionally alter the program status indicators.
1	Interrupt Control	Interrupt System	As described in Chapter 2, under "Interrupt System".
2 thru E	Direct Control	DIO system using standard Xerox computer products	Exchange control information and data as required to perform an I/O operation. Note that each mode value is assigned to a different DIO system. (Refer to the Xerox Computer Systems/Interface Design Manual, 90 09 73, for further details.)
F	Direct Control	Specially designed equipment	Exchange control information and data as required to perform an I/O operation. (Refer to the Xerox Computer Systems/Interface Design Manual, 90 09 73, for further details.)

READ DIRECT (MODE 0) INSTRUCTIONS

The format and recommended coding of a typical READ DIRECT (Mode 0) instruction is as follows:



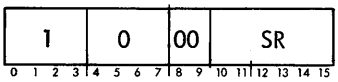
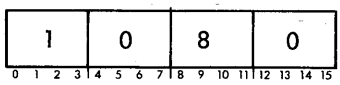
The READ DIRECT (Mode 0) instruction is unique in that if the "RIXS" bits are coded as zeros, the "function" may be encoded directly into the original instruction. Otherwise, the "RIXS" bits and the "displacement" field of the original instruction must be coded with appropriate values, which will result in an effective instruction as shown below.



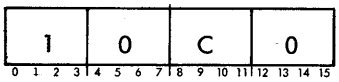
The function field may have any one of 256 different values; however, as described below, not all values of the function field are assigned as part of an effective instruction (see Appendix C).

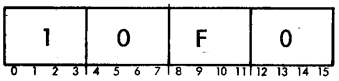
Note: Attempting to use a function value that is not assigned results in an "unimplemented instruction" fault.

For explanation purposes, the function values associated with a READ DIRECT (Mode 0) instruction are divided into four functional groups, which are also readily differentiated by the values of the first two bits of the function field (bits 8 and 9), as described below.

Function field bit positions		Control or nonarithmetic operation performed
8	9	
0	0	All values within this group (except X'00') are assigned. Each value designates a specific general register or an I/O channel register whose contents are copied into the A register. The format and recommended coding for a READ DIRECT instruction to perform the above described operation is as follows:  The "SR" field is coded with the address of the general or I/O channel register that is to be the source register for the copy operation. Affected: (A)
0	1	Except for function values X'41', X'42', X'44', X'48', X'50', and X'60', which are interpreted as I/O instructions, all function values within this group (X'40'-X'7F') are unassigned. The I/O instructions are described in Chapter 4.
1	0	Function values within this group (X'80'-X'BF') are assigned as follows: 1. Function value X'80' causes the contents of the DATA switches to be copied into the A register. The format and recommended coding for a READ DIRECT instruction to perform the above described operation is as follows:  Affected: (A) 2. Excluding function values X'80', X'81', and X'87', the function values associated with field addressing instructions are X'N0', X'N1', X'N7', X'N8', X'N9', and X'NF', where "N" may have a hexadecimal value of 8, 9, A, or B. See "Field Addressing Instructions" for further details. 3. Function values X'8A'-X'8E' are associated with general register operations, other than the A register. See "General Register Instructions" for further details. 4. Function values X'92'-X'96', X'9A'-X'9D', X'A2'-X'A4', X'AA', X'AB', and X'B2' are associated with LOAD/STORE MULTIPLE and doubleword instructions as described under "Multiple-Register Instructions". 5. Function value X'9E' causes the Floating Mode control bit (PSD9) to be set to a 1. See "Floating-Point Instructions" for further details. 6. The following function values are unassigned: X'81'-X'87', X'A5', X'A6', X'AC'-X'AE', X'B3'-X'B6', and X'BA'-X'BE'.
1	1	Within this group of function values (X'C0'-X'FF'), all values except the nine described below are unassigned. Each of

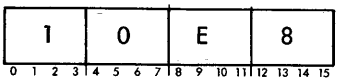
Function field bit positions 8 9	Control or nonarithmetic operation performed
1 1 (cont.)	the assigned values may be encoded into the original READ DIRECT instruction, as illustrated; their format and recommended coding, and the operations performed are as follows:



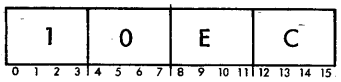


Each of the three READ DIRECT instructions causes the first word of the program Status Doubleword to be copied into the A register.

Copy first word of PSD into A register; then reset the External Interrupt (EI) bit, PSD 11, to zero.

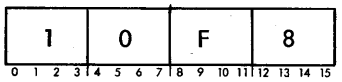


Copy the first word of PSD into A register; then reset the Internal Interrupt (II) bit, PSD 10, to zero.



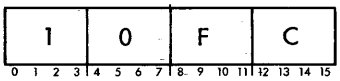
Copy the first word of PSD into A register; then reset the EI and II bits (PSD 10 and 11) to zeros.

Copy first word of PSD into A register; then set the EI bit to 1.



Copy first word of PSD into A register; then set the II bit to 1.

Function field bit positions 8 9	Control or nonarithmetic operation performed
1 1 (cont.)	



Copy first word of PSD into A register; then set the EI and II bits to 1's.

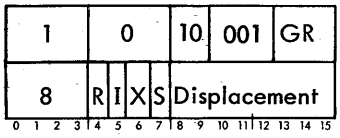
Note that bit positions within the first word of the Program Status Doubleword that do not contain program status indicators are copied into the A register as zeros.

GENERAL REGISTER INSTRUCTIONS

All general register instructions, as listed below, are provided as part of the standard repertoire of instructions.

Instruction Name	Mnemonic
Load Word	LW
Store Word	STW
Add Word	AW
Subtract Word	SW
Logical And	AND
Compare Word	CW

LW LOAD WORD



LOAD WORD is a two-word instruction sequence that loads the effective word (specified by the effective address of the second instruction word) into register L, T, X, B, or E (specified by the GR field of the first instruction word).

No interrupts are processed by the CPU between these two instruction words.

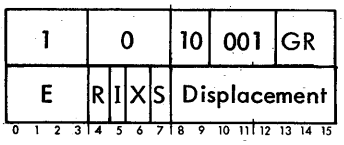
The GR field must be coded with a value of 010 through 110 and is interpreted as follows:

GR Code	Significance
010	General Register 2 (L)
011	General Register 3 (T).
100	General Register 4 (X)

GR Code	Significance
101	General Register 5 (B).
110	General Register 6 (E).

Affected: (L, T, X, B, or E)

STW STORE WORD



STORE WORD is a two-word instruction sequence that stores the contents of register L, T, X, B, or E (specified by the GR field of the first instruction word) into the effective location (specified by the effective address of the second instruction word).

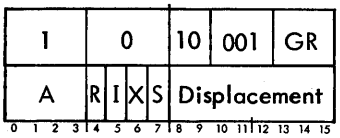
No interrupts are processed by the CPU between these two instruction words.

The GR field must be coded with a value of 010 through 110 and is interpreted as follows:

GR Code	Significance
010	General Register 2 (L).
011	General Register 3 (T).
100	General Register 4 (X).
101	General Register 5 (B).
110	General Register 6 (E).

Affected: (EL)

AW ADD WORD



ADD WORD is a two-word instruction sequence that adds the effective word (specified by the effective address of the second instruction word) to the contents of register L, T, X, B, or E (specified by the GR field of the first instruction word) and then loads the result into the specified register.

The Overflow and Carry indicators are set as described below.

O	C	Significance
1	-	The signs of the two operands are equal but the sign of the result is different.
-	1	A carry occurred from the sign bit position of the adder.
0	0	No overflow or carry occurred.

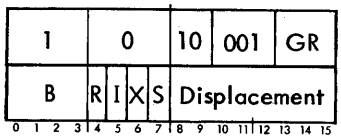
No interrupts are processed by the CPU between these two instruction words.

The GR field must be coded with a value of 010 through 110 and is interpreted as follows:

GR Code	Significance
010	General Register 2 (L).
011	General Register 3 (T).
100	General Register 4 (X).
101	General Register 5 (B).
110	General Register 6 (E).

Affected: (L, T, X, B, or E), O, C

SW SUBTRACT WORD



SUBTRACT WORD is a two-word instruction sequence that forms the one's complement of the effective word (specified by the effective address of the second instruction word), increments by 1, adds this value to the contents of register L, T, X, B, or E (specified by the GR field of the first instruction word), and then loads the result into the specified register.

The Overflow and Carry indicators are set as described below.

O	C	Significance
1	-	The sign of the result in the register is equal to the sign of the effective word but the sign of the original operand in the register was different.
-	1	A carry occurred from the sign bit position of the adder, either during incrementing the one's complement or in adding the value to the specified register: the 16-bit magnitude in the effective location is equal to or less than the 16-bit magnitude in the specified register.
0	0	No overflow or carry occurred.

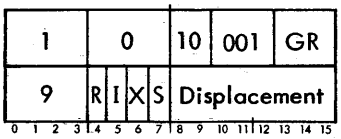
No interrupts are processed by the CPU between these two instruction words.

The GR field must be coded with a value of 010 through 110 and is interpreted as follows:

GR Code	Significance
010	General Register 2 (L).
011	General Register 3 (T).
100	General Register 4 (X).
101	General Register 5 (B).
110	General Register 6 (E).

Affected: (L, T, X, B, or E), O, C

AND LOGICAL AND



LOGICAL AND, as a two-word instruction sequence, forms the logical product between the effective word (specified by the effective address of the second word) and the contents of register L, T, X, B, or E (specified by the GR field). The logical product is loaded into the specified register.

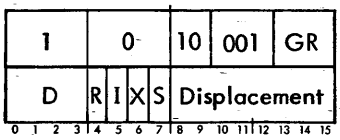
No interrupts are processed by the CPU between these two instruction words.

The GR field must be coded with a value of 010 through 110 and is interpreted as follows:

GR Code	Significance
010	General Register 2 (L).
011	General Register 3 (T).
100	General Register 4 (X).
101	General Register 5 (B).
110	General Register 6 (E).

Affected: (L, T, X, B, or E)

CW COMPARE WORD



COMPARE WORD is a two-word instruction sequence that algebraically compares the contents of register L, T, X, B, or E (specified by the GR field) and the effective word (specified by the second instruction word). Both operands are treated as signed quantities.

The Overflow and Carry indicators are set or reset according to the result of the comparison as follows:

O	C	Significance
0	0	The operand in the specified register is algebraically less than the effective word.
1	0	The operand in the specified register is algebraically greater than the effective word.
1	1	The operand in the specified register is equal to the effective word.

No interrupts are processed by the CPU between these two instruction words.

The GR field must be coded with a value of 010 through 110 and is interpreted as follows:

GR Code	Significance
010	General Register 2 (L).
011	General Register 3 (T).

GR Code	Significance
100	General Register 4 (X).
101	General Register 5 (B).
110	General Register 6 (E).

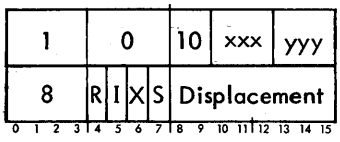
Affected: O, C

MULTIPLE–REGISTER INSTRUCTIONS

All nonfloating-point multiple-register instructions, as listed below, are included with the standard repertoire of instructions.

Instruction Name	Mnemonic
Load Multiple	LDM
Load Double	LDD
Store Multiple	STM
Store Double	STD
Double Add	DAD
Double Subtract	SDB
Compare Double	CPD

LDM LOAD MULTIPLE

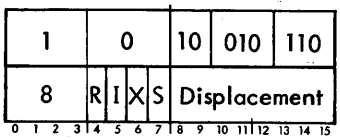


LOAD MULTIPLE is a two-word instruction sequence that loads two or more consecutive registers with a corresponding number of effective words from consecutive memory locations. The function field of the first instruction word specifies the number of registers to be loaded (value of X field), and the address of the first register (value of Y field). (The X and Y fields must be coded with values of 010 through 110.) The effective address of the second instruction word points to the first effective word in memory.

No interrupts are processed by the CPU between these two instruction words.

Affected: (specified registers)

LDD LOAD DOUBLE

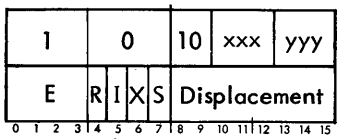


LOAD DOUBLE is a LOAD MULTIPLE instruction in which the X field is coded with a value of 2 (specifying the number of registers to be loaded) and the Y field is coded with a value of 6 (specifying the extended accumulator as the first register to be loaded). The effective address of the second instruction word points to the first effective word in memory.

No interrupts are processed by the CPU between these two instruction words.

Affected: (E), (A)

STM STORE MULTIPLE

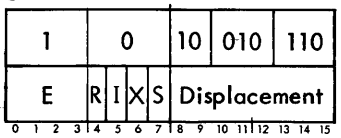


STORE MULTIPLE is a two-word instruction sequence that stores the contents of specified registers into specified effective locations. The function field of the first instruction word specifies the number of registers to be stored (value of X field), and the address of the first register (value of Y field). (The X and Y fields must be coded with values of 010 through 110.) The effective address of the second instruction word points to the first word of the effective location.

No interrupts are processed by the CPU between these two instruction words.

Affected: (specified effective locations)

STD STORE DOUBLE

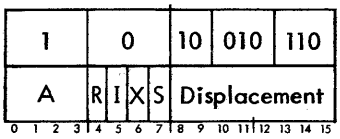


STORE DOUBLE is a STORE MULTIPLE instruction in which the X field is coded with a value of 2 (specifying the number of registers) and the Y field is coded with a value of 6 (specifying the extended accumulator as the first register). The effective address of the second instruction word points to the first of two effective locations in which the contents of the E and A registers will be stored as a doubleword.

No interrupts are processed by the CPU between these two instruction words.

Affected: (two specified effective locations)

DAD DOUBLE ADD



DOUBLE ADD is a two-word instruction sequence that adds the contents of the effective locations (as specified by the effective address of the second instruction word) to the contents of the extended accumulator and accumulator (E and A registers), as specified by the function field of the first instruction word, and then loads the result into the extended accumulator and accumulator.

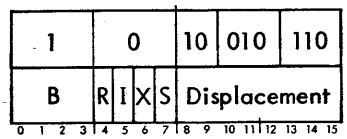
The Overflow and Carry indicators may be set as follows:

O	C	Significance
1	-	The signs of the two operands are equal but the sign of the result is different.
-	1	A carry occurred from the sign bit position of the adder.
0	0	No overflow or carry occurred.

No interrupts are processed by the CPU between these two instruction words.

Affected: (E), (A), O, C

DSB DOUBLE SUBTRACT



DOUBLE SUBTRACT is a two-word instruction sequence that forms the one's complement of the effective doubleword (as specified by the effective address of the second instruction word), increments by 1, adds this value to the contents of the extended accumulator and accumulator (as specified by the function field of the first instruction word), and then loads the result into the extended accumulator and accumulator.

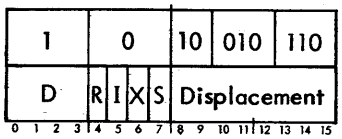
The Overflow and Carry indicators may be set as follows:

O	C	Significance
1	-	The sign of the result in the extended accumulator is equal to the sign of the effective doubleword but the sign of the original operand in the extended accumulator was different.
-	1	A carry occurred from the sign bit position of the adder, either during incrementing the one's complement or in adding the value to the extended accumulator and accumulator: the 32-bit magnitude of the effective doubleword is equal to or less than the 32-bit magnitude in the extended accumulator and accumulator.

No interrupts are processed by the CPU between these two instruction words.

Affected: (E), (A), O, C

CPD COMPARE DOUBLE



COMPARE DOUBLE is a two-word instruction sequence that algebraically compares the 32 bits of the extended accumulator and accumulator (E and A registers) and the effective doubleword. The function field of the first instruction word specifies the number of general registers involved (the value of the X field must be 2), and the address of the first register (the value of the Y field must be 6). The effective address of the second instruction word points to the effective doubleword. Both operands are treated as signed quantities.

The Overflow and Carry indicators are set or reset according to the result of the comparison as follows:

O	C	Result of Comparison
0	0	The 32-bit operand in the extended accumulator and accumulator is algebraically less than the effective doubleword.
1	0	The 32-bit operand in the extended accumulator and accumulator is algebraically greater than the effective doubleword.
1	1	The 32-bit operand in the extended accumulator and accumulator is equal to the effective doubleword.

No interrupts are processed by the CPU between these two instruction words.

Affected: O, C

FLOATING–POINT INSTRUCTIONS

The floating-point feature consists of the hardware implementation of seven optional floating-point instructions:

Instruction Name	Mnemonic
Floating Load	FLD
Floating Store	FST
Floating Add	FAD
Floating Subtract	FSB
Floating Multiply	FMP
Floating Divide	FDV
Floating Compare	FCP

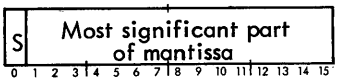
Each of these floating-point instructions is evoked by executing the corresponding fixed-point instruction (LOAD, STORE, ADD, SUBTRACT, MULTIPLY, DIVIDE, or COMPARE) with the Floating Mode (FM) bit in the Program Status Doubleword (PSD) set to a 1 (see "Floating-Point Mode Control").

All floating-point instructions operate on the Scratchpad Floating Accumulator (SFA) described below and a specified three-word floating operand contained within three contiguous memory locations. (See "Floating-Point Numbers" for other characteristics of floating-point operands.) The effective address of the floating-point instruction, which may be generated using any mode of effective address computation available to memory reference type of instructions, points to the first location containing the floating operand. Except for FLOATING STORE and FLOATING COMPARE instructions, the results of all floating-point instructions replace the original contents of the SFA and the floating operand in main memory is not affected. For FLOATING STORE instructions, the contents of the SFA is stored into the floating operand. For FLOATING COMPARE instructions, neither the contents of SFA nor the floating operand is modified.

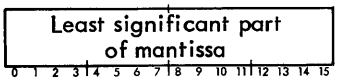
FLOATING–POINT NUMBERS

An extended precision floating-point number consists of the three consecutive 16-bit words with the following format:

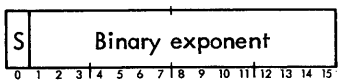
First word:



Second word:



Third word:



The first two words contain a signed two's complement mantissa with the binary point following bit 0 of the first word. The third word contains a signed two's complement integer binary exponent.

A floating-point number (N) has the following formal definition:

1. A nonzero floating-point number has the value

   $N = (\text{mantissa})(2^X)$

   where X represents the binary exponent.

2. A positive floating-point number with a mantissa of zero and a binary exponent of zero is called a "true" zero. All floating-point operations that produce a zero result will always produce a "true" zero.

3. A positive nonzero floating-point number is normalized if and only if the mantissa is contained in the interval

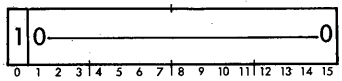
   $1/2 \leq \text{mantissa} < 1$

4. A negative floating-point number is formed by taking the two's complement of the mantissa of its positive representation. The binary exponent portion of a negative floating-point number is identical to the binary exponent portion of its positive representation.

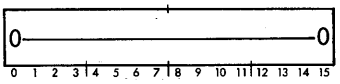
5. A negative floating-point number is normalized if and only if the floating-point number formed by taking the two's complement of the mantissa is a normalized positive floating-point number.

By this definition, a floating-point number of the form

First word:



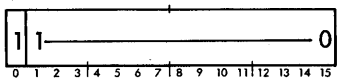
Second word:



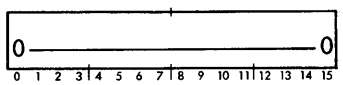
Third word: Binary Exponent = X

is not normalized because the two's complement of the mantissa is not a normalized positive number. Since all floating-point arithmetic instructions produce a result that is either a true zero or a normalized floating-point number, whenever a number of the form shown above might be generated, it is converted by the hardware into a floating-point number of the form

First word:



Second word:



Third word: Binary Exponent = Y

where the binary exponent Y is one greater than the binary exponent X.

Table 9 contains examples of floating-point numbers.

Table 9. Floating-Point Numbers

Decimal Number	Extended Precision Floating-Point Number		
	Mantissa		Exponent
	First Word	Second Word	Third Word
0 (called true zero)	X'0000'	X'0000'	X'0000'
$+1 = (1/2) \times 2^{1}$	X'4000'	X'0000'	X'0001'
$-1 = -(1/2) \times 2^{1}$	X'C000'	X'0000'	X'0001'
$+10 = (5/8) \times 2^{4}$	X'5000'	X'0000'	X'0004'
$-10 = -(5/8) \times 2^{4}$	X'B000'	X'0000'	X'0004'
$+100 = (25/32) \times 2^{7}$	X'6400'	X'0000'	X'0007'
$-100 = -(25/32) \times 2^{7}$	X'9C00'	X'0000'	X'0007'
$(1-2^{-31}) \times 2^{32767} \approx 10^{9864}$	X'7FFF'	X'FFFF'	X'7FFF'
$(1-2^{-31}) \times 2^{-32768} \approx 10^{-9864}$	X'7FFF'	X'FFFF'	X'8000'

EXPONENT UNDERFLOW AND OVERFLOW

Exponent underflow occurs during any floating-point arithmetic operation whenever an arithmetic operation results in a binary exponent that is a larger negative number than can be properly represented in two's complement form in 16 bits including the sign. Whenever exponent underflow occurs during the operation of a floating-point instruction, the result of that instruction will be a true zero.

Exponent overflow occurs during any floating-point arithmetic operation whenever an arithmetic operation results in a binary exponent that is a larger positive number than can be properly represented in 16 bits including the sign. An attempt to divide by zero during a floating-point divide operation will also cause exponent overflow. Whenever exponent overflow occurs during the operation of a floating-point instruction, the result of that instruction will be the largest possible normalized floating-point number with the same sign as the correct result of the instruction would have had if exponent overflow had not occurred. If the correct result of the floating-point instruction would have been positive and exponent overflow occurs, the hexadecimal result of the instruction will be

first word: X'7FFF'

second word: X'FFFF'

third word: X'7FFF'

If the correct result of the floating-point instruction would have been negative and exponent overflow occurs, the hexadecimal result of the instruction will be

first word: X'8000'

second word: X'0001'

third word: X'7FFF'

The Overflow (O) indicator is set or reset at the conclusion of each arithmetic floating-point instruction to indicate whether exponent overflow or underflow occurred during that instruction as follows:

O	Result of Instruction
0	Neither exponent overflow nor exponent underflow occurred.
1	Exponent overflow or exponent underflow occurred.

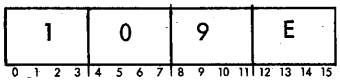
FLOATING-POINT MODE CONTROL

Hardware-implemented floating-point operations are automatically invoked whenever bit 9 (FM) of the Program Status Doubleword is set to a 1. The LOAD, STORE, ADD, SUBTRACT, MULTIPLY, DIVIDE, and COMPARE instructions

are automatically modified into FLOATING LOAD, FLOATING STORE, FLOATING ADD, FLOATING SUBTRACT, FLOATING MULTIPLY, FLOATING DIVIDE, and FLOATING COMPARE, respectively. All other instructions are not affected by the state of FM.

The FM bit is set to a 1 whenever any of the following operations are performed:

1. By executing a SET FLOATING MODE (SFM) instruction. The format and required coding of the SFM instruction is as follows:



   The SFM instruction is a nonprivileged READ DIRECT (Mode 0) instruction that can be executed from either protected or unprotected memory. The "RIXS" bits of the instruction must all be coded as zeros.

   If an attempt is made to set FM when the floating-point option is not installed, bit 9 of the PSD remains a zero and machine fault interrupt is triggered.

2. If bit 9 of the A register (general register 7) is a 1, it will be copied into the FM position whenever the privileged WRITE DIRECT instruction with a final effective address of X'00C0' is executed. This WD instruction can only be executed from protected memory.

3. If the FM bit was a 1 at the time it was stored (saved) by an interrupt entry procedure, the FM bit will be restored as a 1 when exiting the interrupt routine (a WRITE DIRECT instruction with a final effective address of X'00D8' followed immediately with a LOAD INDEX (LDX) instruction). Bit 9 of the effective word for the LDX instruction contains the FM bit that was saved.

Hardware-implemented floating-point operations are automatically inhibited whenever the FM bit of the PSD is reset. The FM bit is reset to 0 by any of the following conditions or operations:

1. Whenever a general CPU reset signal is generated (occurs when primary power is initially applied to the system and when the RESET switch on the Processor Control Panel is raised).

2. By executing an unconditional BRANCH (B) instruction from either protected or unprotected memory.

3. If bit 9 of the A register (general register 7) is a zero, it will be copied into the FM position whenever the privileged WRITE DIRECT instruction with a final effective address of X'00C0' is executed. This WD instruction can only be executed from protected memory.

4. When an interrupt level moves from the waiting state to the active state as the result of the occurrence of an interrupt that alters the normal sequence of instructions, the state of the FM bit will be saved with the rest of the current PSD at the memory location specified by the contents of the interrupt location. After saving the previous state of the FM bit, the FM bit is reset.

The status of the FM bit (as well as other program status indicators) can be read under program control by privileged READ DIRECT (Mode 0) instructions that copy the first word of the Program Status Doubleword into the A register.

SCRATCHPAD FLOATING ACCUMULATOR

A Scratchpad Floating Accumulator (SFA), consisting of three 16-bit high-speed registers, is used in conjunction with all hardware floating-point instructions. The contents of the SFA can be modified and read only by executing hardware floating-point instructions. For all hardware floating-point instructions, the SFA contains one of the initial arguments prior to execution of the instruction. At the conclusion of all arithmetic floating-point instructions, the initial argument in SFA is replaced by the result of the floating-point instruction.

Whenever an operation occurs that causes the FM bit of the PSD to be reset, the contents of SFA is automatically saved in memory by storing it in three contiguous memory locations (called Memory Floating Accumulator). This automatic transfer will occur regardless of how the FM bit is reset (including interrupt entry) except that no transfers will occur when the general CPU reset signal occurs. The address of the first location of the Memory Floating Accumulator is specified by the contents of memory location 1.

Note: If the program is operating under Basic Control Monitor (BCM) or Real-Time Batch Monitor (RBM), the Memory Floating Accumulator is the first three locations in the user's temporary stack in memory.

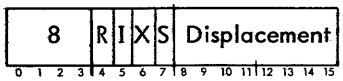
Whenever an operation occurs that causes the FM bit of the PSD to be set, the contents of the Memory Floating Accumulator replaces the previous contents of SFA. This automatic transfer will occur regardless of how the FM bit is set, including interrupt exit where the interrupted program being restored was operating with the FM bit set.

The contents of SFA will also be transferred automatically to the Memory Floating Accumulator when the FM bit is set and the COMPUTE switch on the Processor Control Panel (PCP) is switched from RUN or STEP to IDLE.

When the COMPUTE switch is switched from IDLE to RUN or STEP and the FM bit is set, the contents of the Memory Floating Accumulator replaces the previous contents of SFA prior to an instruction execution. The availability of the contents of SFA in memory when the CPU is in an IDLE state facilitates testing and debugging floating-point operations from the PCP.

FLD FLOATING LOAD (optional)

If FM is set, the execution of the following instruction will result in a floating-point load operation:

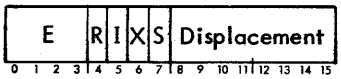


The effective address of the instruction is the address of the first word of the floating operand. FLD loads the floating operand into SFA. The first word of the floating operand also replaces the previous contents of the A register (general register 7).

Affected: SFA, (A)

FST FLOATING STORE (optional)

If FM is set, the execution of the following instruction will result in a floating-point store operation:

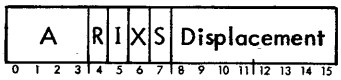


The effective address of the instruction is the address of the first word of the floating operand. FST replaces the floating operand with the contents of SFA. The first word of SFA also replaces the previous contents of the A register (general register 7).

Affected: Floating operand, (A).

FAD FLOATING ADD (optional)

If FM is set, the execution of the following instruction will result in a floating-point add operation:



The effective address of the instruction is the address of the first word of the floating operand. FAD adds the floating operand to the contents of the SFA and then loads the result into SFA. The first word of the result also replaces the previous contents of the A register (general register 7).

Prior to the FAD, SFA and the floating operand must both contain either a normalized nonzero number or a true zero. The result in SFA after FAD is always either a normalized nonzero number or a true zero. The result is unpredictable if, prior to the FAD, either SFA or the floating operand contains a floating-point number that is neither a normalized nonzero number nor a true zero. There is one exception to the previous statement; if the floating operand contains a true zero and SFA contains neither a normalized nonzero number nor a true zero, FAD will replace the contents of SFA with either the properly normalized nonzero number or a true zero.

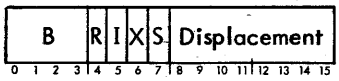
During the operation of FAD no guard digits are used. If prealignment shifting of one of the arguments is required, none of the bits shifted off the right end of the mantissa are preserved. If postnormalization of the result is required, zeros will be shifted into the right end of the mantissa.

If exponent underflow occurs, the Overflow indicator is set to 1 and the result stored in SFA is a true zero. If exponent overflow occurs, the Overflow indicator is set to 1 and the result stored in SFA is the largest possible normalized floating-point number with the same sign as the correct result of the instruction would have had if exponent overflow had not occurred. If neither exponent underflow nor overflow occurs, the Overflow indicator is set to 0.

Affected: SFA, (A), O

FSB FLOATING SUBTRACT (optional)

If FM is set, the execution of the following instruction will result in a floating-point subtract operation:



The effective address of the instruction is the address of the floating operand. FSB subtracts the floating operand from the contents of SFA and then loads the results into SFA. The first word of the result also replaces the previous contents of the A register (general register 7).

Prior to the FSB, SFA and the floating operand must both contain either a normalized nonzero number or a true zero. The result in SFA after FSB is always either a normalized nonzero number or a true zero. The result is unpredictable if, prior to the FSB, either SFA or the floating operand contains a floating-point number that is neither a normalized nonzero number nor a true zero.

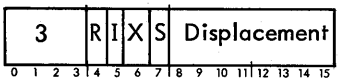
During the operation of FSB no guard digits are used. If prealignment shifting of one of the arguments is required, none of the bits shifted off the right end of the mantissa are preserved. If postnormalization of the result is required, zeros will be shifted into the right end of the mantissa.

If exponent underflow occurs, the Overflow indicator is set to 1 and the result stored in SFA is a true zero. If exponent overflow occurs, the Overflow indicator is set to 1 and the result stored in SFA is the largest possible normalized floating-point number with the same sign as the correct result of the instruction would have had if exponent overflow had not occurred. If neither exponent underflow nor overflow occurs, the Overflow indicator is set to 0.

Affected: SFA, (A), O

FMP FLOATING MULTIPLY (optional)

If FM is set, the execution of the following instruction will result in a floating-point multiply operation:



The effective address of the instruction is the address of the first word of the floating operand. FMP multiplies the floating operand by the contents of SFA and then loads the result into SFA. The first word of the result also replaces the previous contents of the A register (general register 7).

Prior to the FMP, SFA and the floating operand must both contain either a normalized nonzero number or a true zero. The result in SFA after FMP is always either a normalized nonzero number or a true zero. The result is unpredictable if, prior to the FMP, either SFA or the floating operand contains a floating-point number that is neither a normalized nonzero number nor a true zero.

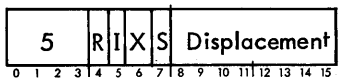
All the bits of the mantissa of the normalized product will be significant. One additional bit of the product will be generated so that if a one-place postnormalization shift is required, a significant bit can be shifted into the right end of the mantissa rather than a zero.

If exponent underflow occurs, the Overflow indicator is set to 1 and the result stored in SFA is a true zero. If exponent overflow occurs, the Overflow indicator is set to 1 and the result stored in SFA is the largest possible normalized floating-point number with the same sign as the correct result of the instruction would have had if exponent overflow had not occurred. If neither exponent underflow nor overflow occurs, the Overflow indicator is set to 0.

Affected: SFA, (A), O

FDV FLOATING DIVIDE (optional)

If FM is set, the execution of the following instruction will result in a floating-point divide operation:



The effective address of the instruction is the address of the first word of the floating operand. FDV divides the contents of SFA by the floating operand and then loads the result into SFA. The first word of the result also replaces the previous contents of the A register (general register 7).

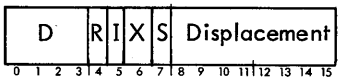
Prior to the FDV, SFA and the floating operand must both contain either a normalized nonzero number or a true zero. The result in SFA after FDV is always either a normalized nonzero number or a true zero. The result is unpredictable if, prior to the FDV, either SFA or the floating operand contains a floating-point number that is neither a normalized nonzero number nor a true zero.

If exponent underflow occurs, the Overflow indicator is set to 1 and the result stored in SFA is a true zero. If the floating operand is zero or if exponent overflow occurs, the Overflow indicator is set to 1 and the result stored in SFA is the largest possible normalized floating-point number with the same sign that the correct result of the instruction would have had if exponent overflow had not occurred. If neither exponent underflow nor overflow occurs, the Overflow indicator is set to 0.

Affected: SFA, (A), O

FCP FLOATING COMPARE (optional)

If FM is set, the execution of the following instruction will result in a floating-point compare operation:



The effective address of the instruction is the address of the first word of the floating operand. FCP compares the floating operand to the contents of SFA. The result of the comparison is used to set and reset the Overflow and Carry indicators as follows:

O	C	Result of Comparison
0	0	The contents of SFA is less than the floating operand.
1	0	The contents of SFA is greater than the floating operand.
1	1	The contents of SFA is equal to the floating operand.

The contents of the A register (general register 7) are not affected by the execution of FCP. The result of FCP will be unpredictable if either SFA or the floating operand contains a floating-point number that is neither a normalized nonzero number nor a true zero.

Affected: O, C

FIELD ADDRESSING INSTRUCTIONS

All of the following field addressing instructions are optional:

Instruction Name	Mnemonic
Load Logical Field	LLF
Load Arithmetic Field	LAF
Store Field	STF

Instruction Name	Mnemonic
Store Zero Field	SZF
Store Ones Field	SOF
Compare Logical Field	CLF
Compare Arithmetic Field	CAF
Sense Left Bit of Field	SLF

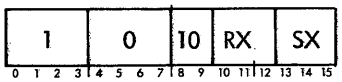
Field addressing instructions facilitate specifying and operating upon fields of information (1 through 16 contiguous bits) within memory without restriction in regard to byte or word boundaries (i.e., a specified field of 16 bits may occupy bit positions 9 through 15 of memory location "n" and bit positions 0 through 8 of memory location "n + 1"). The ability to perform bit and byte manipulations as well as push-down stack operations permit field addressing instructions to operate efficiently upon component elements (fields) of three types of logical structures.

1. Tables – Field addressing instructions facilitate the creation of table structures (collections of different sized fields) that make efficient use of memory and allow direct access to any element within those tables. Table structures utilized by field addressing instructions can be modified without requiring the modification of the code that references those tables.

2. Strings – Field addressing instructions efficiently operate on strings of identically sized elements. Loop termination control, without the use of any of the general registers, is provided for strings of up to and including 256 elements. Byte string operations are an example of such a string.

3. Push-Down Stacks – Field addressing instructions allow the creation of any number of push-down stacks in memory. Various modes of field addressing instructions allow pushing new elements into any stack, pulling the top element from any stack, accessing words within a stack that are a known number of word locations away from the current top-of-stack and also detecting when the last available space within a stack has been used.

A typical field addressing instruction consists of two contiguous instruction words. The first instruction word is a nonprivileged READ DIRECT (Mode 0) instruction with the following format:



The SX field of the function value must be coded as either 000, 001, or 111. The SX field specifies one of three self-indexing modes as follows:

Function Value of READ DIRECT	Self-Indexing Mode
0 0 0 0 0 0 0 0 1 0 [RX] 0 0 0	No Self-Indexing
0 0 0 0 0 0 0 0 1 0 [RX] 0 0 1	Self-Incrementing
0 0 0 0 0 0 0 0 1 0 [RX] 1 1 1	Self-Decrementing

Self-incrementing and self-decrementing are described further below.

For all three self-indexing modes, the RX field causes identical action and must be coded to some three-bit value (other than 000). The RX field specifies one of two register-indexing modes as follows:

RX Field	Register-Indexing Mode
0 0 1	No Register Indexing
0 1 0 0 1 1 1 0 0 1 0 1 1 1 0 1 1 1	Register Indexing

Register indexing is described further below.

The second instruction word has the format of a single-word referencing instruction.

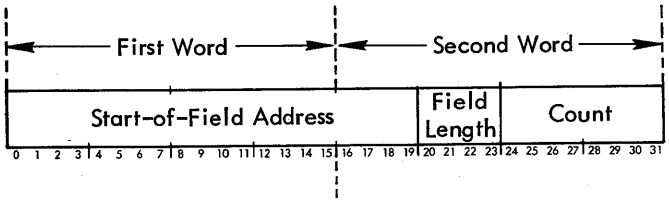
The operation code (OP) field indicates the specific field manipulation operation that is to be performed: Load Logical Field, Load Arithmetic Field, Store Field, Store Zero Field, Store Ones Field, Compare Logical Field, Compare Arithmetic Field, and Sense Left Bit of Field.

The effective address of the second instruction word is the address of the first word of a two-word "field descriptor" in main memory. This effective address may be generated using any mode (described under "Effective Address Computation"), thereby allowing, for example, indexing into a table of field descriptors.

No interrupts are processed by the CPU between these two instruction words.

FIELD DESCRIPTOR

The two-word field descriptor may start on any word boundary and has the following format:



START-OF-FIELD ADDRESS

If no self-indexing is specified (SX field of the first instruction word is coded 000) and if no register indexing is specified (RX field is coded 001), the first 20 bits (all 16 bits of word 1 and the first 4 bits of word 2) of the field descriptor (unmodified) become the effective start-of-field address. Otherwise, these 20 bits are modified as specified by the field addressing instruction to generate the effective start-of-field address. The final result of any and all address modification is the bit-address of the first (leftmost) bit of the effective field (the field actually operated on by the field addressing instruction). The first 16 bits of the effective start-of-field address point to the word containing the first bit of the effective field and the next 4 bits specify the bit position occupied by that bit within that word.

FIELD LENGTH

This 4-bit field is contiguous to the start-of-field address and specifies the length (number of bits) of the effective field. The value in this field is one less than the number of bits in the field (i.e., 0000 indicates a 1-bit field, 0001 indicates a 2-bit field, ..., 1111 indicates a 16-bit field.

COUNT

This 8-bit field is used in conjunction with self-incrementing and self-decrementing to provide loop termination control when operating on a string of identically sized elements or to provide a "stack-full" indication for push-down stack operations.

SELF-INCREMENTING OF THE START-OF-FIELD ADDRESS

If self-incrementing is specified (SX field of the first instruction word is coded with 001), the start-of-field address in the field descriptor is incremented by the number of bits in the field (field length plus one), the count field in the field descriptor is incremented by one and then the field descriptor as modified by these two addition operations replaces the original field descriptor in memory. The addition of one to the count field (bits 24-31) is performed modulo 256 so that the field length field (bits 20-23) as restored to memory is always unmodified from the original field length.

This new incremented start-of-field address is either the effective start-of-field address or is used as the basis for register indexing as described subsequently.

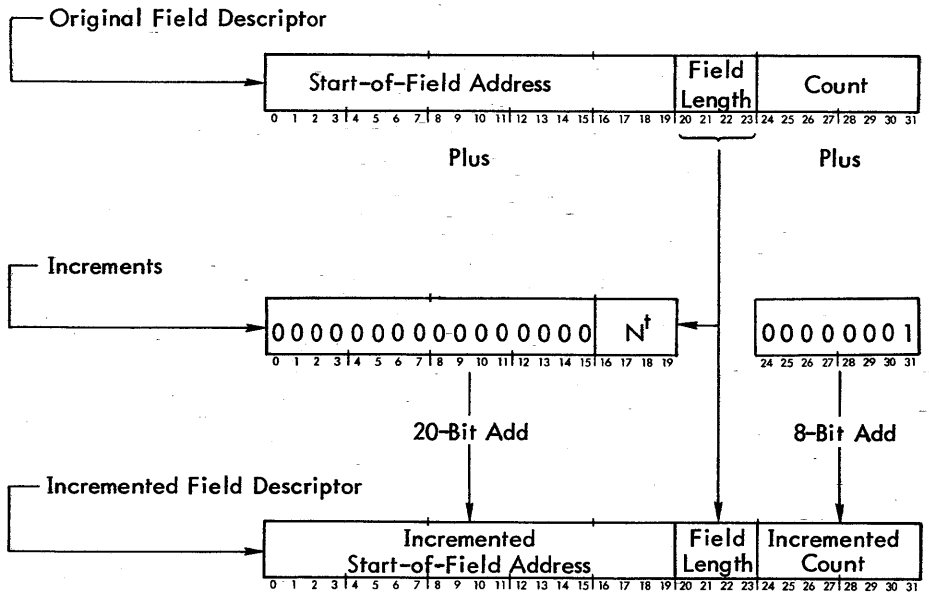
The self-incrementing operation is illustrated in Figure 4.

For all field addressing instructions, except COMPARE LOGICAL FIELD and COMPARE ARITHMETIC FIELD, the Overflow indicator (O) is set or reset depending on whether overflow occurred when incrementing the count field (assuming the count field to be an eight-bit positive integer). The Overflow indicator is set if the incremented count restored to memory is equal to all zeros and is reset otherwise. For COMPARE LOGICAL FIELD and COMPARE ARITHMETIC FIELD, the Overflow indicator is used to indicate the results of a comparison and is not affected by the contents of the incremented count field.

Self-incrementing is useful as part of the inner loop when operating sequentially on every element in a string of elements of identical size. If prior to the field addressing instruction, the field descriptor points at the last element processed, then a field addressing instruction with self-incrementing will modify the field descriptor to point at the next element in the string as part of the effective start-of-field address computation. In this case, the count field can be used to provide loop termination control. If prior to starting to process a string of identical sized elements, the count field in the initial descriptor contains a number equal to 256 minus the number of elements in the string, then, when the string is processed using field addressing instructions with self-incrementing, the Overflow indicator will be set by the field addressing instruction that processes the last element.

Self-incrementing is also used for operating on push-down stacks. If a push-down stack consists of a sequential string of identical sized elements with the current top-of-stack element having an address larger than the address of the other elements within the stack, then self-incrementing is used for pushing a new element into the stack. If prior to a field addressing instruction, the field descriptor points at the current top-of-stack element, then a Store Field instruction with self-incrementing will push a new element into the stack and leave the field descriptor pointing to that new element as the current top-of-stack. In this case the count field can be used to indicate that the stack is full. If the initial descriptor that is set up when the push-down stack is empty contains a number in the count field equal to 256 minus the maximum allowable size of the push-down stack, then the Overflow indicator will be set following the field addressing instruction that pushed a new element into the last available space in the stack.

If self-incrementing is specified and no register indexing is specified (RX field of the first instruction word is coded with 001), then the effective start-of-field address is equal to the incremented start-of-field address in the field descriptor as restored to memory.



The incremented field descriptor is restored to memory and also used in subsequent effective start-of-address computation.

[t]N is the number of bits in the effective field (equal to the contents of the field-length field plus one).

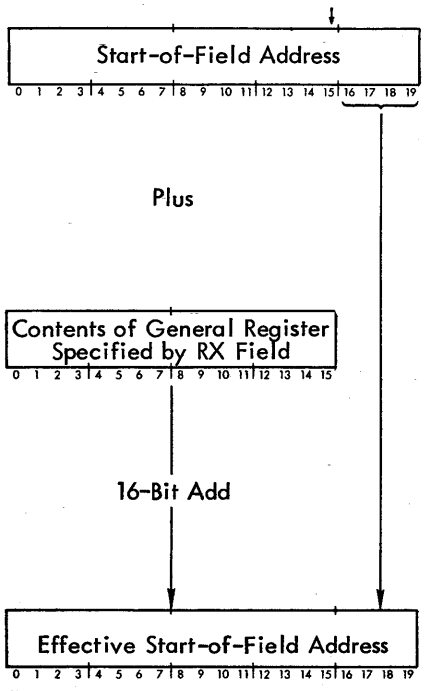
Figure 4. Self-Incrementing Operation

REGISTER INDEXING OF THE START-OF-FIELD ADDRESS

Register indexing can be invoked in conjunction with any of the self-indexing modes. Register indexing adds the contents of one of the general registers to the word address portion (bits 0-15) of the start-of-field address contained in the field descriptor in memory. If either self-decrementing or no self-indexing is specified, the field descriptor in memory at the time register indexing is performed will be identical to the original field descriptor in memory. If self-incrementing is specified, the field descriptor in memory at the time register indexing is performed will be the incremented result of the self-incrementing operation. The RX field of the first instruction word specifies whether register indexing is to be invoked and if so, which general register is to be used.

RX Field	General Register
001	No register indexing
010	L
011	T
100	X
101	B
110	E
111	A

If register indexing is specified, the operation performed on the start-of-field address currently contained in the field descriptor in memory is illustrated by the following diagram:



Register indexing can only affect the word address portion of the effective start-of-field address. The result of the register indexing addition is the effective start-of-field address; however, the field descriptor in memory is not modified as a result of register-incrementing. Note this important distinction between self-incrementing and register indexing. Whenever self-incrementing is performed, the result replaces the original field descriptor in memory whereas the results of register-indexing never affects the field descriptor in memory.

As an example of the use of register indexing, consider a table made up of "n" sequential 16-bit words in memory. Each word contains a collection of software status associated with a particular I/O device. The I/O devices are arbitrarily numbered sequentially from "0" to "n-1" for a system with "n" devices. The first word of the table contains software status information associated with device number "0", the second word contains software status information associated with device number "1", etc. The format of each word in the table is identical; the same specific piece of status information for each device appears in the same bit positions in each word of the table. Given such a table structure, a set of field descriptors would exist (one for each different piece of status information in the table). These field descriptors would all have a start-of-field address indicating the left bit of the appropriate field in the table entry for device "0". This same set of descriptors would then be used to locate the appropriate status information for any device. This would be accomplished by loading one of the general registers 2 through 7 with the device number of the device of interest. Then, a field addressing instruction would be executed pointing at the field descriptor for the desired piece of status information for device "0" and invoking register indexing using the register containing the device number.

Register indexing can also be used to reference elements contained within a push-down stack provided that the elements within the push-down stack are all 16-bit words. A word that is known to be "n" words below the current top-of-stack word can be accessed by loading one of the general registers 2 through 7 with a number equal to "-n". Then, a field addressing instruction would be executed with a field descriptor which points to the current top-of-stack word and invoking register-indexing using the register containing the value "-n".

OTHER CHARACTERISTICS OF FIELD ADDRESSING INSTRUCTIONS

Information presented thus far has been associated with the identification of the effective field (generating an effective start-of-field address). This section briefly describes the operation performed on the effective field by the various field addressing instructions.

The eight field addressing instructions fall logically into four categories: load, store, compare, and sense. In all of the following descriptions, the A register refers to general register 7.

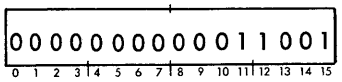
There are two load instructions, LOAD LOGICAL FIELD and LOAD ARITHMETIC FIELD. LOAD LOGICAL FIELD loads the effective field into the right-hand end of the A register (i.e., the rightmost bit of the effective field replaces bit 15 of A) and loads zeros into the remainder of the A register. LOAD ARITHMETIC FIELD loads the effective field into the right hand end of the A register and replaces the remainder of the bits in the A register with the leftmost bit of the effective field. LOAD ARITHMETIC FIELD treats the field to be a signed quantity and, therefore, loading the remainder of the bits in the A register with the leftmost bit of the field is really sign extension.

As an example of the two load operations, consider a five-bit effective field with a binary configuration of

Effective Field | 1 1 0 0 1 |

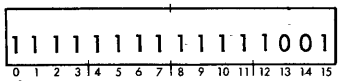
The final result in the A register for each of the load instructions is as follows:

Result of LOAD LOGICAL FIELD



A Register

Result of LOAD ARITHMETIC FIELD



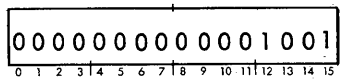
A Register

However, if the five-bit effective field has a binary configuration of

Effective Field | 0 1 0 0 1 |

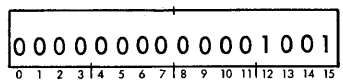
Then the result in the A register of each of the load instructions is as follows:

Result of LOAD LOGICAL FIELD



A Register

Result of LOAD ARITHMETIC FIELD



A Register

If the field length is 16 bits, the results of LOAD LOGICAL FIELD and LOAD ARITHMETIC FIELD are identical regardless of the binary configuration of the field.

There are three store instructions, STORE FIELD, STORE ZERO FIELD, and STORE ONES FIELD. STORE FIELD replaces the effective field in memory by the "n" rightmost bits of the A register where "n" is equal to the number of bits in the effective field. For example, if the contents of the field length field specifies a field length of three, then

STORE FIELD replaces the effective field in memory by the contents of bits 13-15 of the A register. The contents of the A register is not affected by a STORE FIELD instruction. STORE ZERO FIELD replaces every bit of the effective field in memory with zeros. STORE ONES FIELD replaces every bit of the effective field with ones. For all store instructions, no other bits in the memory word or words containing the effective field are modified except those bits which are part of the effective field.

If the contents of the field length field in the field descriptor specifies a field length of one bit, STORE ZERO FIELD and STORE ONES FIELD perform the function of clearing or setting a particular bit in memory.

There are two compare instructions, COMPARE LOGICAL FIELD, and COMPARE ARITHMETIC FIELD. COMPARE LOGICAL FIELD forms a 16-bit word containing the effective field at the right-hand end and zeros in all other bit positions and then compares this word with the contents of the A register. The Overflow and Carry indicators are set or reset as in the fixed-point compare instruction to indicate the result of the comparison. COMPARE ARITHMETIC FIELD is the same as COMPARE LOGICAL FIELD except that the 16-bit word compared to the contents of the A register is formed by sign extending the leftmost bit of the effective field into all other bit positions rather than zeros. Neither compare instruction modifies the contents of the A register.

There is one sense instruction called SENSE LEFT BIT OF FIELD. This instruction loads the leftmost bit of the effective field into the Carry indicator (C) without modifying either the effective field or the A register. If the contents of the field length field in the field descriptor specifies a field length of one bit, SENSE LEFT BIT OF FIELD performs the function of testing the state of a particular bit in memory.

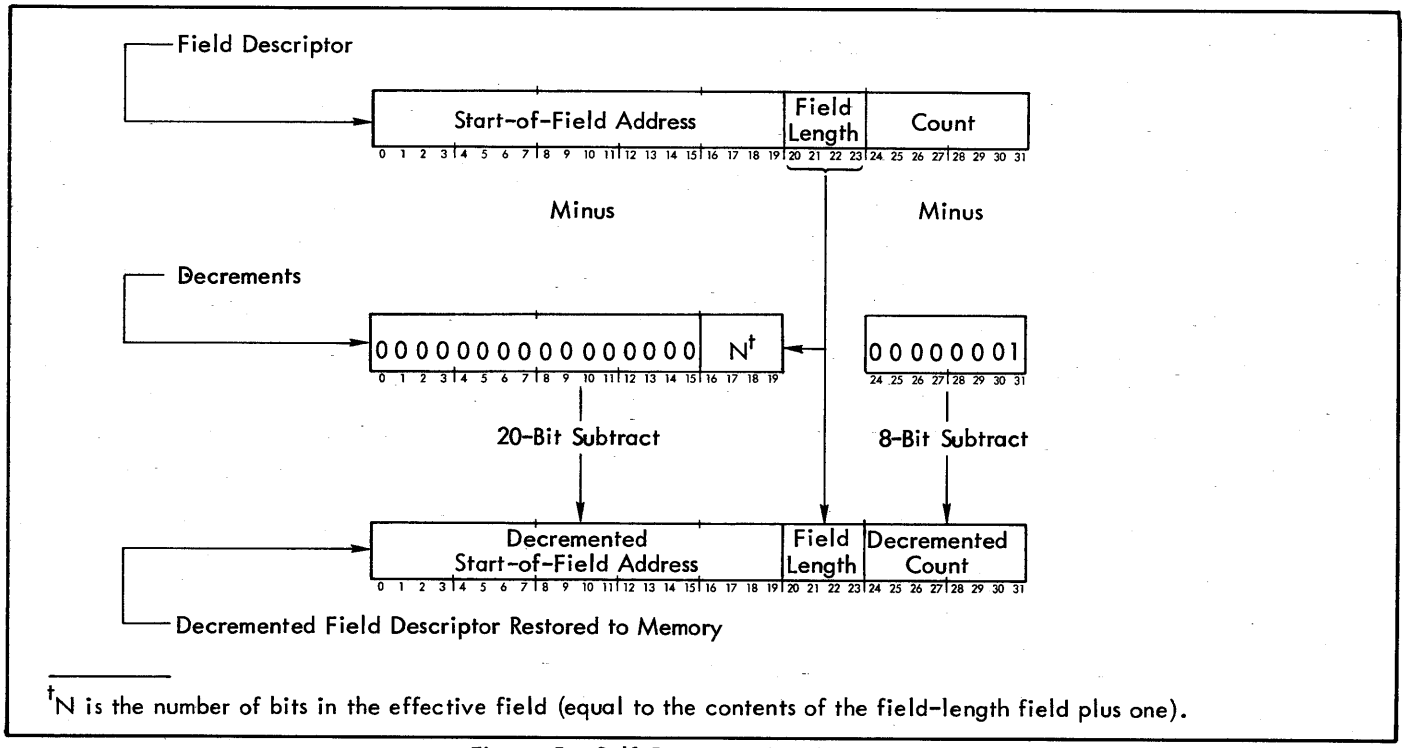
SELF-DECREMENTING OF THE START-OF-FIELD ADDRESS

Self-decrementing has no effect on the effective start-of-field address calculation for the current field addressing instruction. Self-decrementing is essentially the converse operation to self-incrementing. However, where self-incrementing modifies the field descriptor in memory and affects the effective start-of-field address calculation, self-decrementing is performed after the operation on the effective field is complete and, therefore, only modifies the field descriptor in memory. Self-decrementing will, however, affect the effective start-of-field address of any subsequent field instructions using the same field descriptor. Self-incrementing and self-decrementing cannot both be invoked in the same field addressing instruction.

If self-decrementing is specified (SX field of first instruction word is coded with 111), then, after all operations involving the effective field are completed, the start-of-field address in the field descriptor in memory is decremented by the number of bits in the field, the count field in the field descriptor is decremented by one, and then the field descriptor as modified by these two subtraction operations replaces the original field descriptor in memory.

The subtraction of one from the count field (bits 24-31) is performed modulo 256 so that the field length (bits 20-23) as restored to memory is always unmodified from the original field length.

The self-decrementing operation is illustrated in Figure 5.



†N is the number of bits in the effective field (equal to the contents of the field-length field plus one).

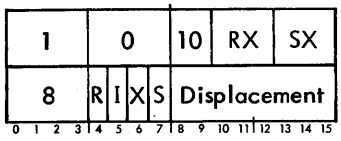
Figure 5. Self-Decrementing Operation

For all field addressing instructions, except COMPARE LOGICAL FIELD and COMPARE ARITHMETIC FIELD, the Overflow indicator (O) is set or reset depending on whether overflow occurred when decrementing the count field (assuming the count field to be an eight-bit positive integer). The Overflow indicator is set if the decremented count restored to memory is equal to all ones and is reset otherwise. For COMPARE LOGICAL FIELD and COMPARE ARITHMETIC FIELD, the Overflow indicator is used to indicate the results of a comparison and is not affected by the contents of the decremented count field.

If self-decrementing and register-indexing are both specified, the register indexing operation is performed on the original field descriptor as part of the effective start-of-field address calculation as described under "Register Indexing of the Start-of-Field Address". Then the resultant effective start-of-field address is used to find the effective field and the operation is performed as described under "Other Characteristics of Field Addressing Instructions". After all operations involving the effective field are completed, the self-decrementing takes place. The original field descriptor from memory is used for self-decrementing as described above.

Self-decrementing is used for operating on push-down stacks of the type previously described under "Self-Incrementing of the Start-of-Field Address". If, prior to a field addressing instruction, the field descriptor points at the current top-of-stack element, then a load field instruction with self-decrementing will pull the current top-of-stack element off of the stack and load it into the A register and then decrement the field descriptor so that it is pointing to the next element down in the stack as the current top-of-stack.

LLF LOAD LOGICAL FIELD (optional)



LOAD LOGICAL FIELD is a two-word instruction sequence that loads the effective field into the right-hand end of the A register (general register 7). The rightmost bit of the effective field replaces bit 15 of the A register, etc. If the number of bits in the effective field is less than 16, then zeros replace the remainder of bits in the A register.

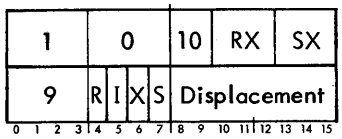
The effective address of the second instruction word is the address of the first word of a 32-bit field descriptor. The contents of the field descriptor in conjunction with the contents of the RX and SX fields of the first instruction word are used to locate the effective field. The contents of the SX field may also cause the field descriptor in memory to be modified.

If self-incrementing is specified and the incremented count field that replaces the original count field in the field descriptor in memory contains all zeros, the Overflow (O) indicator is set. If self-decrementing is specified and the decremented count field that replaces the original count field in the field descriptor in memory contains all ones, the Overflow indicator is set. If neither of the above two conditions occurs, the Overflow indicator will be reset as a result of the LOAD LOGICAL FIELD instruction.

No interrupts are processed by the CPU between these two instruction words.

Affected: (A), O, Field descriptor if self-incrementing or self-decrementing is specified.

LAF LOAD ARITHMETIC FIELD (optional)



LOAD ARITHMETIC FIELD is a two-word instruction sequence that loads the effective field into the right-hand end of the A register (general register 7). The rightmost bit of the effective field replaces bit 15 of the A register, etc. If the number of bits in the effective field is less than 16, the leftmost bit of the effective field replaces the remainder of bits in the A register (sign extension).

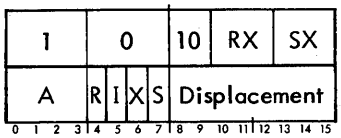
The effective address of the second instruction word is the address of the first word of a 32-bit field descriptor. The contents of the field descriptor in conjunction with the contents of the RX and SX fields of the first instruction are used to locate the effective field. The contents of the SX field may also cause the field descriptor in memory to be modified.

If self-incrementing is specified and the incremented count field that replaces the original count field in the field descriptor in memory contains all zeros, the Overflow (O) indicator is set. If self-decrementing is specified and the decremented count field that replaces the original count field in the field descriptor in memory contains all ones, the Overflow indicator is set. If neither of the above two conditions occurs, the Overflow indicator will be reset as a result of the LOAD ARITHMETIC FIELD instruction.

No interrupts are processed by the CPU between these two instruction words.

Affected: (A), O, Field descriptor if self-incrementing or self-decrementing is specified.

STF STORE FIELD (optional)



STORE FIELD is a two-word instruction sequence that replaces the effective field in memory with "n" rightmost bits of the contents of the A register (general register 7), where "n" is equal to the number of bits in the field. Bit 15 of the A register replaces the rightmost bit of the effective field, etc. The contents of the A register are unchanged by the STORE FIELD instruction. No other bits in the memory word or words containing the effective field are modified except those bits that are part of the effective field.

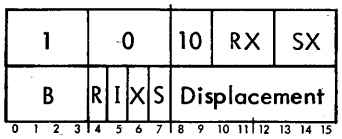
The effective address of the second instruction word is the address of the first word of a 32-bit field descriptor. The contents of the field descriptor in conjunction with the contents of the RX and SX fields of the first instruction are used to locate the effective field. The contents of the SX field may also cause the field descriptor in memory to be modified.

If self-incrementing is specified and the incremented count field that replaces the original count field in the field descriptor in memory contains all zeros, the Overflow (O) indicator is set. If self-decrementing is specified and the decremented count field that replaces the original count field in the field descriptor in memory contains all ones, the Overflow indicator is set. If neither of the above two conditions occurs, the Overflow indicator will be reset as a result of the STORE FIELD Instruction.

No interrupts are processed by the CPU between these two instruction words.

Affected: Effective field, O, Field descriptor if self-incrementing or self-decrementing is specified.

SZF STORE ZERO FIELD (optional)



STORE ZERO FIELD is a two-word instruction sequence that replaces every bit of the effective field in memory with a zero. No other bits in the memory word or words containing the effective field are modified except those bits that are part of the effective field.

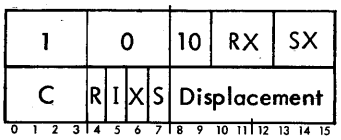
The effective address of the second instruction is the address of the first word of a 32-bit field descriptor. The contents of the field descriptor in conjunction with the contents of the RX and SX fields of the first instruction are used to locate the effective field. The contents of the SX field may also cause the field descriptor in memory to be modified.

If self-incrementing is specified and the incremented count field that replaces the original count field in the field descriptor in memory contains all zeros, the Overflow (O) indicator is set. If self-decrementing is specified and the decremented count field that replaces the original count field in the field descriptor in memory contains all ones, the Overflow indicator is set. If neither of the above two conditions occurs, the Overflow indicator will be reset as a result of the STORE ZERO FIELD instruction.

No interrupts are processed by the CPU between these two instruction words.

Affected: Effective field, O, Field descriptor if self-incrementing or self-decrementing is specified.

SOF STORE ONES FIELD (optional)



STORE ONES FIELD is a two-word instruction sequence that replaces every bit of the effective field in memory with a one. No other bits in the memory word or words containing the effective field are modified except those bits that are part of the effective field.

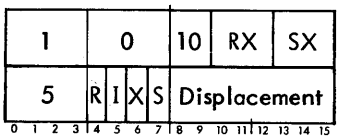
The effective address of the second instruction is the address of the first word of a 32-bit field descriptor. The contents of the field descriptor in conjunction with the contents of the RX and SX fields of the first instruction are used to locate the effective field. The contents of the SX field may also cause the field descriptor in memory to be modified.

If self-incrementing is specified and the incremented count field that replaces the original count field in the field descriptor in memory contains all zeros, the Overflow (O) indicator is set. If self-decrementing is specified and the decremented count field that replaces the original count field in the field descriptor in memory contains all ones, the Overflow indicator is set. If neither of the above two conditions occurs, the Overflow indicator will be reset as a result of the STORE ONES FIELD instruction.

No interrupts are processed by the CPU between these two instruction words.

Affected: Effective field, O, Field descriptor if self-incrementing or self-decrementing is specified.

CLF COMPARE LOGICAL FIELD (optional)



COMPARE LOGICAL FIELD is a two-word instruction sequence that forms a 16-bit word made up of the contents of the effective field in the right-hand end and zeros in all

other bit positions. COMPARE LOGICAL FIELD then algebraically compares the contents of this word containing the logical field with the contents of the A register (general register 7).

The effective address of the second instruction is the address of the first word of a 32-bit field descriptor. The contents of the field descriptor in conjunction with the contents of the RX and SX fields of the first instruction are used to locate the effective field. The contents of the SX field may also cause the field descriptor in memory to be modified.

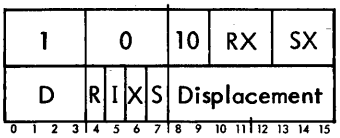
The Overflow (O) and Carry (C) indicators are set or reset according to the result of the comparison as follows:

O	C	Result of Comparison
0	0	The contents of the A register are algebraically less than the contents of the word containing the logical field.
1	0	The contents of the A register are algebraically greater than the contents of the word containing the logical field.
1	1	The contents of the A register are equal to the contents of the word containing the logical field.

No interrupts are processed by the CPU between these two instruction words.

Affected: O, C, Field descriptor if self-incrementing or self-decrementing is specified.

CAF COMPARE ARITHMETIC FIELD (optional)



COMPARE ARITHMETIC FIELD is a two-word instruction sequence that forms a 16-bit word made up of the contents of the effective field in the right-hand end and all other bit positions equal to the leftmost bit of the effective field. COMPARE ARITHMETIC FIELD then algebraically compares the contents of this word containing the sign-extended field with the contents of the A register (general register 7).

The effective address of the second instruction is the address of the first word of a 32-bit field descriptor. The contents of the field descriptor in conjunction with the contents of the RX and SX fields of the first instruction are used to locate the effective field. The contents of the SX field may also cause the field descriptor in memory to be modified.

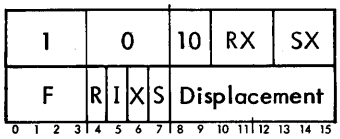
The Overflow (O) and Carry (C) indicators are set or reset according to the results of the comparison as follows:

O	C	Result of Comparison
0	0	The contents of the A register are algebraically less than the contents of the word containing the sign-extended field.
1	0	The contents of the A register are algebraically greater than the contents of the word containing the sign-extended field.
1	1	The contents of the A register are equal to the contents of the word containing the sign-extended field.

No interrupts are processed by the CPU between these two instruction words.

Affected: O, C, Field descriptor if self-incrementing or self-decrementing is specified.

SLF SENSE LEFT BIT OF FIELD (optional)



SENSE LEFT BIT OF FIELD is a two-word instruction sequence that sets or resets the Carry (C) indicator based on whether the leftmost bit of the effective field is a one or a zero. Neither the effective field nor the A register (general register 7) is modified by this instruction.

The effective address of the second instruction is the address of the first word of a 32-bit field descriptor. The contents of the field descriptor in conjunction with the contents of the RX and SX fields of the first instruction are used to locate the effective field. The contents of the SX field may also cause the field descriptor in memory to be modified.

If self-incrementing is specified and the incremented count field that replaces the original count field in the field descriptor in memory contains all zeros, the Overflow (O) indicator is set. If self-decrementing is specified and the decremented count field that replaces the original count field in the field descriptor in memory contains all ones, the Overflow indicator is set. If neither of the above conditions occurs, the Overflow indicator will be reset as a result of the SENSE LEFT BIT OF FIELD instruction.

No interrupts are processed by the CPU between these two instruction words.

Affected: O, C, Field descriptor if self-incrementing or self-decrementing is specified.

4. INPUT/OUTPUT SYSTEMS

This chapter contains information pertaining to byte-oriented input/output processors (IOPs) and to external direct input/output (DIO) systems.

IOP SYSTEMS

The computer system may have one or two input/output processors: IOP-1 (standard) with 16 channels, and IOP-2 (optional) with 12 channels. The maximum transfer rate of an IOP is 640,000 bytes per second for controllers connected to the internal interface and 504,000 bytes per second for controllers connected to the external interface using the optional two-byte interface feature.

The CPU initiates an I/O operation by selecting a device and IOP channel (see "Device Numbers") and executing an I/O instruction (see "SIO Instruction"). The IOP assumes control of the I/O operation and performs it automatically in accordance with prestored control parameters contained within a pair of channel registers (see "I/O Control Doubleword") and main memory (see "I/O Tables"). Data chaining is automatic under control of these parameters.

While the I/O operation is being performed, the CPU is free to execute instructions or perform other operations. Any IOP event that requires CPU intervention is reported either as an I/O interrupt or as a machine fault interrupt. An interrupt-servicing routine entered as a result of an I/O interrupt must contain an AIO instruction (see "I/O Instructions"). An interrupt-servicing routine entered as a result of a machine fault interrupt must contain a READ DIRECT (Mode 1) instruction (see "Fault System").

At the end of each IOP operation, the IOP automatically provides information regarding the I/O operation (see "Operational Status Byte").

Although the time-consuming details of a byte-oriented I/O operation are assumed by an IOP, the CPU retains "master" control at all times. When necessary, the CPU may determine the progress of any I/O operation or the status of any I/O device without affecting the I/O operation. The CPU may also stop any I/O operation or reset the entire I/O system (see "I/O Instructions" and "Device Status Byte").

DEVICE NUMBER

Each device within an IOP system is assigned an eight-bit device number at installation time. This number is manually selected by physical means, based on the equipment configuration for the specific installation. The device number of a given device, when loaded into bit positions 8-15 of the accumulator, is used as an I/O address by I/O instructions. It determines which I/O channel register pair is used to govern the data transmission to and from the device. Table 10 illustrates the relationship between device numbers and channel register pairs.

Table 10. IOP Channel Register and Channel-Device-Controller Numbers (Hexadecimal)

Contents of Bits 8-15		Channel Register	
Single Device Controllers[t]	Single or Multiple Device Controllers	Word Address	Byte Count
IOP-1 (First or Only IOP)			
00	8X	08	09
01	9X	0A	0B
02	AX	0C	0D
03	BX	0E	0F
04	CX[tt]	10	11
05	DX[tt]	12	13
06	EX[tt]	14	15
07	FX[tt]	16	17
08		18	19
09		1A	1B
0A		1C	1D
0B		1E	1F
0C		20	21
0D		22	23
0E		24	25
0F		26	27
IOP-2 (Second IOP)			
10	CX[tt]	28	29
11	DX[tt]	2A	2B
12	EX[tt]	2C	2D
13	FX[tt]	2E	2F
14		30	31
15		32	33
16		34	35
17		36	37
18		38	39
19		3A	3B
1A		3C	3D
1B		3E	3F

[t] All other device numbers in the range of X'00'-X'7F' are unassigned and not used.

[tt] If the system contains only IOP-1, these four groups of device numbers are assigned to IOP-1. If the system contains IOP-1 and IOP-2, these device numbers are assigned to IOP-2 only.

Device numbers are generally of two types: single-unit device numbers that are assigned to devices whose controllers permit only that device to be governed by the associated channel register pair (for example, a card reader or card

punch); or multiunit device numbers that are assigned to devices whose controller permits more than one device to be governed by the associated channel register pair (for example, magnetic tape units). The two types of device numbers are illustrated below:

Single-unit device number

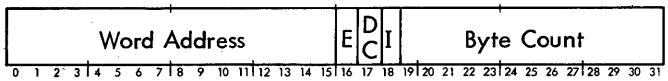
Multiunit device number

For single-unit device numbers, bits 8-10 are coded as zeros and bits 11-15 identify the device. For multiunit device numbers, bit 8 is coded as a 1, bits 9-11 specify the device controller, and bits 12-15 identify the individual device to be used with that controller.

Note: As shown in Table 10, both single-unit and multiunit device numbers are associated with the first eight channels of IOP-1 and the first four channels of IOP-2. Each of these channels may accommodate devices of either the single-unit or multiunit types, but not both. The last eight channels of each IOP may each accommodate only one device of the single-unit type.

I/O CONTROL DOUBLEWORD (IOCD)

Each I/O channel has a pair of I/O channel registers associated with it. The first IOCD for each I/O operation must be copied into the appropriate I/O channel registers by the CPU prior to initiating the I/O operation. The address of each I/O channel register is listed in columns 3 and 4 in Table 10. Subsequent IOCDs required to complete the I/O operation as a result of data chaining are fetched automatically by the IOP from the current I/O table in memory after all data transfers for that I/O table have been completed. During an I/O operation, the I/O channel registers contain the current I/O Control Doubleword, which has the following format:



The even-numbered register contains a word address that points to a memory location that is part of an I/O table (described below). The odd-numbered register contains three flag bits and a byte count. The first flag bit (E) is an error flag that is set to a 1 if a memory fault (memory parity, address parity, or nonexistent memory error – see "Fault System") is detected during a memory access for either input or output operations, or if any parity error is detected on bytes received from a device. The next two flag bits, called data chaining (DC) and interrupt (I) flags, specify action to be taken by the IOP system when the data transmission, as specified by the byte count of the current IODC, is completed. If the DC flag is 0 when the byte count is reduced to zero, the device is told (via a "count done" signal) that the I/O operation is over and that it should neither send nor receive more data but should terminate its operation. At the conclusion of an I/O operation, when all data has been transmitted and all checking associated with the data record has been performed, the device transmits an Operational Status Byte which is loaded into the even-numbered I/O channel register containing "channel end" and/or "unusual end". The device controller may generate an "unusual end" signal in place of or in conjunction with the "channel end" signal. The action caused is the same as for "channel end", except that the Operational Status Byte will contain different information. "Unusual end" may occur at any time during an I/O operation, and causes termination of all I/O operations for the device controller involved; the data chaining flag is ignored.

During normal operations, if the DC flag is set to a 1 when the byte count reaches zero, the IOP system automatically fetches the next IOCD from a doubleword location in the current I/O table and loads it into the I/O channel registers in place of the previous IOCD. Data transmission continues using the new IOCD and a new I/O table.

Note: The Operational Status Byte is loaded into the even-numbered I/O channel register only if the I/O operation has been either completed or terminated, as signaled by "channel end" or "unusual end".

If the interrupt (I) flag is set to a 1, the IOP system will instruct the device controller to generate an interrupt request if

1. The byte count reaches zero.
2. The I/O operation has been abnormally terminated by the IOP or device controller ("unusual end").
3. The I/O operation has been normally completed ("channel end").

The I/O interrupt servicing routine includes an AIO instruction to determine which device controller (with the highest priority) is interrupting and the reason for the interrupt (see "Device Interrupt").

Data chaining must be specified

1. When the amount of data required to complete the I/O operation exceeds the amount that can be specified by the current IOCD. The byte count field of the first IOCD may specify one order byte and up to 8191 data bytes. The byte count field of each subsequent IOCD relating to the same I/O operation may specify up to 8192 data bytes.

   Note: A maximum byte count is specified when the initial value of the byte count field is zero. The byte count field configuration goes from all zeros to all ones after the first byte is transferred into or out of the current I/O table.

2. When the amount of data required to complete an I/O operation cannot be contained in one I/O table because the available memory is comprised of fragments (isolated regions of unused memory locations), none large enough to contain all of the required data bytes and/or control information (see "I/O Tables").

OPERATIONAL STATUS BYTE

At the conclusion of the I/O operation, the device transmits the operational status byte to the CPU, which loads the status byte into bit positions 0-7 of the even-numbered I/O channel register associated with the device and loads zeros into the remainder of the register. (The loading of the operational status byte occurs even if channel end is signaled in the middle of an I/O table transmission.) The operational status byte contains five flags, as shown in the following diagram.

Bit	Function
0[†]	The transmission error (TE) flag is set to 1 if the device or the device controller has detected any errors during the operation. This includes such errors as parity check on magnetic tape, the parity check at the end of a RAD sector, and memory parity error on an output operation.
1[†]	The incorrect length (IL) flag indicates whether (1) or not (0) the input or output record contained the number of bytes specified by the controlling IOCD's byte count. Incorrect length may or may not be considered an error, depending on the type of operation performed. For example, during a card read operation, if a byte count of 80 is specified, then the length is correct, because only 80 bytes can be read from the card in the EBCDIC format. If, however, a count of 75 bytes is specified, the card reader will receive a count done signal before it reaches the end of the card, which causes the incorrect length flag to be set to 1. Similarly, if the reader detects the end of the card before it reaches a count done signal, the incorrect length flag is set to 1.
2[†]	The chaining modifier (CM) flag is set to 1 by some devices to indicate that a special condition has been encountered. For example, the unbuffered card punch requires the output image to be transmitted 12 times, once for each row. After the 12th row is punched, the punch controller sets the chaining modifier flag to 1 to indicate that the last transmission has been received and that no further transmissions are required for the current card. The chaining modifier may be used in different ways by other devices.
3	The channel end (CE) flag is set to 1 at the conclusion of every error free I/O operation to indicate that all data involved in the operation have been transmitted and all checking associated with the data has been performed.
4	The unusual end (UE) flag is set to 1 if the operation terminated because of some unusual condition. The unusual condition may or may not be an erroneous or faulty condition; in any event, it is not a normal termination. For example, a magnetic tape Read operation that encountered an end-of-file record instead of a data record would produce an unusual end condition. A faulty operation such as a card jam in the middle of a card-reading operation would also produce unusual end. If the UE flag is set, the state of the CE flag is not specified.
5-7	These bits are always loaded as zeros.

[†]These functions are not necessarily implemented in all peripheral device controllers. Refer to peripheral device reference manuals for more complete information.

I/O TABLES

All I/O operations are performed to and from an I/O table. The IOCD controlling the first I/O table must be loaded into the I/O channel registers by the CPU. A specific configuration of the WRITE DIRECT instruction is used to transfer the initial IOCD from the accumulator to the I/O channel registers (see "WRITE DIRECT (Mode 0)" instructions).

Each I/O table contains a data section and/or control information for an I/O operation controlled by an IOP. The type and number of I/O tables required for an operation is dependent upon the type of operation and the number of data bytes to be transferred. An I/O table (see Figure 6) may have an order byte, a data section, and a next I/O Control Doubleword. All parts of an I/O table must be contained within contiguous memory locations and may occupy any region of memory except dedicated memory locations (e.g., memory locations assigned to or reserved for interrupt levels).

The six types of I/O tables that may be formed by using the order byte, data section, and next IOCD either individually or in combination with one another are described in Table 11.

ORDER BYTE

This part of an I/O table is required only in the first I/O table for an I/O operation. The eight bits that comprise the order byte contain control information for the device controller and I/O device (see "Device Orders"). In a multi-I/O table operation, the order byte from the first I/O table prevails for the entire operation. For some control operations (e.g., stop), the order byte constitutes the I/O table. For a data transfer operation, the order byte is followed immediately by a data section. If the data section contains an odd number of bytes, the order byte occupies the first byte of the first word of the first I/O table. If the data section contains an even number of bytes, the order byte occupies the second byte of the first word of the first I/O table and the first byte of the first word is ignored.

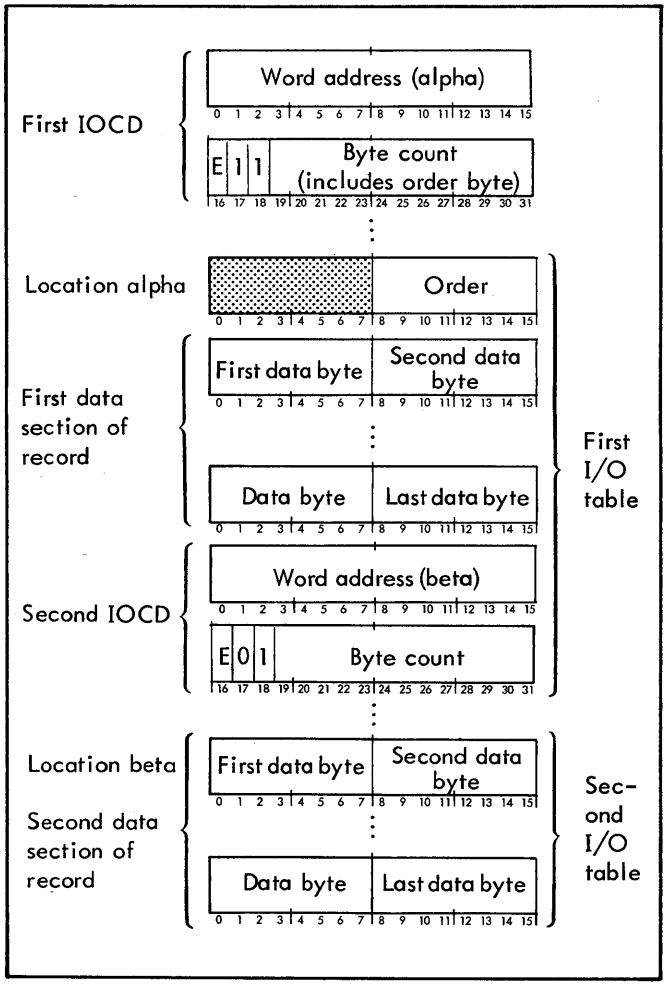


Figure 6. I/O Control Doublewords and I/O Tables

DATA SECTION

This part of an I/O table is required for all operations involving data transfers into or out of main memory. The data section will be the second part of the first I/O table and the first or only part of subsequent I/O tables relating to the same I/O operation. If additional data is required, the data section will be followed immediately by the next I/O Control Doubleword. As part of the first I/O table, the data section may contain up to 8191 data bytes. As part of subsequent I/O tables, the data section may contain up to 8192 data bytes. For output operations, the data section contains information that has been pre-stored by the CPU in preparation of a forthcoming output operation. For input operations, the data section will contain the information after the input operation has been performed. Thus, data transfers into or out of main memory via IOP operations are accomplished into or out of data sections of I/O tables.

NEXT IOCD

This part of an I/O table is required only if data chaining is specified by the current IOCD. The four bytes that comprise the next IOCD are not included in the byte count value. When present, the next IOCD is always the last part of an I/O table. The next IOCD is automatically copied by the IOP into the appropriate I/O channel registers when the byte count of the associated IOCD reaches zero. After being copied into the I/O channel registers, the next IOCD becomes the current IOCD and assumes all associated functions.

Table 11. I/O Tables

Order Byte	Data Section	Next IOCD	Description
1	0	0	I/O table is comprised of an order byte only and may be stored in any available memory location. In the associated IOCD, data chaining is not specified and the byte count value is 1. This form of I/O table is used for operations in which no data is transferred (e.g., Stop of Control orders as described under "Device Orders").
1	0	1	I/O table is comprised of an order byte and the next IOCD. In the associated IOCD, data chaining is specified and the byte count value is 1 (the four bytes containing the next IOCD are not included in the byte count). No data transfers will take place. This form of I/O table permits utilizing a five-byte memory fragment as the first I/O table of a multitable I/O operation.
1	1	0	I/O table is comprised of an order byte and a data section. In the associated IOCD, data chaining is not specified and the byte count value must reflect the order byte and the number of data bytes to be transferred. Maximum number of data bytes may be 8191. This form of I/O table is used when all data bytes for the I/O operation are contained in this I/O table. The size of the I/O table is as specified by the byte count of the associated IOCD.

Table 11. I/O Tables (cont.)

Order Byte	Data Section	Next IOCD	Description
1	1	1	I/O table is comprised of an order byte, a data section, and the next IOCD. In the associated IOCD, data chaining is specified and the byte count value must reflect the order byte and the number of data bytes to be transferred. Maximum number of data bytes is 8191. Byte count does not include the four bytes containing the next IOCD. This form of I/O table is used as the first I/O table in a multitable I/O operation. The size of the I/O table is four bytes more than specified by the byte count of the associated IOCD.
0	1	1	I/O table is comprised of a data section and the next IOCD. In the associated IOCD, data chaining is specified and the byte count value reflects only the number of data bytes to be transferred. Maximum number of data bytes may be 8192. Byte count does not include the four bytes containing the next IOCD. This form of I/O table is used as an intermediate (not the first or last) I/O table in a multitable I/O operation. The size of the I/O table is four bytes more than specified by the byte count of the associated IOCD.
0	1	0	I/O table is comprised of a data section only. In the associated IOCD, data chaining is not specified and the byte count value reflects only the number of data bytes to be transferred. Maximum number of data bytes may be 8192. This form of I/O table is used as the last I/O table of a multitable I/O operation. The size of the I/O table is as specified by the byte count of the associated IOCD.

DEVICE ORDERS

When a device is started for an I/O operation, it first requests an order from the IOP system to determine what operation is to be performed. An order byte, which has been prestored in the first word of the first I/O table in anticipation of this I/O operation, is transmitted to the device under control of the I/O channel to which the device is attached. The orders that may be accepted by a device are Write, Read, Read Backward, Control, Sense, and Stop. The code format for each order is shown below. Bit positions marked "M" specify unique modifications that depend on the device to which the order is sent.

	Bit position of device order byte							
Order	0	1	2	3	4	5	6	7
Write	M	M	M	M	M	M	0	1
Read	M	M	M	M	M	M	1	0
Read Backward	M	M	M	M	1	1	0	0
Control	M	M	M	M	M	M	1	1
Sense	M	M	M	M	0	1	0	0
Stop	I	0	0	0	0	0	0	0

The device orders operate in the following manner:

1. Write. The Write order causes the device controller to initiate an output operation. The controller makes output requests to the IOP system and data bytes are transmitted from memory, under control of the IOCD, to the device. The output operation normally continues until no further data chaining is to take place and the byte count of the last IOCD is reduced to zero. At this time, the IOP signals count done and the device generates channel end. Channel end occurs when the device has received all information associated with the output operation, has generated all checking information, and (if possible) has performed post-write checking. It is possible for some devices to generate channel end before count done is received.

2. Read. The Read order causes the device to initiate an input operation. Bytes are transmitted by the devices, then stored in memory under control of the IOCD. The input operation continues until the device generates channel end or until the byte count is reduced to zero and count done is signaled to the device. In either case, the operation is eventually terminated by a channel end signal when all checking has been performed on the input record.

3. Read Backward. The Read Backward order can be executed only by certain peripheral devices. The Read Backward order causes the device that can execute it to start operation in a backward direction and to transmit bytes; however, the record appears in memory in reverse sequence from the way it was originally written.

4. Control. The Control order is used to initiate special operations by the device. For some operations, the Control order itself may be sufficient to specify the entire operation to be performed. With magnetic tape operations, for example, the Control order initiates such operations as rewind, backspace record, backspace file, space record, etc. These orders can all be specified by the modifier (M) bits of the Control order.

If, however, the controller requires additional information for a particular operation, it is provided by the same IOCD that controls the transmission of the Control order. When all data necessary for the operation have been transmitted (and, in some cases when the operation itself is complete), the device controller signals channel end.

5. Sense. The Sense order causes the device to transmit one or more bytes of information describing its current operational status. These bytes are stored in memory under control of the IOCD. The type of status information that may be transmitted is a function of each individual device.

6. Stop. The Stop order (interpreted by some devices) causes a device to terminate its operation immediately. The I modifier bit (in position 0 of the Stop order) indicates that the device is to trigger the I/O interrupt level at the time it receives the Stop order. Bit positions 1, 2, and 3 of the Stop order are ignored.

DEVICE INTERRUPTS

All device controllers (and in the case of multiunit devices, the devices themselves) can generate a device interrupt. Each device remembers that it has generated an interrupt so that when the instruction ACKNOWLEDGE I/O INTERRUPT (AIO) is executed, the device with the highest priority identifies itself to the program. Device interrupts are generated by the device at the time of data chaining or at unusual end or channel end if the interrupt (I) flag in the controlling IOCD is set to 1. The interrupt flag is inspected by the I/O system at channel end time, unusual end time, and at data chaining time.

In addition to these normal times for interrupts, some devices can accept a Control order (or even a Read or Write order) that directs the device to interrupt after the transmission operation is completed. This type of interrupt generally occurs at channel end (that time during the operation of the device when all mechanical motion associated with a previously initiated operation has been completed). For example, a magnetic tape unit can be directed (with a Control order) to rewind and to interrupt when the rewind is complete. The order is accepted and channel end is generated immediately after the rewind operation begins. The device remembers the necessity to interrupt and, when the load point is encountered, the tape stops and channel end occurs; at this time the device generates an interrupt (and holds the interrupt-pending status until it is acknowledged). In this case, the magnetic tape control unit may be busy controlling the operation of another device for a read or write function. The pending device interrupt is a status condition that can be read by I/O instructions.

I/O INSTRUCTIONS

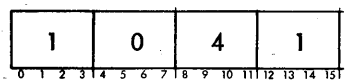
The CPU initiates and controls I/O operations using six instructions.

- Start Input/Output (SIO)
- Test Input/Output (TIO)
- Test Device (TDV)
- Halt Input/Output (HIO)
- Acknowledge I/O Interrupt (AIO)
- Reset Input/Output.

These instructions are coded as internal control functions of the READ DIRECT instruction. All instructions except AIO and Reset Input/Output require a device number in bit positions 8-15 of the accumulator when the instruction is executed.

To permit IOPs to operate at optimum data transfer rates, short program loops that repetitively execute I/O instructions should be avoided.

SIO Start Input/Output



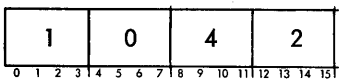
SIO is used to initiate an input or output operation with the device selected by the device number contained in bit positions 8-15 of the accumulator. If a device recognizes the number, it returns its device status byte into bit positions 0-7 of the accumulator (see "Device Status Byte"); otherwise, zeros are returned to these positions.

The Overflow and Carry indicators are set or reset, according to the result of the instruction, as follows:

O	C	Significance
0	0	I/O address recognized and SIO accepted.
0	1	I/O address recognized but SIO not accepted.
1	0	Controller "busy" with device other than one addressed and unable to send status.
1	1	I/O address not recognized.

Affected: $(A)_{0-7}$, O, C Timing: See Appendix B

TIO Test Input/Output



TIO causes the device whose device number is in bit positions 8-15 of the accumulator to make the same responses it would make to an SIO instruction, except that the device is not started nor is its state altered. If a device

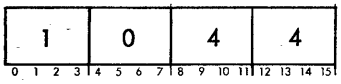
recognizes the device number, it returns its device status byte to bit positions 0-7 of the accumulator (see "Device Status Byte"); otherwise, zeros are returned to these positions.

The Overflow and Carry indicators are set or reset, according to the result of the instruction, as follows:

O	C	Significance
0	0	I/O address recognized and SIO can be accepted.
0	1	I/O address recognized but SIO cannot be accepted.
1	0	Controller "busy" with device other than one addressed and unable to send status.
1	1	I/O address not recognized.

Affected: $(A)_{0-7}$,O,C Timing: See Appendix B

TDV Test Device



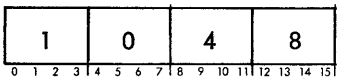
TDV is used to obtain specific information about the device whose device number is contained in bit positions 8-15 of the accumulator. The device state is not altered. If a device recognizes the device number, it returns its device status byte to bit positions 0-7 of the accumulator (see "Device Status Byte"); otherwise, zeros are returned to these positions.

The Overflow and Carry indicators are set or reset, according to the result of the instruction, as follows:

O	C	Significance
0	0	I/O address recognized.
0	1	I/O address recognized and controller is in a test mode.
1	0	Controller "busy" with device other than one addressed and unable to send status.
1	1	I/O address not recognized.

Affected: $(A)_{0-7}$,O,C Timing: See Appendix B

HIO Halt Input/Output



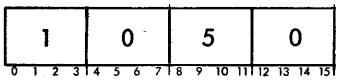
HIO causes the device whose device number is in bit positions 8-15 of the accumulator to stop its current operation immediately. The HIO instruction may cause the device to terminate improperly. In the case of magnetic tape units, for example, the device is forced to stop whether it has reached an interrecord gap or not. A pending interrupt within the device will be reset. If a device recognizes the device number, it returns its device status byte to bit positions 0-7 of the accumulator (see "Device Status Byte"); otherwise, zeros are returned to these positions.

The Overflow and Carry indicators are set or reset, according to the result of the instruction, as follows:

O	C	Significance
0	0	I/O address recognized and the device controller is not "busy".
0	1	I/O address recognized and the device controller was "busy" at the time of the halt.
1	0	HIO not accepted. Controller "busy" with device other than one addressed and unable to send status.
1	1	I/O address not recognized.

Affected: $(A)_{0-7}$, O, C Timing: See Appendix B

AIO Acknowledge I/O Interrupt



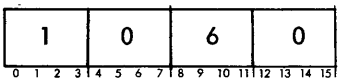
AIO is used to acknowledge an interrupt generated by an I/O device. It causes the highest-priority interrupting device to identify itself and return not only status, but also its device number. If any devices have interrupts pending, the highest-priority device clears its pending interrupt and returns its status (which is loaded into bit positions 0-7 of the accumulator) and its device number (which is loaded into bit positions 8-15) (see "Device Status Byte"); if no interrupt is recognized, zeros are returned into bit positions 0-7 of the accumulator.

The Overflow and Carry indicators are set or reset, according to the result of the instruction, as follows:

O	C	Significance
0	0	Normal interrupt recognition.
0	1	Unusual interrupt recognition or controller in test mode.
1	0	Invalid code.
1	1	No interrupt recognition.

Affected: (A),O,C Timing: See Appendix B

Reset Input/Output



Reset Input/Output causes all units connected to the internal DIO bus to be initialized (e.g., Direct Memory Adapters, External DIO, IOPs, and the interrupt system.

Note: The integrity of the I/O channels cannot be assured to be the same after executing a Reset Input/Output instruction.

DEVICE STATUS BYTE

As the result of executing an I/O instruction, if there is a device whose number corresponds to the number in the

accumulator, its Device Status Byte is loaded into bit positions 0-7 of the accumulator. (The device number in bits 8-15 is not altered.)

The AIO instruction does not require the device number, since one of its functions is to obtain the number of the device that triggered the I/O interrupt level.

The overflow and carry indicators are set to record the nature of the response to all I/O instructions. The I/O status loaded into the accumulator by the I/O instructions is summarized in Table 12.

For the instructions SIO, TIO, and HIO the status indicators have the following meaning:

Device Interrupt Pending. Bit 0 indicates whether (if it is a 1) or not (if it is a 0) the device has generated an interrupt signal that has not yet been acknowledged. A new I/O operation cannot be initiated on this device until the pending interrupt signal has been acknowledged by means of an AIO instruction.

Device Condition.[†] Bits 1 and 2 describe which of four possible conditions the device is currently in. The device conditions are

00 Device ready. The device can accept and act upon an SIO instruction if no device interrupt is pending.

01 Device not operational. A nonoperational device does not accept an SIO instruction. It requires operator intervention before any action can be taken with regard to its operation.

10 Device unavailable.

11 Device busy. The device has accepted an SIO instruction and has not yet concluded the operation.

Device Mode. Bit 3, the mode status indicator, is a 1 if the operator has cleared the device for operation and has actuated the START switch, placing the device in the "automatic" mode. If the mode status indicator is a 0, the device is in the "manual" mode and requires operator intervention before it can operate. A ready device in the "manual" mode can accept an SIO instruction even though it cannot begin to operate until it is placed in the "automatic" mode. Some devices are "permanently" in the automatic mode.

Unusual End Termination. Bit 4 is set to 1 if the previous operation on this device resulted in an unusual end; otherwise, bit 4 is reset to 0.

Device Contoller Condition. Bits 5 and 6 describe which of four possible conditions the device controller is currently in. These conditions are identical in meaning to the device conditions. The controller need not be in the same condition as the device in the case of a multiunit device controller. The device controller conditions are

00 Device controller ready. If the controller is ready and the device is ready, an SIO instruction can be accepted.

01 Device controller not operational.

10 Device controller unavailable.

11 Device controller busy. The controller and the device connected to it (or one of the devices connected to it) have accepted an SIO instruction and the I/O operation thus initiated has not terminated.

Note that in addition to the Device Status Byte in position 0-7, the instruction AIO also causes the device number to be loaded into bit positions 8-15 of the accumulator.

[†]For single-unit device controllers, bits 1-2 and 5-6 are identical. Some devices only differentiate between the "ready" and "busy" states rather than identifying four distinct states.

Table 12. Device Status Byte

Position and state in A								Significance for SIO, HIO, TIO
0	1	2	3	4	5	6	7	
1	-	-	-	-	-	-	-	device interrupt pending
-	0	0	-	-	-	-	-	device ready
-	0	1	-	-	-	-	-	device not operational
-	1	0	-	-	-	-	-	device unavailable
-	1	1	-	-	-	-	-	device busy
-	-	-	0	-	-	-	-	device manual
-	-	-	1	-	-	-	-	device automatic
-	-	-	-	1	-	-	-	device unusual end
-	-	-	-	-	0	0	-	device controller ready
-	-	-	-	-	0	1	-	device controller not operational
-	-	-	-	-	1	0	-	device controller unavailable
-	-	-	-	-	1	1	-	device controller busy
-	-	-	-	-	-	-	0	unassigned

Position and state in A								Significance for TDV, AIO
0	1	2	3	4	5	6	7	
1	-	-	-	-	-	-	-	unique to the device (see peripheral device reference manuals)
-	1	-	-	-	-	-	-	
-	-	1	-	-	-	-	-	
-	-	-	1	-	-	-	-	
-	-	-	-	1	-	-	-	
-	-	-	-	-	1	-	-	
-	-	-	-	-	-	1	-	
-	-	-	-	-	-	-	1	

EXTERNAL DIO

With the optional External DIO, the READ DIRECT and WRITE DIRECT instructions are used to communicate with special system devices. WRITE DIRECT is used to transmit a control signal, along with 16 data bits, to a device. Similarly, READ DIRECT is used to transmit a control signal and then accept 16 data bits from the external unit. Both instructions can be used to obtain a two-bit status response from the device.

When the External DIO feature is installed, the WRITE DIRECT instruction can set up the 16 control lines plus the 16 data lines; these remain stable until an acknowledgment signal is received from the device. A delay by the device in responding to WRITE DIRECT does not have any adverse effect on the operation of the byte-oriented IOP system.

The READ DIRECT instruction operates in a similar fashion. The 16 control lines are held stable and the device responds with its acknowledge signal and 16 data bits. Xerox publication 90 09 73 (Interface Design Manual) describes the External DIO in detail.

5. OPERATOR CONTROLS

PROCESSOR CONTROL PANEL

The processor control panel (PCP), illustrated in Figure 7, contains switches and indicators that permit operating and maintenance personnel to control and monitor the computer system. Each switch and/or switch position is identified with functional and/or operational information. The DATA indicators are labeled with positional information; all other indicators are backlighted and identified functionally.

PCP switches are described in Table 13. PCP indicators are described in Table 14.

BASIC OPERATING PROCEDURES

INITIALIZATION (POWER ON AND NORMAL LOAD)

1. Set all toggle switches located above DATA indicators to the down position.
2. Set the PCP MODE SELECT switch to the NORMAL position.
3. Turn the Keylock switch to the PCP ENABLED position.
4. After the power-on sequence is completed (signified by the POWER ON indicator), proceed to next step.
5. Prepare input device.
6. Set DATA switch 7 (0 = IOP-1; 1 = IOP-2).
7. Set DATA switches 8-15 to value of device number.
8. Momentarily lift RESET switch.
9. Momentarily lift LOAD switch.
10. Place RUN switch in the up position (self-locking). As a result of the above procedures, a micrologic test of the basic CPU functions will be executed and the first record will be loaded from the input device to location X'0000' through X'003F' and the WAIT indicator will be backlighted.
11. Place RUN switch in the down position and then return to the up position. The record that has been loaded, as described in the preceding steps, will now be executed.
12. The Keylock switch may be turned to the PROTECT ON position if subsequent operator interventions are required only for PCP interrupts. All switches other than DISPLAY SELECT and INTERRUPT are disabled.

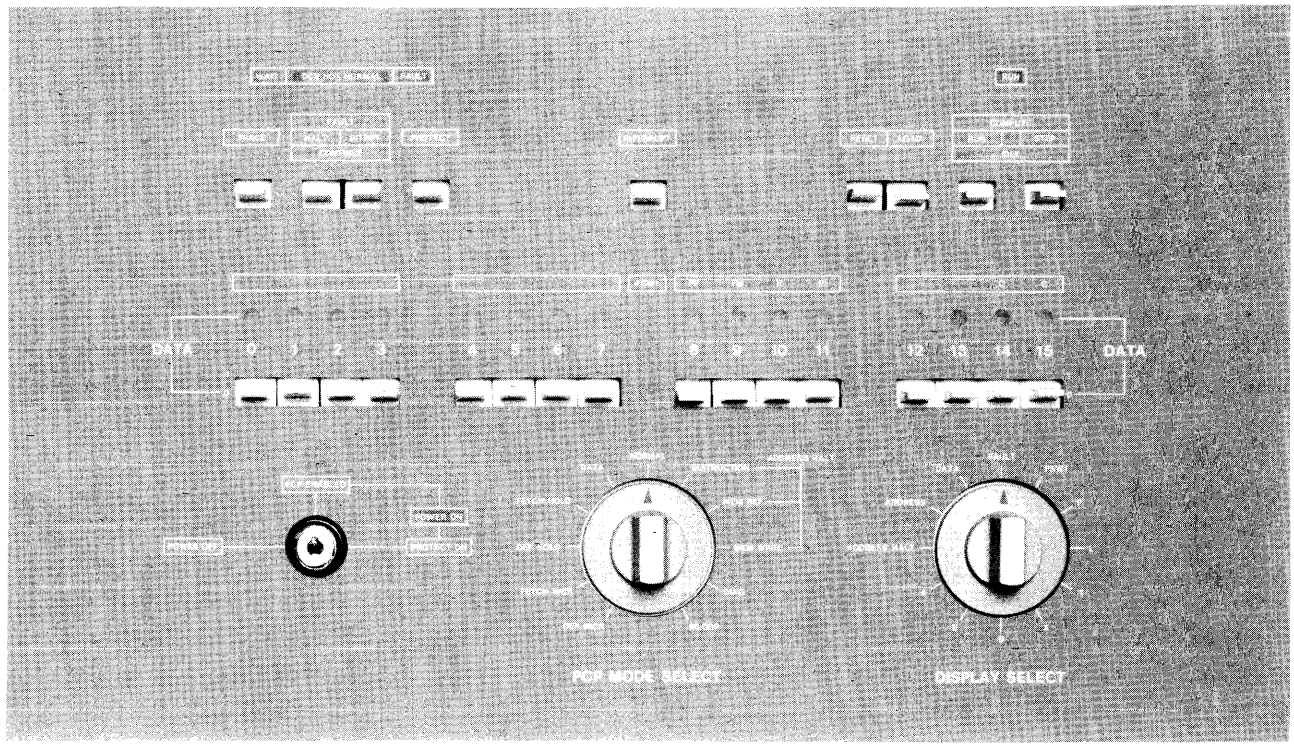


Figure 7. Processor Control Panel

Table 13. PCP Switches

Name (Type)	Position	Function
(Keylock)	POWER OFF	Removes ac power from the computer system.
	PCP ENABLED	Permits ac power to be applied to the computer system. When this switch is initially moved from the POWER OFF to the PCP ENABLED position, an automatic power-on sequence is started and the POWER ON indicator is lighted. When the keylock switch is in this position, all switches on the PCP are enabled.
	PROTECT ON	Disables all PCP switches except the INTERRUPT and the DISPLAY SELECT. This position is normally used when the CPU is executing a program that requires a minimal amount of attention and service from the computer operator.
PCP MODE SELECT (Rotary) This switch is effective only when the PCP is enabled.	NORMAL	Allows the CPU to operate in a normal manner as determined by the other control switches and programmed controls.
	INSTRUCTION ADDRESS HALT	Allows an address match to cause an Idle condition only if the access is one of the instruction itself.
	MEMORY REFERENCE ADDRESS HALT	Allows any address match to cause an Idle condition.
	MEMORY WRITE ADDRESS HALT	Allows an address match to cause an Idle condition only if a write access is attempted.
	DIAG	Modifies the load and run operation such that a predetermined machine language test routine is transferred from read-only-memory to the first 256 main memory locations and executed from main memory. The normal load and run operation does not occur.
	MLOOP	In conjunction with the run function, permits continuous execution of the CPU micrologic test.
	ENTER DATA	Enables the operator to store the contents of the DATA indicators into the register selected by the DISPLAY switch by activating the STEP or RUN switch.
	FETCH/HOLD	Allows the contents of the memory location specified by the address register to be read into the data register when the STEP or RUN switch is activated. The address register is not incremented as a result of this operation.
	DEPOSIT/HOLD	Allows the contents of the DATA switches to be stored in the location specified by the memory address register and in the data register when the RUN or STEP switch is activated. The address register is not incremented by this operation.
	FETCH/ INCREMENT	The same as FETCH/HOLD except the address register is incremented.
	DEPOSIT/ INCREMENT	The same as DEPOSIT/HOLD except the address register is incremented.

Table 13. PCP Switches (cont.)

Name (Type)	Position	Function
DISPLAY SELECT (Rotary) This switch is always effective.	PSW1	Allows the contents of the first word of the program status word to be displayed and modified. Bits 0 through 7 are always zero.
	P	Allows the contents of the P register (general register 1) to be displayed and modified.
	L	Allows the contents of the L register (general register 2) to be displayed and modified.
	T	Allows the contents of the T register (general register 3) to be displayed and modified.
	X	Allows the contents of the X register (general register 4) to be displayed and modified.
	B	Allows the contents of the B register (general register 5) to be displayed and modified.
	E	Allows the contents of the E register (general register 6) to be displayed and modified.
	A	Allows the contents of the A register (general register 7) to be displayed and modified.
	ADDRESS HALT	Allows the contents of the address halt register to be displayed and modified. The address halt register is used in conjunction with the three address halt modes as selected by the PCP MODE SELECT switch (INSTRUCTION ADDRESS HALT, MEMORY ADDRESS HALT, and MEMORY WRITE ADDRESS HALT). The contents of the address halt register must be modified appropriately by the operator prior to entering the RUN mode with the PCP MODE SELECT switch in one of the three ADDRESS HALT positions. Otherwise, the results of the operation are indeterminate. Note: The address halt register is also used to contain the top-of-memory address in memory scan operations.
	DATA	Allows the contents of the data register to be displayed. This register contains the contents of the memory location pointed to by the contents of the address register.
	ADDRESS	Allows the contents of specific memory locations to be displayed and modified.
	FAULT	Allows the contents of the fault register to be displayed only. The fault register can not be modified via the PCP.
DATA 0 through DATA 15		Each of these 16 general-purpose switches may be set to the 1 or 0 position to represent data or control information during load, register modification, memory display, memory modification, address halt, and memory scan operations.
TRACE (Locking Toggle)	up	If PCP is enabled, this switch provides a continuous trigger of the PCP interrupt level, which, if this level is armed and enabled, will cause an interrupt to occur at every interruptible point of each instruction executed (interrupt routines can not be traced). Note: If the INTERRUPT switch is raised while the TRACE switch is up, the machine fault interrupt will be triggered and the fault register will uniquely identify this "pseudo-fault" combination of events.

Table 13. PCP Switches (cont.)

Name (Type)	Position	Function
TRACE (cont.)	down	Allows INTERRUPT switch to function normally when PCP is enabled.
HALT (Locking Toggle)	up	Causes the CPU to halt if a fault condition is encountered.
	down	If the HALT and INTRPT switches are both down, any fault condition will be ignored.
INTRPT (Locking Toggle	up	Causes the CPU to enter an interrupt service routine if a fault condition is encountered (overrides HALT switch).
	down	If the HALT and INTRPT switches are both down, any fault condition will be ignored.
PROTECT (Locking Toggle)	up	Enables memory protect feature if PCP is enabled.
	down	Disables memory protect feature if PCP is enabled.
INTERRUPT (MOMENTARY Toggle)	up	A manual means for generating PCP interrupt (see also TRACE).
	down	None.
RESET (Momentary Toggle)	up	Causes the CPU system, including the I/O, to be manually reset by the operator.
	down	None.
LOAD (Momentary Toggle)	up	Permits the operator to initiate the load operation (input data from I/O device).
	down	None.
RUN (Locking Toggle)	up	If the PCP MODE SELECT switch is in the NORMAL position, the CPU enters the compute mode of operation whereby instructions are executed in a continuous manner under program control.
	down	If the RUN and STEP switches are down at the same time, the CPU is in the Idle mode.
STEP (Momentary Toggle)	up	If the PCP MODE SELECT switch is in the NORMAL position, the CPU enters the compute mode and executes a single instruction. This switch is disabled if the RUN switch is in the up position.
	down	If the STEP and RUN switches are both down, the CPU is in the Idle mode.

Table 14. PCP Indicators

Name	Significance (when lighted)
WAIT	Indicates that the CPU is in the Wait state.
DCV NOT NORMAL	Indicates that a power supply has been set to the LOW MARGIN condition.
FAULT	Indicates a fault has occurred. The type of fault may be determined by displaying the fault register.
RUN	The CPU is busy executing instructions in a continuous manner.
DATA 0 through DATA 15	When the CPU is in the Idle state, these 16 indicators display the contents of registers selected by the DISPLAY SELECT switch. When the CPU is in the Run state, these indicators will display either the contents of the CPU W register or the fault register. Each indicator is labeled with a positional notation that corresponds to the bit position of a word. DATA indicators 8, 9, 10, 11, 14, and 15 are also functionally marked (above each mentioned position) to reflect the significance of the bits when displaying the first word of the program status doubleword (PSW1).

The Keylock switch in the PROTECT ON position forces the following PCP switches into the state listed below:

RESET	DISABLED
LOAD	DISABLED
TRACE	DISABLED
COMPUTE	RUN
FAULT	INTERRUPT
PROTECT	ON
PCP MODE SELECT	NORMAL

13. If the loading operation failed,

    a. Check for obvious peripheral failures (e.g., card jam, runaway tape, etc.).

    b. If a peripheral failure is not immediately observed, set the DISPLAY SELECT switch to the FAULT position and, if the display indicates a machine fault, record the machine environment immediately for use by maintenance personnel.

REGISTER MODIFICATION

1. Set PCP to Idle state (the Keylock switch is in the PCP ENABLED position and the RUN and STEP switches are both in the down position).

2. Rotate the DISPLAY SELECT switch to select the appropriate register. The current contents of the select register is displayed in the DATA indicators.

3. Rotate the PCP MODE SELECT switch to the ENTER DATA position.

4. Set the individual DATA switches to the desired configuration.

5. Momentarily lift the STEP switch. The DATA indicators now display the same bit configuration as the DATA switches.

Note: The contents of the fault register may be selected and displayed at any time. However, the contents can not be modified by the operator via the PCP.

ENTER LOOP

If the RUN switch is used instead of the STEP switch in step 5 of "Register Modification", the enter data function is cycled continuously.

MEMORY DISPLAY

1. Set PCP to Idle mode (the Keylock switch is in the PCP ENABLED position and the RUN and STEP switches are both in the down position).

2. Set the PCP MODE SELECT switch to the ENTER DATA position.

3. Set the DATA switches to the address of the first memory location to be displayed.

4. Set the DISPLAY SELECT switch to the ADDRESS position.

5. Momentarily lift STEP switch.

6. Set the DISPLAY SELECT switch to the DATA position.

7. Set the PCP MODE SELECT switch to the FETCH/HOLD position (if a single location access is desired) or to the FETCH/INCREMENT position (if a series of sequential accesses is desired).

8. Momentarily lift the STEP switch. The DATA indicators now display the contents of the desired memory location. If the PCP MODE SELECT switch is in the FETCH/INCREMENT position, the contents of the next sequential memory location will be displayed each time the STEP switch is momentarily lifted.

MEMORY MODIFICATION (SINGLE)

1. Set PCP to Idle mode (Keylock switch is in the PCP ENABLED position, and the RUN and STEP switches are both in the down position).

2. Set PCP MODE SELECT switch to the ENTER DATA position.

3. Set address of location to be modified in DATA switches.

4. Set DISPLAY SELECT switch to ADDRESS position.

5. Mementarily lift STEP switch.

6. Set PCP MODE SELECT switch to DEPOSIT/HOLD position.

7. Set desired data configuration in DATA switches.

8. Momentarily lift STEP switch.

MEMORY MODIFICATION (MULTIPLE SEQUENTIAL LOCATIONS)

1. Set PCP to Idle mode (Keylock switch is in the PCP ENABLED position and the RUN and STEP switches are both in the down position).

2. Set PCP MODE SELECT switch to the ENTER DATA position.

3. Set DATA switches with address of first location to be modified.

4. Set DISPLAY SELECT switch to ADDRESS position.

5. Momentarily lift STEP switch.

6. Set PCP MODE SELECT switch to DEPOSIT/INCREMENT position.

7. Set DATA switches with desired data.

8. Momentarily lift STEP switch.

9. Repeat steps 7 and 8 until all memory locations are modified.

ADDRESS HALT

1. Set PCP to Idle state (Keylock switch must be in the PCP ENABLED position, the RUN switch must be down, and the STEP switch must be down).

2. Set the DISPLAY SELECT switch to the ADDRESS HALT position.

3. Set PCP MODE SELECT switch to the ENTER DATA position.

4. Set DATA switches to the desired stop address.

5. Momentarily lift the STEP switch.

6. Set the PCP MODE SELECT switch to one of the following positions:

   a. INSTRUCTION
   b. MEMORY REFERENCE
   c. MEMORY WRITE

   } ADDRESS HALT

7. Set the RUN switch in the up position. Program execution is performed until a match occurs. The RUN indicator is turned off. To continue, place the RUN switch in the down position and then return it to the up position.

   Note: The "Address Halt" mode may be used during load operations by performing this step after step 6 and before step 7 as described under "Initialization".

MEMORY SCAN

1. To cycle on a single location, follow the procedures listed under "Memory Display" or "Memory Modification", as desired, but use the RUN switch instead of the STEP switch.

2. To cycle on all or some portion of memory other than a single location, enter a top-of-memory (or top-of-test area) into the ADDRESS HALT register first; then perform the procedures listed for "Memory Display" or "Memory Modification (Sequential)" but use the RUN switch instead of the STEP switch.

APPENDIX A. REFERENCE TABLES

This appendix contains the following reference material:

Title

Standard Symbols and Codes

Standard 8-Bit Computer Codes (EBCDIC)

Standard 7-Bit Communication Codes (ANSCII)

Standard Symbol-Code Correspondences

Hexadecimal Arithmetic

- Addition Table
- Multiplication Table
- Table of Powers of Sixteen$_{10}$
- Table of Powers of Ten$_{16}$

Hexadecimal-Decimal Integer Conversion Table

Hexadecimal-Decimal Fraction Conversion Table

Table of Powers of Two

Mathematical Constants

STANDARD SYMBOLS AND CODES

The symbol and code standards described in this publication are applicable to all Xerox computer products, both hardware and software. They may be expanded or altered from time to time to meet changing requirements.

The symbols listed here include two types: graphic symbols and control characters. Graphic symbols are displayable and printable; control characters are not. Hybrids are SP, the symbol for a blank space; and DEL, the delete code, which is not considered a control command.

Three types of code are shown: (1) the 8-bit Xerox Standard Computer Code, i.e., the Extended Binary-Coded-Decimal Interchange Code (EBCDIC); (2) the 7-bit American National Standard Code for Information Interchange (ANSCII); and (3) the Xerox standard card code.

STANDARD CHARACTER SETS

1. EBCDIC

   57-character set: uppercase letters, numerals, space, and & - / . < > ( ) + | $ * : ; , % # @ ' =

   63-character set: same as above plus ¢ ! _ ? " ¬

   89-character set: same as 63-character set plus lowercase letters

2. ANSCII

   64-character set: uppercase letters, numerals, space, and ! " $ % & ' ( ) * + , - . / \ ; : = < > ? @ _ [ ] ^ #

   95-character set: same as above plus lowercase letters and { } ¦ ~ `

CONTROL CODES

In addition to the standard character sets listed above, the symbol repertoire includes 37 control codes and the hybrid code DEL (hybrid code SP is considered part of all character sets). These are listed in the table titled Standard Symbol-Code Correspondences.

SPECIAL CODE PROPERTIES

The following two properties of all standard codes will be retained for future standard code extensions:

1. All control codes, and only the control codes, have their two high-order bits equal to "00". DEL is not considered a control code.

2. No two graphic EBCDIC codes have their seven low-order bits equal.

STANDARD 8-BIT COMPUTER CODES (EBCDIC)

Least Significant Digits	Hexadecimal	Binary	Most Significant Digits 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
			0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0	0000	NUL	DLE	ds		SP	&	-									0
1	1	0001	SOH	DC1	ss				/		a	j		\ [1]	A	J		1
2	2	0010	STX	DC2	fs						b	k	s	{ [1]	B	K	S	2
3	3	0011	ETX	DC3	si						c	l	t	} [1]	C	L	T	3
4	4	0100	EOT	DC4							d	m	u	[[1]	D	M	U	4
5	5	0101	HT	LF NL			Will not be assigned				e	n	v	] [1]	E	N	V	5
6	6	0110	ACK	SYN							f	o	w		F	O	W	6
7	7	0111	BEL	ETB							g	p	x		G	P	X	7
8	8	1000	EOM BS	CAN							h	q	y		H	Q	Y	8
9	9	1001	ENQ	EM							i	r	z		I	R	Z	9
A	A	1010	NAK	SUB			¢ [2]	!	^ [1]	:								
B	B	1011	VT	ESC			.	$	,	#								
C	C	1100	FF	FS			<	*	%	@					Will not be assigned			
D	D	1101	CR	GS			(	)	_	'								
E	E	1110	SO	RS			+	;	>	=								
F	F	1111	SI	US			| [2]	¬ [2]	?	"								DEL
			3				4				5							

NOTES:

1 The characters ^ \ { } [ ] are ANSCII characters that do not appear in any of the EBCDIC-based character sets, though they are shown in the EBCDIC table.

2 The characters ¢ | ¬ appear in the 63- and 89-character EBCDIC sets but not in either of the ANSCII-based sets. However, Xerox software translates the characters c into ANSCII characters as follows:

EBCDIC	=	ANSCII
¢		` (6-0)
|		¦ (7-12)
¬		~ (7-14)

3 The EBCDIC control codes in columns 0 and 1 and their binary representation are exactly the same as those in the ANSCII table, except for two interchanges: LF/NL with NAK, and HT with ENQ.

4 Characters enclosed in heavy lines are included only in the standard 63- and 89-character EBCDIC sets.

5 These characters are included only in the standard 89-character EBCDIC set.

STANDARD 7-BIT COMMUNICATION CODES (ANSCII) [1]

Least Significant Digits	Decimal (rows) (col's.)	Binary [1]	Most Significant Digits 0	1	2	3	4	5	6	7
			x000	x001	x010	x011	x100	x101	x110	x111
0	0	0000	NUL	DLE	SP	0	@	P	`	p
1	1	0001	SOH	DC1	! [5]	1	A	Q	a	q
2	2	0010	STX	DC2	"	2	B	R	b	r
3	3	0011	ETX	DC3	#	3	C	S	c	s
4	4	0100	EOT	DC4	$	4	D	T	d	t
5	5	0101	ENQ	NAK	%	5	E	U	e	u
6	6	0110	ACK	SYN	&	6	F	V	f	v
7	7	0111	BEL	ETB	'	7	G	W	g	w
8	8	1000	BS	CAN	(	8	H	X	h	x
9	9	1001	HT	EM	)	9	I	Y	i	y
10	A	1010	LF NL	SUB	*	:	J	Z	j	z
11	B	1011	VT	ESC	+	;	K	[[5]	k	{
12	C	1100	FF	FS	,	<	L	\	l	¦
13	D	1101	CR	GS	-	=	M	] [5]	m	}
14	E	1110	SO	RS	.	>	N	[4] ^ [5]	n	~ [4]
15	F	1111	SI	US	/	?	O	_ [4]	o	DEL
			2		3					

NOTES:

1 Most significant bit, added for 8-bit format, is either 0 or even parity.

2 Columns 0-1 are control codes.

3 Columns 2-5 correspond to the 64-character ANSCII set.
Columns 2-7 correspond to the 95-character ANSCII set.

4 On many current teletypes, the symbol

^	is	↑	(5-14)
_	is	←	(5-15)
~	is	ESC or ALTMODE control	(7-14)

and none of the symbols appearing in columns 6-7 are provided. Except for the three symbol differences noted above, therefore, such teletypes provide all the characters in the 64-character ANSCII set. (The Xerox 7015 Remote Keyboard Printer provides the 64-character ANSCII set also, but prints ^ as ∧.)

5 On the Xerox 7670 Remote Batch Terminal, the symbol

!	is	|	(2-1)
[	is	¢	(5-11)
]	is	!	(5-13)
^	is	¬	(5-14)

and none of the symbols appearing in columns 6-7 are provided. Except for the four symbol differences noted above, therefore, this terminal provides all the characters in the 64-character ANSCII set.

STANDARD SYMBOL-CODE CORRESPONDENCES

EBCDIC† Hex.	EBCDIC† Dec.	Symbol	Card Code	ANSCII††	Meaning	Remarks
00	0	NUL	12-0-9-8-1	0-0	null	00 through 23 and 2F are control codes.
01	1	SOH	12-9-1	0-1	start of header	
02	2	STX	12-9-2	0-2	start of text	
03	3	ETX	12-9-3	0-3	end of text	
04	4	EOT	12-9-4	0-4	end of transmission	
05	5	HT	12-9-5	0-9	horizontal tab	
06	6	ACK	12-9-6	0-6	acknowledge (positive)	
07	7	BEL	12-9-7	0-7	bell	
08	8	BS or EOM	12-9-8	0-8	backspace or end of message	EOM is used only on Xerox Keyboard/ Printers Models 7012, 7020, 8091, and 8092.
09	9	ENQ	12-9-8-1	0-5	enquiry	
0A	10	NAK	12-9-8-2	1-5	negative acknowledge	
0B	11	VT	12-9-8-3	0-11	vertical tab	
0C	12	FF	12-9-8-4	0-12	form feed	
0D	13	CR	12-9-8-5	0-13	carriage return	
0E	14	SO	12-9-8-6	0-14	shift out	
0F	15	SI	12-9-8-7	0-15	shift in	
10	16	DLE	12-11-9-8-1	1-0	data link escape	
11	17	DC1	11-9-1	1-1	device control 1	
12	18	DC2	11-9-2	1-2	device control 2	
13	19	DC3	11-9-3	1-3	device control 3	
14	20	DC4	11-9-4	1-4	device control 4	
15	21	LF or NL	11-9-5	0-10	line feed or new line	
16	22	SYN	11-9-6	1-6	sync	
17	23	ETB	11-9-7	1-7	end of transmission block	
18	24	CAN	11-9-8	1-8	cancel	
19	25	EM	11-9-8-1	1-9	end of medium	
1A	26	SUB	11-9-8-2	1-10	substitute	Replaces characters with parity error.
1B	27	ESC	11-9-8-3	1-11	escape	
1C	28	FS	11-9-8-4	1-12	file separator	
1D	29	GS	11-9-8-5	1-13	group separator	
1E	30	RS	11-9-8-6	1-14	record separator	
1F	31	US	11-9-8-7	1-15	unit separator	
20	32	ds	11-0-9-8-1		digit selector	20 through 23 are used with Sigma EDIT BYTE STRING (EBS) instruction – not input/output control codes. 24 through 2E are unassigned.
21	33	ss	0-9-1		significance start	
22	34	fs	0-9-2		field separation	
23	35	si	0-9-3		immediate significance start	
24	36		0-9-4			
25	37		0-9-5			
26	38		0-9-6			
27	39		0-9-7			
28	40		0-9-8			
29	41		0-9-8-1			
2A	42		0-9-8-2			
2B	43		0-9-8-3			
2C	44		0-9-8-4			
2D	45		0-9-8-5			
2E	46		0-9-8-6			
2F	47		0-9-8-7			
30	48		12-11-0-9-8-1			30 through 3F are unassigned.
31	49		9-1			
32	50		9-2			
33	51		9-3			
34	52		9-4			
35	53		9-5			
36	54		9-6			
37	55		9-7			
38	56		9-8			
39	57		9-8-1			
3A	58		9-8-2			
3B	59		9-8-3			
3C	60		9-8-4			
3D	61		9-8-5			
3E	62		9-8-6			
3F	63		9-8-7			

†Hexadecimal and decimal notation.

††Decimal notation (column-row).

STANDARD SYMBOL-CODE CORRESPONDENCES (cont.)

EBCDIC† Hex.	EBCDIC† Dec.	Symbol	Card Code	ANSCII††	Meaning	Remarks
40	64	SP	blank	2-0	blank	
41	65		12-0-9-1			41 through 49 will not be assigned.
42	66		12-0-9-2			
43	67		12-0-9-3			
44	68		12-0-9-4			
45	69		12-0-9-5			
46	70		12-0-9-6			
47	71		12-0-9-7			
48	72		12-0-9-8			
49	73		12-8-1			
4A	74	¢ or ‘	12-8-2	6-0	cent or accent grave	Accent grave used for left single quote. On model 7670, ‘ not available, and ¢ = ANSCII 5-11.
4B	75	.	12-8-3	2-14	period	
4C	76	<	12-8-4	3-12	less than	
4D	77	(	12-8-5	2-8	left parenthesis	
4E	78	+	12-8-6	2-11	plus	
4F	79	| or ¦	12-8-7	7-12	vertical bar or broken bar	On Model 7670, ¦ not available, and | = ANSCII 2-1.
50	80	&	12	2-6	ampersand	
51	81		12-11-9-1			51 through 59 will not be assigned.
52	82		12-11-9-2			
53	83		12-11-9-3			
54	84		12-11-9-4			
55	85		12-11-9-5			
56	86		12-11-9-6			
57	87		12-11-9-7			
58	88		12-11-9-8			
59	89		11-8-1			
5A	90	!	11-8-2	2-1	exclamation point	On Model 7670, ! is |.
5B	91	$	11-8-3	2-4	dollars	
5C	92	*	11-8-4	2-10	asterisk	
5D	93	)	11-8-5	2-9	right parenthesis	
5E	94	;	11-8-6	3-11	semicolon	
5F	95	~ or ¬	11-8-7	7-14	tilde or logical not	On Model 7670, ~ is not available, and ¬ = ANSCII 5-14.
60	96	-	11	2-13	minus, dash, hyphen	
61	97	/	0-1	2-15	slash	
62	98		11-0-9-2			62 through 69 will not be assigned.
63	99		11-0-9-3			
64	100		11-0-9-4			
65	101		11-0-9-5			
66	102		11-0-9-6			
67	103		11-0-9-7			
68	104		11-0-9-8			
69	105		0-8-1			
6A	106	^	12-11	5-14	circumflex	On Model 7670 ^ is ¬. On Model 7015 ^ is ∧ (caret).
6B	107	,	0-8-3	2-12	comma	
6C	108	%	0-8-4	2-5	percent	
6D	109	_	0-8-5	5-15	underline	Underline is sometimes called "break character"; may be printed along bottom of character line.
6E	110	>	0-8-6	3-14	greater than	
6F	111	?	0-8-7	3-15	question mark	
70	112		12-11-0			70 through 79 will not be assigned.
71	113		12-11-0-9-1			
72	114		12-11-0-9-2			
73	115		12-11-0-9-3			
74	116		12-11-0-9-4			
75	117		12-11-0-9-5			
76	118		12-11-0-9-6			
77	119		12-11-0-9-7			
78	120		12-11-0-9-8			
79	121		8-1			
7A	122	:	8-2	3-10	colon	
7B	123	#	8-3	2-3	number	
7C	124	@	8-4	4-0	at	
7D	125	'	8-5	2-7	apostrophe (right single quote)	
7E	126	=	8-6	3-13	equals	
7F	127	"	8-7	2-2	quotation mark	

†Hexadecimal and decimal notation.

††Decimal notation (column-row).

EBCDIC† Hex.	EBCDIC† Dec.	Symbol	Card Code	ANSCII††	Meaning	Remarks
80	128		12-0-8-1			80 is unassigned.
81	129	a	12-0-1	6-1		81-89, 91-99, A2-A9 comprise the lowercase alphabet. Available only in standard 89- and 95-character sets.
82	130	b	12-0-2	6-2		
83	131	c	12-0-3	6-3		
84	132	d	12-0-4	6-4		
85	133	e	12-0-5	6-5		
86	134	f	12-0-6	6-6		
87	135	g	12-0-7	6-7		
88	136	h	12-0-8	6-8		
89	137	i	12-0-9	6-9		
8A	138		12-0-8-2			8A through 90 are unassigned.
8B	139		12-0-8-3			
8C	140		12-0-8-4			
8D	141		12-0-8-5			
8E	142		12-0-8-6			
8F	143		12-0-8-7			
90	144		12-11-8-1			
91	145	j	12-11-1	6-10		
92	146	k	12-11-2	6-11		
93	147	l	12-11-3	6-12		
94	148	m	12-11-4	6-13		
95	149	n	12-11-5	6-14		
96	150	o	12-11-6	6-15		
97	151	p	12-11-7	7-0		
98	152	q	12-11-8	7-1		
99	153	r	12-11-9	7-2		
9A	154		12-11-8-2			9A through A1 are unassigned.
9B	155		12-11-8-3			
9C	156		12-11-8-4			
9D	157		12-11-8-5			
9E	158		12-11-8-6			
9F	159		12-11-8-7			
A0	160		11-0-8-1			
A1	161		11-0-1			
A2	162	s	11-0-2	7-3		
A3	163	t	11-0-3	7-4		
A4	164	u	11-0-4	7-5		
A5	165	v	11-0-5	7-6		
A6	166	w	11-0-6	7-7		
A7	167	x	11-0-7	7-8		
A8	168	y	11-0-8	7-9		
A9	169	z	11-0-9	7-10		
AA	170		11-0-8-2			AA through B0 are unassigned.
AB	171		11-0-8-3			
AC	172		11-0-8-4			
AD	173		11-0-8-5			
AE	174		11-0-8-6			
AF	175		11-0-8-7			
B0	176		12-11-0-8-1			
B1	177	\	12-11-0-1	5-12	backslash	
B2	178	{	12-11-0-2	7-11	left brace	
B3	179	}	12-11-0-3	7-13	right brace	
B4	180	[	12-11-0-4	5-11	left bracket	On Model 7670, [is ¢.
B5	181	]	12-11-0-5	5-13	right bracket	On Model 7670,] is !.
B6	182		12-11-0-6			B6 through BF are unassigned.
B7	183		12-11-0-7			
B8	184		12-11-0-8			
B9	185		12-11-0-9			
BA	186		12-11-0-8-2			
BB	187		12-11-0-8-3			
BC	188		12-11-0-8-4			
BD	189		12-11-0-8-5			
BE	190		12-11-0-8-6			
BF	191		12-11-0-8-7			

†Hexadecimal and decimal notation.

††Decimal notation (column-row).

STANDARD SYMBOL-CODE CORRESPONDENCES (cont.)

EBCDIC[t] Hex.	EBCDIC[t] Dec.	Symbol	Card Code	ANSCII[tt]	Meaning	Remarks
C0	192		12-0			C0 is unassigned.
C1	193	A	12-1	4-1		C1-C9, D1-D9, E2-E9 comprise the uppercase alphabet.
C2	194	B	12-2	4-2		
C3	195	C	12-3	4-3		
C4	196	D	12-4	4-4		
C5	197	E	12-5	4-5		
C6	198	F	12-6	4-6		
C7	199	G	12-7	4-7		
C8	200	H	12-8	4-8		
C9	201	I	12-9	4-9		
CA	202		12-0-9-8-2			CA through CF will not be assigned.
CB	203		12-0-9-8-3			
CC	204		12-0-9-8-4			
CD	205		12-0-9-8-5			
CE	206		12-0-9-8-6			
CF	207		12-0-9-8-7			
D0	208		11-0			D0 is unassigned.
D1	209	J	11-1	4-10		
D2	210	K	11-2	4-11		
D3	211	L	11-3	4-12		
D4	212	M	11-4	4-13		
D5	213	N	11-5	4-14		
D6	214	O	11-6	4-15		
D7	215	P	11-7	5-0		
D8	216	Q	11-8	5-1		
D9	217	R	11-9	5-2		
DA	218		12-11-9-8-2			DA through DF will not be assigned.
DB	219		12-11-9-8-3			
DC	220		12-11-9-8-4			
DD	221		12-11-9-8-5			
DE	222		12-11-9-8-6			
DF	223		12-11-9-8-7			
E0	224		0-8-2			E0, E1 are unassigned.
E1	225		11-0-9-1			
E2	226	S	0-2	5-3		
E3	227	T	0-3	5-4		
E4	228	U	0-4	5-5		
E5	229	V	0-5	5-6		
E6	230	W	0-6	5-7		
E7	231	X	0-7	5-8		
E8	232	Y	0-8	5-9		
E9	233	Z	0-9	5-10		
EA	234		11-0-9-8-2			EA through EF will not be assigned.
EB	235		11-0-9-8-3			
EC	236		11-0-9-8-4			
ED	237		11-0-9-8-5			
EE	238		11-0-9-8-6			
EF	239		11-0-9-8-7			
F0	240	0	0	3-0		
F1	241	1	1	3-1		
F2	242	2	2	3-2		
F3	243	3	3	3-3		
F4	244	4	4	3-4		
F5	245	5	5	3-5		
F6	246	6	6	3-6		
F7	247	7	7	3-7		
F8	248	8	8	3-8		
F9	249	9	9	3-9		
FA	250		12-11-0-9-8-2			FA through FE will not be assigned.
FB	251		12-11-0-9-8-3			
FC	252		12-11-0-9-8-4			
FD	253		12-11-0-9-8-5			
FE	254		12-11-0-9-8-6			
FF	255	DEL	12-11-0-9-8-7		delete	Special – neither graphic nor control symbol.

[t] Hexadecimal and decimal notation.

[tt] Decimal notation (column-row).

HEXADECIMAL ARITHMETIC

ADDITION TABLE

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10
2	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11
3	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12
4	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13
5	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14
6	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15
7	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16
8	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17
9	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18
A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19
B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A
C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

MULTIPLICATION TABLE

1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E
3	06	09	0C	0F	12	15	18	1B	1E	21	24	27	2A	2D
4	08	0C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	0A	0F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	0C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	0E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

TABLE OF POWERS OF SIXTEEN $_{10}$

16^n	n	16^{-n}
1	0	0.10000 00000 00000 00000 x 10
16	1	0.62500 00000 00000 00000 x 10^{-1}
256	2	0.39062 50000 00000 00000 x 10^{-2}
4 096	3	0.24414 06250 00000 00000 x 10^{-3}
65 536	4	0.15258 78906 25000 00000 x 10^{-4}
1 048 576	5	0.95367 43164 06250 00000 x 10^{-6}
16 777 216	6	0.59604 64477 53906 25000 x 10^{-7}
268 435 456	7	0.37252 90298 46191 40625 x 10^{-8}
4 294 967 296	8	0.23283 06436 53869 62891 x 10^{-9}
68 719 476 736	9	0.14551 91522 83668 51807 x 10^{-10}
1 099 511 627 776	10	0.90949 47017 72928 23792 x 10^{-12}
17 592 186 044 416	11	0.56843 41886 08080 14870 x 10^{-13}
281 474 976 710 656	12	0.35527 13678 80050 09294 x 10^{-14}
4 503 599 627 370 496	13	0.22204 46049 25031 30808 x 10^{-15}
72 057 594 037 927 936	14	0.13877 78780 78144 56755 x 10^{-16}
1 152 921 504 606 846 976	15	0.86736 17379 88403 54721 x 10^{-18}

TABLE OF POWERS OF TEN $_{16}$

10^n	n	10^{-n}
1	0	1.0000 0000 0000 0000
A	1	0.1999 9999 9999 999A
64	2	0.28F5 C28F 5C28 F5C3 x 16^{-1}
3E8	3	0.4189 374B C6A7 EF9E x 16^{-2}
2710	4	0.68DB 8BAC 710C B296 x 16^{-3}
1 86A0	5	0.A7C5 AC47 1B47 8423 x 16^{-4}
F 4240	6	0.10C6 F7A0 B5ED 8D37 x 16^{-4}
98 9680	7	0.1AD7 F29A BCAF 4858 x 16^{-5}
5F5 E100	8	0.2AF3 1DC4 6118 73BF x 16^{-6}
3B9A CA00	9	0.44B8 2FA0 9B5A 52CC x 16^{-7}
2 540B E400	10	0.6DF3 7F67 5EF6 EADF x 16^{-8}
17 4876 E800	11	0.AFEB FF0B CB24 AAFF x 16^{-9}
E8 D4A5 1000	12	0.1197 9981 2DEA 1119 x 16^{-9}
918 4E72 A000	13	0.1C25 C268 4976 81C2 x 16^{-10}
5AF3 107A 4000	14	0.2D09 370D 4257 3604 x 16^{-11}
3 8D7E A4C6 8000	15	0.480E BE7B 9D58 566D x 16^{-12}
23 86F2 6FC1 0000	16	0.734A CA5F 6226 F0AE x 16^{-13}
163 4578 5D8A 0000	17	0.B877 AA32 36A4 B449 x 16^{-14}
DE0 B6B3 A764 0000	18	0.1272 5DD1 D243 ABA1 x 16^{-14}
8AC7 2304 89E8 0000	19	0.1D83 C94F B6D2 AC35 x 16^{-15}

HEXADECIMAL-DECIMAL INTEGER CONVERSION TABLE

The table below provides for direct conversions between hexadecimal integers in the range 0–FFF and decimal integers in the range 0–4095. For conversion of larger integers, the table values may be added to the following figures:

Hexadecimal	Decimal	Hexadecimal	Decimal
01 000	4 096	20 000	131 072
02 000	8 192	30 000	196 608
03 000	12 288	40 000	262 144
04 000	16 384	50 000	327 680
05 000	20 480	60 000	393 216
06 000	24 576	70 000	458 752
07 000	28 672	80 000	524 288
08 000	32 768	90 000	589 824
09 000	36 864	A0 000	655 360
0A 000	40 960	B0 000	720 896
0B 000	45 056	C0 000	786 432
0C 000	49 152	D0 000	851 968
0D 000	53 248	E0 000	917 504
0E 000	57 344	F0 000	983 040
0F 000	61 440	100 000	1 048 576
10 000	65 536	200 000	2 097 152
11 000	69 632	300 000	3 145 728
12 000	73 728	400 000	4 194 304
13 000	77 824	500 000	5 242 880
14 000	81 920	600 000	6 291 456
15 000	86 016	700 000	7 340 032
16 000	90 112	800 000	8 388 608
17 000	94 208	900 000	9 437 184
18 000	98 304	A00 000	10 485 760
19 000	102 400	B00 000	11 534 336
1A 000	106 496	C00 000	12 582 912
1B 000	110 592	D00 000	13 631 488
1C 000	114 688	E00 000	14 680 064
1D 000	118 784	F00 000	15 728 640
1E 000	122 880	1 000 000	16 777 216
1F 000	126 976	2 000 000	33 554 432

Hexadecimal fractions may be converted to decimal fractions as follows:

1. Express the hexadecimal fraction as an integer times 16^{-n}, where n is the number of significant hexadecimal places to the right of the hexadecimal point.

$$0.\,CA9BF3_{16} = CA9\,BF3_{16} \times 16^{-6}$$

2. Find the decimal equivalent of the hexadecimal integer

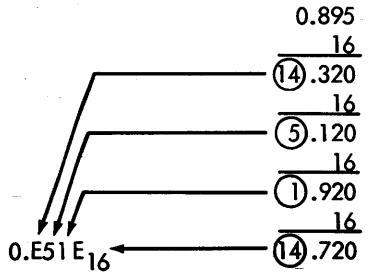
$$CA9\,BF3_{16} = 13\,278\,195_{10}$$

3. Multiply the decimal equivalent by 16^{-n}

$$\begin{array}{r} 13\,278\,195 \\ \times\ 596\,046\,448 \times 10^{-16} \\ \hline 0.791\,442\,096_{10} \end{array}$$

Decimal fractions may be converted to hexadecimal fractions by successively multiplying the decimal fraction by 16_{10}. After each multiplication, the integer portion is removed to form a hexadecimal fraction by building to the right of the hexadecimal point. However, since decimal arithmetic is used in this conversion, the integer portion of each product must be converted to hexadecimal numbers.

Example: Convert 0.895_{10} to its hexadecimal equivalent



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	0010	0011	0012	0013	0014	0015
010	0016	0017	0018	0019	0020	0021	0022	0023	0024	0025	0026	0027	0028	0029	0030	0031
020	0032	0033	0034	0035	0036	0037	0038	0039	0040	0041	0042	0043	0044	0045	0046	0047
030	0048	0049	0050	0051	0052	0053	0054	0055	0056	0057	0058	0059	0060	0061	0062	0063
040	0064	0065	0066	0067	0068	0069	0070	0071	0072	0073	0074	0075	0076	0077	0078	0079
050	0080	0081	0082	0083	0084	0085	0086	0087	0088	0089	0090	0091	0092	0093	0094	0095
060	0096	0097	0098	0099	0100	0101	0102	0103	0104	0105	0106	0107	0108	0109	0110	0111
070	0112	0113	0114	0115	0116	0117	0118	0119	0120	0121	0122	0123	0124	0125	0126	0127
080	0128	0129	0130	0131	0132	0133	0134	0135	0136	0137	0138	0139	0140	0141	0142	0143
090	0144	0145	0146	0147	0148	0149	0150	0151	0152	0153	0154	0155	0156	0157	0158	0159
0A0	0160	0161	0162	0163	0164	0165	0166	0167	0168	0169	0170	0171	0172	0173	0174	0175
0B0	0176	0177	0178	0179	0180	0181	0182	0183	0184	0185	0186	0187	0188	0189	0190	0191
0C0	0192	0193	0194	0195	0196	0197	0198	0199	0200	0201	0202	0203	0204	0205	0206	0207
0D0	0208	0209	0210	0211	0212	0213	0214	0215	0216	0217	0218	0219	0220	0221	0222	0223
0E0	0224	0225	0226	0227	0228	0229	0230	0231	0232	0233	0234	0235	0236	0237	0238	0239
0F0	0240	0241	0242	0243	0244	0245	0246	0247	0248	0249	0250	0251	0252	0253	0254	0255

HEXADECIMAL-DECIMAL INTEGER CONVERSION TABLE (cont.)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
100	0256	0257	0258	0259	0260	0261	0262	0263	0264	0265	0266	0267	0268	0269	0270	0271
110	0272	0273	0274	0275	0276	0277	0278	0279	0280	0281	0282	0283	0284	0285	0286	0287
120	0288	0289	0290	0291	0292	0293	0294	0295	0296	0297	0298	0299	0300	0301	0302	0303
130	0304	0305	0306	0307	0308	0309	0310	0311	0312	0313	0314	0315	0316	0317	0318	0319
140	0320	0321	0322	0323	0324	0325	0326	0327	0328	0329	0330	0331	0332	0333	0334	0335
150	0336	0337	0338	0339	0340	0341	0342	0343	0344	0345	0346	0347	0348	0349	0350	0351
160	0352	0353	0354	0355	0356	0357	0358	0359	0360	0361	0362	0363	0364	0365	0366	0367
170	0368	0369	0370	0371	0372	0373	0374	0375	0376	0377	0378	0379	0380	0381	0382	0383
180	0384	0385	0386	0387	0388	0389	0390	0391	0392	0393	0394	0395	0396	0397	0398	0399
190	0400	0401	0402	0403	0404	0405	0406	0407	0408	0409	0410	0411	0412	0413	0414	0415
1A0	0416	0417	0418	0419	0420	0421	0422	0423	0424	0425	0426	0427	0428	0429	0430	0431
1B0	0432	0433	0434	0435	0436	0437	0438	0439	0440	0441	0442	0443	0444	0445	0446	0447
1C0	0448	0449	0450	0451	0452	0453	0454	0455	0456	0457	0458	0459	0460	0461	0462	0463
1D0	0464	0465	0466	0467	0468	0469	0470	0471	0472	0473	0474	0475	0476	0477	0478	0479
1E0	0480	0481	0482	0483	0484	0485	0486	0487	0488	0489	0490	0491	0492	0493	0494	0495
1F0	0496	0497	0498	0499	0500	0501	0502	0503	0504	0505	0506	0507	0508	0509	0510	0511
200	0512	0513	0514	0515	0516	0517	0518	0519	0520	0521	0522	0523	0524	0525	0526	0527
210	0528	0529	0530	0531	0532	0533	0534	0535	0536	0537	0538	0539	0540	0541	0542	0543
220	0544	0545	0546	0547	0548	0549	0550	0551	0552	0553	0554	0555	0556	0557	0558	0559
230	0560	0561	0562	0563	0564	0565	0566	0567	0568	0569	0570	0571	0572	0573	0574	0575
240	0576	0577	0578	0579	0580	0581	0582	0583	0584	0585	0586	0587	0588	0589	0590	0591
250	0592	0593	0594	0595	0596	0597	0598	0599	0600	0601	0602	0603	0604	0605	0606	0607
260	0608	0609	0610	0611	0612	0613	0614	0615	0616	0617	0618	0619	0620	0621	0622	0623
270	0624	0625	0626	0627	0628	0629	0630	0631	0632	0633	0634	0635	0636	0637	0638	0639
280	0640	0641	0642	0643	0644	0645	0646	0647	0648	0649	0650	0651	0652	0653	0654	0655
290	0656	0657	0658	0659	0660	0661	0662	0663	0664	0665	0666	0667	0668	0669	0670	0671
2A0	0672	0673	0674	0675	0676	0677	0678	0679	0680	0681	0682	0683	0684	0685	0686	0687
2B0	0688	0689	0690	0691	0692	0693	0694	0695	0696	0697	0698	0699	0700	0701	0702	0703
2C0	0704	0705	0706	0707	0708	0709	0710	0711	0712	0713	0714	0715	0716	0717	0718	0719
2D0	0720	0721	0722	0723	0724	0725	0726	0727	0728	0729	0730	0731	0732	0733	0734	0735
2E0	0736	0737	0738	0739	0740	0741	0742	0743	0744	0745	0746	0747	0748	0749	0750	0751
2F0	0752	0753	0754	0755	0756	0757	0758	0759	0760	0761	0762	0763	0764	0765	0766	0767
300	0768	0769	0770	0771	0772	0773	0774	0775	0776	0777	0778	0779	0780	0781	0782	0783
310	0784	0785	0786	0787	0788	0789	0790	0791	0792	0793	0794	0795	0796	0797	0798	0799
320	0800	0801	0802	0803	0804	0805	0806	0807	0808	0809	0810	0811	0812	0813	0814	0815
330	0816	0817	0818	0819	0820	0821	0822	0823	0824	0825	0826	0827	0828	0829	0830	0831
340	0832	0833	0834	0835	0836	0837	0838	0839	0840	0841	0842	0843	0844	0845	0846	0847
350	0848	0849	0850	0851	0852	0853	0854	0855	0856	0857	0858	0859	0860	0861	0862	0863
360	0864	0865	0866	0867	0868	0869	0870	0871	0872	0873	0874	0875	0876	0877	0878	0879
370	0880	0881	0882	0883	0884	0885	0886	0887	0888	0889	0890	0891	0892	0893	0894	0895
380	0896	0897	0898	0899	0900	0901	0902	0903	0904	0905	0906	0907	0908	0909	0910	0911
390	0912	0913	0914	0915	0916	0917	0918	0919	0920	0921	0922	0923	0924	0925	0926	0927
3A0	0928	0929	0930	0931	0932	0933	0934	0935	0936	0937	0938	0939	0940	0941	0942	0943
3B0	0944	0945	0946	0947	0948	0949	0950	0951	0952	0953	0954	0955	0956	0957	0958	0959
3C0	0960	0961	0962	0963	0964	0965	0966	0967	0968	0969	0970	0971	0972	0973	0974	0975
3D0	0976	0977	0978	0979	0980	0981	0982	0983	0984	0985	0986	0987	0988	0989	0990	0991
3E0	0992	0993	0994	0995	0996	0997	0998	0999	1000	1001	1002	1003	1004	1005	1006	1007
3F0	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021	1022	1023

HEXADECIMAL-DECIMAL INTEGER CONVERSION TABLE (cont.)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
400	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035	1036	1037	1038	1039
410	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049	1050	1051	1052	1053	1054	1055
420	1056	1057	1058	1059	1060	1061	1062	1063	1064	1065	1066	1067	1068	1069	1070	1071
430	1072	1073	1074	1075	1076	1077	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087
440	1088	1089	1090	1091	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103
450	1104	1105	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119
460	1120	1121	1122	1123	1124	1125	1126	1127	1128	1129	1130	1131	1132	1133	1134	1135
470	1136	1137	1138	1139	1140	1141	1142	1143	1144	1145	1146	1147	1148	1149	1150	1151
480	1152	1153	1154	1155	1156	1157	1158	1159	1160	1161	1162	1163	1164	1165	1166	1167
490	1168	1169	1170	1171	1172	1173	1174	1175	1176	1177	1178	1179	1180	1181	1182	1183
4A0	1184	1185	1186	1187	1188	1189	1190	1191	1192	1193	1194	1195	1196	1197	1198	1199
4B0	1200	1201	1202	1203	1204	1205	1206	1207	1208	1209	1210	1211	1212	1213	1214	1215
4C0	1216	1217	1218	1219	1220	1221	1222	1223	1224	1225	1226	1227	1228	1229	1230	1231
4D0	1232	1233	1234	1235	1236	1237	1238	1239	1240	1241	1242	1243	1244	1245	1246	1247
4E0	1248	1249	1250	1251	1252	1253	1254	1255	1256	1257	1258	1259	1260	1261	1262	1263
4F0	1264	1265	1266	1267	1268	1269	1270	1271	1272	1273	1274	1275	1276	1277	1278	1279
500	1280	1281	1282	1283	1284	1285	1286	1287	1288	1289	1290	1291	1292	1293	1294	1295
510	1296	1297	1298	1299	1300	1301	1302	1303	1304	1305	1306	1307	1308	1309	1310	1311
520	1312	1313	1314	1315	1316	1317	1318	1319	1320	1321	1322	1323	1324	1325	1326	1327
530	1328	1329	1330	1331	1332	1333	1334	1335	1336	1337	1338	1339	1340	1341	1342	1343
540	1344	1345	1346	1347	1348	1349	1350	1351	1352	1353	1354	1355	1356	1357	1358	1359
550	1360	1361	1362	1363	1364	1365	1366	1367	1368	1369	1370	1371	1372	1373	1374	1375
560	1376	1377	1378	1379	1380	1381	1382	1383	1384	1385	1386	1387	1388	1389	1390	1391
570	1392	1393	1394	1395	1396	1397	1398	1399	1400	1401	1402	1403	1404	1405	1406	1407
580	1408	1409	1410	1411	1412	1413	1414	1415	1416	1417	1418	1419	1420	1421	1422	1423
590	1424	1425	1426	1427	1428	1429	1430	1431	1432	1433	1434	1435	1436	1437	1438	1439
5A0	1440	1441	1442	1443	1444	1445	1446	1447	1448	1449	1450	1451	1452	1453	1454	1455
5B0	1456	1457	1458	1459	1460	1461	1462	1463	1464	1465	1466	1467	1468	1469	1470	1471
5C0	1472	1473	1474	1475	1476	1477	1478	1479	1480	1481	1482	1483	1484	1485	1486	1487
5D0	1488	1489	1490	1491	1492	1493	1494	1495	1496	1497	1498	1499	1500	1501	1502	1503
5E0	1504	1505	1506	1507	1508	1509	1510	1511	1512	1513	1514	1515	1516	1517	1518	1519
5F0	1520	1521	1522	1523	1524	1525	1526	1527	1528	1529	1530	1531	1532	1533	1534	1535
600	1536	1537	1538	1539	1540	1541	1542	1543	1544	1545	1546	1547	1548	1549	1550	1551
610	1552	1553	1554	1555	1556	1557	1558	1559	1560	1561	1562	1563	1564	1565	1566	1567
620	1568	1569	1570	1571	1572	1573	1574	1575	1576	1577	1578	1579	1580	1581	1582	1583
630	1584	1585	1586	1587	1588	1589	1590	1591	1592	1593	1594	1595	1596	1597	1598	1599
640	1600	1601	1602	1603	1604	1605	1606	1607	1608	1609	1610	1611	1612	1613	1614	1615
650	1616	1617	1618	1619	1620	1621	1622	1623	1624	1625	1626	1627	1628	1629	1630	1631
660	1632	1633	1634	1635	1636	1637	1638	1639	1640	1641	1642	1643	1644	1645	1646	1647
670	1648	1649	1650	1651	1652	1653	1654	1655	1656	1657	1658	1659	1660	1661	1662	1663
680	1664	1665	1666	1667	1668	1669	1670	1671	1672	1673	1674	1675	1676	1677	1678	1679
690	1680	1681	1682	1683	1684	1685	1686	1687	1688	1689	1690	1691	1692	1693	1694	1695
6A0	1696	1697	1698	1699	1700	1701	1702	1703	1704	1705	1706	1707	1708	1709	1710	1711
6B0	1712	1713	1714	1715	1716	1717	1718	1719	1720	1721	1722	1723	1724	1725	1726	1727
6C0	1728	1729	1730	1731	1732	1733	1734	1735	1736	1737	1738	1739	1740	1741	1742	1743
6D0	1744	1745	1746	1747	1748	1749	1750	1751	1752	1753	1754	1755	1756	1757	1758	1759
6E0	1760	1761	1762	1763	1764	1765	1766	1767	1768	1769	1770	1771	1772	1773	1774	1775
6F0	1776	1777	1778	1779	1780	1781	1782	1783	1784	1785	1786	1787	1788	1789	1790	1791

HEXADECIMAL-DECIMAL INTEGER CONVERSION TABLE (cont.)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
700	1792	1793	1794	1795	1796	1797	1798	1799	1800	1801	1802	1803	1804	1805	1806	1807
710	1808	1809	1810	1811	1812	1813	1814	1815	1816	1817	1818	1819	1820	1821	1822	1823
720	1824	1825	1826	1827	1828	1829	1830	1831	1832	1833	1834	1835	1836	1837	1838	1839
730	1840	1841	1842	1843	1844	1845	1846	1847	1848	1849	1850	1851	1852	1853	1854	1855
740	1856	1857	1858	1859	1860	1861	1862	1863	1864	1865	1866	1867	1868	1869	1870	1871
750	1872	1873	1874	1875	1876	1877	1878	1879	1880	1881	1882	1883	1884	1885	1886	1887
760	1888	1889	1890	1891	1892	1893	1894	1895	1896	1897	1898	1899	1900	1901	1902	1903
770	1904	1905	1906	1907	1908	1909	1910	1911	1912	1913	1914	1915	1916	1917	1918	1919
780	1920	1921	1922	1923	1924	1925	1926	1927	1928	1929	1930	1931	1932	1933	1934	1935
790	1936	1937	1938	1939	1940	1941	1942	1943	1944	1945	1946	1947	1948	1949	1950	1951
7A0	1952	1953	1954	1955	1956	1957	1958	1959	1960	1961	1962	1963	1964	1965	1966	1967
7B0	1968	1969	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983
7C0	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999
7D0	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
7E0	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031
7F0	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047
800	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063
810	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079
820	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095
830	2096	2097	2098	2099	2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111
840	2112	2113	2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127
850	2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141	2142	2143
860	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155	2156	2157	2158	2159
870	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169	2170	2171	2172	2173	2174	2175
880	2176	2177	2178	2179	2180	2181	2182	2183	2184	2185	2186	2187	2188	2189	2190	2191
890	2192	2193	2194	2195	2196	2197	2198	2199	2200	2201	2202	2203	2204	2205	2206	2207
8A0	2208	2209	2210	2211	2212	2213	2214	2215	2216	2217	2218	2219	2220	2221	2222	2223
8B0	2224	2225	2226	2227	2228	2229	2230	2231	2232	2233	2234	2235	2236	2237	2238	2239
8C0	2240	2241	2242	2243	2244	2245	2246	2247	2248	2249	2250	2251	2252	2253	2254	2255
8D0	2256	2257	2258	2259	2260	2261	2262	2263	2264	2265	2266	2267	2268	2269	2270	2271
8E0	2272	2273	2274	2275	2276	2277	2278	2279	2280	2281	2282	2283	2284	2285	2286	2287
8F0	2288	2289	2290	2291	2292	2293	2294	2295	2296	2297	2298	2299	2300	2301	2302	2303
900	2304	2305	2306	2307	2308	2309	2310	2311	2312	2313	2314	2315	2316	2317	2318	2319
910	2320	2321	2322	2323	2324	2325	2326	2327	2328	2329	2330	2331	2332	2333	2334	2335
920	2336	2337	2338	2339	2340	2341	2342	2343	2344	2345	2346	2347	2348	2349	2350	2351
930	2352	2353	2354	2355	2356	2357	2358	2359	2360	2361	2362	2363	2364	2365	2366	2367
940	2368	2369	2370	2371	2372	2373	2374	2375	2376	2377	2378	2379	2380	2381	2382	2383
950	2384	2385	2386	2387	2388	2389	2390	2391	2392	2393	2394	2395	2396	2397	2398	2399
960	2400	2401	2402	2403	2404	2405	2406	2407	2408	2409	2410	2411	2412	2413	2414	2415
970	2416	2417	2418	2419	2420	2421	2422	2423	2424	2425	2426	2427	2428	2429	2430	2431
980	2432	2433	2434	2435	2436	2437	2438	2439	2440	2441	2442	2443	2444	2445	2446	2447
990	2448	2449	2450	2451	2452	2453	2454	2455	2456	2457	2458	2459	2460	2461	2462	2463
9A0	2464	2465	2466	2467	2468	2469	2470	2471	2472	2473	2474	2475	2476	2477	2478	2479
9B0	2480	2481	2482	2483	2484	2485	2486	2487	2488	2489	2490	2491	2492	2493	2494	2495
9C0	2496	2497	2498	2499	2500	2501	2502	2503	2504	2505	2506	2507	2508	2509	2510	2511
9D0	2512	2513	2514	2515	2516	2517	2518	2519	2520	2521	2522	2523	2524	2525	2526	2527
9E0	2528	2529	2530	2531	2532	2533	2534	2535	2536	2537	2538	2539	2540	2541	2542	2543
9F0	2544	2545	2546	2547	2548	2549	2550	2551	2552	2553	2554	2555	2556	2557	2558	2559

HEXADECIMAL-DECIMAL INTEGER CONVERSION TABLE (cont.)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A00	2560	2561	2562	2563	2564	2565	2566	2567	2568	2569	2570	2571	2572	2573	2574	2575
A10	2576	2577	2578	2579	2580	2581	2582	2583	2584	2585	2586	2587	2588	2589	2590	2591
A20	2592	2593	2594	2595	2596	2597	2598	2599	2600	2601	2602	2603	2604	2605	2606	2607
A30	2608	2609	2610	2611	2612	2613	2614	2615	2616	2617	2618	2619	2620	2621	2622	2623
A40	2624	2625	2626	2627	2628	2629	2630	2631	2632	2633	2634	2635	2636	2637	2638	2639
A50	2640	2641	2642	2643	2644	2645	2646	2647	2648	2649	2650	2651	2652	2653	2654	2655
A60	2656	2657	2658	2659	2660	2661	2662	2663	2664	2665	2666	2667	2668	2669	2670	2671
A70	2672	2673	2674	2675	2676	2677	2678	2679	2680	2681	2682	2683	2684	2685	2686	2687
A80	2688	2689	2690	2691	2692	2693	2694	2695	2696	2697	2698	2699	2700	2701	2702	2703
A90	2704	2705	2706	2707	2708	2709	2710	2711	2712	2713	2714	2715	2716	2717	2718	2719
AA0	2720	2721	2722	2723	2724	2725	2726	2727	2728	2729	2730	2731	2732	2733	2734	2735
AB0	2736	2737	2738	2739	2740	2741	2742	2743	2744	2745	2746	2747	2748	2749	2750	2751
AC0	2752	2753	2754	2755	2756	2757	2758	2759	2760	2761	2762	2763	2764	2765	2766	2767
AD0	2768	2769	2770	2771	2772	2773	2774	2775	2776	2777	2778	2779	2780	2781	2782	2783
AE0	2784	2785	2786	2787	2788	2789	2790	2791	2792	2793	2794	2795	2796	2797	2798	2799
AF0	2800	2801	2802	2803	2804	2805	2806	2807	2808	2809	2810	2811	2812	2813	2814	2815
B00	2816	2817	2818	2819	2820	2821	2822	2823	2824	2825	2826	2827	2828	2829	2830	2831
B10	2832	2833	2834	2835	2836	2837	2838	2839	2840	2841	2842	2843	2844	2845	2846	2847
B20	2848	2849	2850	2851	2852	2853	2854	2855	2856	2857	2858	2859	2860	2861	2862	2863
B30	2864	2865	2866	2867	2868	2869	2870	2871	2872	2873	2874	2875	2876	2877	2878	2879
B40	2880	2881	2882	2883	2884	2885	2886	2887	2888	2889	2890	2891	2892	2893	2894	2895
B50	2896	2897	2898	2899	2900	2901	2902	2903	2904	2905	2906	2907	2908	2909	2910	2911
B60	2912	2913	2914	2915	2916	2917	2918	2919	2920	2921	2922	2923	2924	2925	2926	2927
B70	2928	2929	2930	2931	2932	2933	2934	2935	2936	2937	2938	2939	2940	2941	2942	2943
B80	2944	2945	2946	2947	2948	2949	2950	2951	2952	2953	2954	2955	2956	2957	2958	2959
B90	2960	2961	2962	2963	2964	2965	2966	2967	2968	2969	2970	2971	2972	2973	2974	2975
BA0	2976	2977	2978	2979	2980	2981	2982	2983	2984	2985	2986	2987	2988	2989	2990	2991
BB0	2992	2993	2994	2995	2996	2997	2998	2999	3000	3001	3002	3003	3004	3005	3006	3007
BC0	3008	3009	3010	3011	3012	3013	3014	3015	3016	3017	3018	3019	3020	3021	3022	3023
BD0	3024	3025	3026	3027	3028	3029	3030	3031	3032	3033	3034	3035	3036	3037	3038	3039
BE0	3040	3041	3042	3043	3044	3045	3046	3047	3048	3049	3050	3051	3052	3053	3054	3055
BF0	3056	3057	3058	3059	3060	3061	3062	3063	3064	3065	3066	3067	3068	3069	3070	3071
C00	3072	3073	3074	3075	3076	3077	3078	3079	3080	3081	3082	3083	3084	3085	3086	3087
C10	3088	3089	3090	3091	3092	3093	3094	3095	3096	3097	3098	3099	3100	3101	3102	3103
C20	3104	3105	3106	3107	3108	3109	3110	3111	3112	3113	3114	3115	3116	3117	3118	3119
C30	3120	3121	3122	3123	3124	3125	3126	3127	3128	3129	3130	3131	3132	3133	3134	3135
C40	3136	3137	3138	3139	3140	3141	3142	3143	3144	3145	3146	3147	3148	3149	3150	3151
C50	3152	3153	3154	3155	3156	3157	3158	3159	3160	3161	3162	3163	3164	3165	3166	3167
C60	3168	3169	3170	3171	3172	3173	3174	3175	3176	3177	3178	3179	3180	3181	3182	3183
C70	3184	3185	3186	3187	3188	3189	3190	3191	3192	3193	3194	3195	3196	3197	3198	3199
C80	3200	3201	3202	3203	3204	3205	3206	3207	3208	3209	3210	3211	3212	3213	3214	3215
C90	3216	3217	3218	3219	3220	3221	3222	3223	3224	3225	3226	3227	3228	3229	3230	3231
CA0	3232	3233	3234	3235	3236	3237	3238	3239	3240	3241	3242	3243	3244	3245	3246	3247
CB0	3248	3249	3250	3251	3252	3253	3254	3255	3256	3257	3258	3259	3260	3261	3262	3263
CC0	3264	3265	3266	3267	3268	3269	3270	3271	3272	3273	3274	3275	3276	3277	3278	3279
CD0	3280	3281	3282	3283	3284	3285	3286	3287	3288	3289	3290	3291	3292	3293	3294	3295
CE0	3296	3297	3298	3299	3300	3301	3302	3303	3304	3305	3306	3307	3308	3309	3310	3311
CF0	3312	3313	3314	3315	3316	3317	3318	3319	3320	3321	3322	3323	3324	3325	3326	3327

HEXADECIMAL-DECIMAL INTEGER CONVERSION TABLE (cont.)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
D00	3328	3329	3330	3331	3332	3333	3334	3335	3336	3337	3338	3339	3340	3341	3342	3343
D10	3344	3345	3346	3347	3348	3349	3350	3351	3352	3353	3354	3355	3356	3357	3358	3359
D20	3360	3361	3362	3363	3364	3365	3366	3367	3368	3369	3370	3371	3372	3373	3374	3375
D30	3376	3377	3378	3379	3380	3381	3382	3383	3384	3385	3386	3387	3388	3389	3390	3391
D40	3392	3393	3394	3395	3396	3397	3398	3399	3400	3401	3402	3403	3404	3405	3406	3407
D50	3408	3409	3410	3411	3412	3413	3414	3415	3416	3417	3418	3419	3420	3421	3422	3423
D60	3424	3425	3426	3427	3428	3429	3430	3431	3432	3433	3434	3435	3436	3437	3438	3439
D70	3440	3441	3442	3443	3444	3445	3446	3447	3448	3449	3450	3451	3452	3453	3454	3455
D80	3456	3457	3458	3459	3460	3461	3462	3463	3464	3465	3466	3467	3468	3469	3470	3471
D90	3472	3473	3474	3475	3476	3477	3478	3479	3480	3481	3482	3483	3484	3485	3486	3487
DA0	3488	3489	3490	3491	3492	3493	3494	3495	3496	3497	3498	3499	3500	3501	3502	3503
DB0	3504	3505	3506	3507	3508	3509	3510	3511	3512	3513	3514	3515	3516	3517	3518	3519
DC0	3520	3521	3522	3523	3524	3525	3526	3527	3528	3529	3530	3531	3532	3533	3534	3535
DD0	3536	3537	3538	3539	3540	3541	3542	3543	3544	3545	3546	3547	3548	3549	3550	3551
DE0	3552	3553	3554	3555	3556	3557	3558	3559	3560	3561	3562	3563	3564	3565	3566	3567
DF0	3568	3569	3570	3571	3572	3573	3574	3575	3576	3577	3578	3579	3580	3581	3582	3583
E00	3584	3585	3586	3587	3588	3589	3590	3591	3592	3593	3594	3595	3596	3597	3598	3599
E10	3600	3601	3602	3603	3604	3605	3606	3607	3608	3609	3610	3611	3612	3613	3614	3615
E20	3616	3617	3618	3619	3620	3621	3622	3623	3624	3625	3626	3627	3628	3629	3630	3631
E30	3632	3633	3634	3635	3636	3637	3638	3639	3640	3641	3642	3643	3644	3645	3646	3647
E40	3648	3649	3650	3651	3652	3653	3654	3655	3656	3657	3658	3659	3660	3661	3662	3663
E50	3664	3665	3666	3667	3668	3669	3670	3671	3672	3673	3674	3675	3676	3677	3678	3679
E60	3680	3681	3682	3683	3684	3685	3686	3687	3688	3689	3690	3691	3692	3693	3694	3695
E70	3696	3697	3698	3699	3700	3701	3702	3703	3704	3705	3706	3707	3708	3709	3710	3711
E80	3712	3713	3714	3715	3716	3717	3718	3719	3720	3721	3722	3723	3724	3725	3726	3727
E90	3728	3729	3730	3731	3732	3733	3734	3735	3736	3737	3738	3739	3740	3741	3742	3743
EA0	3744	3745	3746	3747	3748	3749	3750	3751	3752	3753	3754	3755	3756	3757	3758	3759
EB0	3760	3761	3762	3763	3764	3765	3766	3767	3768	3769	3770	3771	3772	3773	3774	3775
EC0	3776	3777	3778	3779	3780	3781	3782	3783	3784	3785	3786	3787	3788	3789	3790	3791
ED0	3792	3793	3794	3795	3796	3797	3798	3799	3800	3801	3802	3803	3804	3805	3806	3807
EE0	3808	3809	3810	3811	3812	3813	3814	3815	3816	3817	3818	3819	3820	3821	3822	3823
EF0	3824	3825	3826	3827	3828	3829	3830	3831	3832	3833	3834	3835	3836	3837	3838	3839
F00	3840	3841	3842	3843	3844	3845	3846	3847	3848	3849	3850	3851	3852	3853	3854	3855
F10	3856	3857	3858	3859	3860	3861	3862	3863	3864	3865	3866	3867	3868	3869	3870	3871
F20	3872	3873	3874	3875	3876	3877	3878	3879	3880	3881	3882	3883	3884	3885	3886	3887
F30	3888	3889	3890	3891	3892	3893	3894	3895	3896	3897	3898	3899	3900	3901	3902	3903
F40	3904	3905	3906	3907	3908	3909	3910	3911	3912	3913	3914	3915	3916	3917	3918	3919
F50	3920	3921	3922	3923	3924	3925	3926	3927	3928	3929	3930	3931	3932	3933	3934	3935
F60	3936	3937	3938	3939	3940	3941	3942	3943	3944	3945	3946	3947	3948	3949	3950	3951
F70	3952	3953	3954	3955	3956	3957	3958	3959	3960	3961	3962	3963	3964	3965	3966	3967
F80	3968	3969	3970	3971	3972	3973	3974	3975	3976	3977	3978	3979	3980	3981	3982	3983
F90	3984	3985	3986	3987	3988	3989	3990	3991	3992	3993	3994	3995	3996	3997	3998	3999
FA0	4000	4001	4002	4003	4004	4005	4006	4007	4008	4009	4010	4011	4012	4013	4014	4015
FB0	4016	4017	4018	4019	4020	4021	4022	4023	4024	4025	4026	4027	4028	4029	4030	4031
FC0	4032	4033	4034	4035	4036	4037	4038	4039	4040	4041	4042	4043	4044	4045	4046	4047
FD0	4048	4049	4050	4051	4052	4053	4054	4055	4056	4057	4058	4059	4060	4061	4062	4063
FE0	4064	4065	4066	4067	4068	4069	4070	4071	4072	4073	4074	4075	4076	4077	4078	4079
FF0	4080	4081	4082	4083	4084	4085	4086	4087	4088	4089	4090	4091	4092	4093	4094	4095

HEXADECIMAL-DECIMAL FRACTION CONVERSION TABLE

Hexadecimal	Decimal	Hexadecimal	Decimal	Hexadecimal	Decimal	Hexadecimal	Decimal
.00 00 00 00	.00000 00000	.40 00 00 00	.25000 00000	.80 00 00 00	.50000 00000	.C0 00 00 00	.75000 00000
.01 00 00 00	.00390 62500	.41 00 00 00	.25390 62500	.81 00 00 00	.50390 62500	.C1 00 00 00	.75390 62500
.02 00 00 00	.00781 25000	.42 00 00 00	.25781 25000	.82 00 00 00	.50781 25000	.C2 00 00 00	.75781 25000
.03 00 00 00	.01171 87500	.43 00 00 00	.26171 87500	.83 00 00 00	.51171 87500	.C3 00 00 00	.76171 87500
.04 00 00 00	.01562 50000	.44 00 00 00	.26562 50000	.84 00 00 00	.51562 50000	.C4 00 00 00	.76562 50000
.05 00 00 00	.01953 12500	.45 00 00 00	.26953 12500	.85 00 00 00	.51953 12500	.C5 00 00 00	.76953 12500
.06 00 00 00	.02343 75000	.46 00 00 00	.27343 75000	.86 00 00 00	.52343 75000	.C6 00 00 00	.77343 75000
.07 00 00 00	.02734 37500	.47 00 00 00	.27734 37500	.87 00 00 00	.52734 37500	.C7 00 00 00	.77734 37500
.08 00 00 00	.03125 00000	.48 00 00 00	.28125 00000	.88 00 00 00	.53125 00000	.C8 00 00 00	.78125 00000
.09 00 00 00	.03515 62500	.49 00 00 00	.28515 62500	.89 00 00 00	.53515 62500	.C9 00 00 00	.78515 62500
.0A 00 00 00	.03906 25000	.4A 00 00 00	.28906 25000	.8A 00 00 00	.53906 25000	.CA 00 00 00	.78906 25000
.0B 00 00 00	.04296 87500	.4B 00 00 00	.29296 87500	.8B 00 00 00	.54296 87500	.CB 00 00 00	.79296 87500
.0C 00 00 00	.04687 50000	.4C 00 00 00	.29687 50000	.8C 00 00 00	.54687 50000	.CC 00 00 00	.79687 50000
.0D 00 00 00	.05078 12500	.4D 00 00 00	.30078 12500	.8D 00 00 00	.55078 12500	.CD 00 00 00	.80078 12500
.0E 00 00 00	.05468 75000	.4E 00 00 00	.30468 75000	.8E 00 00 00	.55468 75000	.CE 00 00 00	.80468 75000
.0F 00 00 00	.05859 37500	.4F 00 00 00	.30859 37500	.8F 00 00 00	.55859 37500	.CF 00 00 00	.80859 37500
.10 00 00 00	.06250 00000	.50 00 00 00	.31250 00000	.90 00 00 00	.56250 00000	.D0 00 00 00	.81250 00000
.11 00 00 00	.06640 62500	.51 00 00 00	.31640 62500	.91 00 00 00	.56640 62500	.D1 00 00 00	.81640 62500
.12 00 00 00	.07031 25000	.52 00 00 00	.32031 25000	.92 00 00 00	.57031 25000	.D2 00 00 00	.82031 25000
.13 00 00 00	.07421 87500	.53 00 00 00	.32421 87500	.93 00 00 00	.57421 87500	.D3 00 00 00	.82421 87500
.14 00 00 00	.07812 50000	.54 00 00 00	.32812 50000	.94 00 00 00	.57812 50000	.D4 00 00 00	.82812 50000
.15 00 00 00	.08203 12500	.55 00 00 00	.33203 12500	.95 00 00 00	.58203 12500	.D5 00 00 00	.83203 12500
.16 00 00 00	.08593 75000	.56 00 00 00	.33593 75000	.96 00 00 00	.58593 75000	.D6 00 00 00	.83593 75000
.17 00 00 00	.08984 37500	.57 00 00 00	.33984 37500	.97 00 00 00	.58984 37500	.D7 00 00 00	.83984 37500
.18 00 00 00	.09375 00000	.58 00 00 00	.34375 00000	.98 00 00 00	.59375 00000	.D8 00 00 00	.84375 00000
.19 00 00 00	.09765 62500	.59 00 00 00	.34765 62500	.99 00 00 00	.59765 62500	.D9 00 00 00	.84765 62500
.1A 00 00 00	.10156 25000	.5A 00 00 00	.35156 25000	.9A 00 00 00	.60156 25000	.DA 00 00 00	.85156 25000
.1B 00 00 00	.10546 87500	.5B 00 00 00	.35546 87500	.9B 00 00 00	.60546 87500	.DB 00 00 00	.85546 87500
.1C 00 00 00	.10937 50000	.5C 00 00 00	.35937 50000	.9C 00 00 00	.60937 50000	.DC 00 00 00	.85937 50000
.1D 00 00 00	.11328 12500	.5D 00 00 00	.36328 12500	.9D 00 00 00	.61328 12500	.DD 00 00 00	.86328 12500
.1E 00 00 00	.11718 75000	.5E 00 00 00	.36718 75000	.9E 00 00 00	.61718 75000	.DE 00 00 00	.86718 75000
.1F 00 00 00	.12109 37500	.5F 00 00 00	.37109 37500	.9F 00 00 00	.62109 37500	.DF 00 00 00	.87109 37500
.20 00 00 00	.12500 00000	.60 00 00 00	.37500 00000	.A0 00 00 00	.62500 00000	.E0 00 00 00	.87500 00000
.21 00 00 00	.12890 62500	.61 00 00 00	.37890 62500	.A1 00 00 00	.62890 62500	.E1 00 00 00	.87890 62500
.22 00 00 00	.13281 25000	.62 00 00 00	.38281 25000	.A2 00 00 00	.63281 25000	.E2 00 00 00	.88281 25000
.23 00 00 00	.13671 87500	.63 00 00 00	.38671 87500	.A3 00 00 00	.63671 87500	.E3 00 00 00	.88671 87500
.24 00 00 00	.14062 50000	.64 00 00 00	.39062 50000	.A4 00 00 00	.64062 50000	.E4 00 00 00	.89062 50000
.25 00 00 00	.14453 12500	.65 00 00 00	.39453 12500	.A5 00 00 00	.64453 12500	.E5 00 00 00	.89453 12500
.26 00 00 00	.14843 75000	.66 00 00 00	.39843 75000	.A6 00 00 00	.64843 75000	.E6 00 00 00	.89843 75000
.27 00 00 00	.15234 37500	.67 00 00 00	.40234 37500	.A7 00 00 00	.65234 37500	.E7 00 00 00	.90234 37500
.28 00 00 00	.15625 00000	.68 00 00 00	.40625 00000	.A8 00 00 00	.65625 00000	.E8 00 00 00	.90625 00000
.29 00 00 00	.16015 62500	.69 00 00 00	.41015 62500	.A9 00 00 00	.66015 62500	.E9 00 00 00	.91015 62500
.2A 00 00 00	.16406 25000	.6A 00 00 00	.41406 25000	.AA 00 00 00	.66406 25000	.EA 00 00 00	.91406 25000
.2B 00 00 00	.16796 87500	.6B 00 00 00	.41796 87500	.AB 00 00 00	.66796 87500	.EB 00 00 00	.91796 87500
.2C 00 00 00	.17187 50000	.6C 00 00 00	.42187 50000	.AC 00 00 00	.67187 50000	.EC 00 00 00	.92187 50000
.2D 00 00 00	.17578 12500	.6D 00 00 00	.42578 12500	.AD 00 00 00	.67578 12500	.ED 00 00 00	.92578 12500
.2E 00 00 00	.17968 75000	.6E 00 00 00	.42968 75000	.AE 00 00 00	.67968 75000	.EE 00 00 00	.92968 75000
.2F 00 00 00	.18359 37500	.6F 00 00 00	.43359 37500	.AF 00 00 00	.68359 37500	.EF 00 00 00	.93359 37500
.30 00 00 00	.18750 00000	.70 00 00 00	.43750 00000	.B0 00 00 00	.68750 00000	.F0 00 00 00	.93750 00000
.31 00 00 00	.19140 62500	.71 00 00 00	.44140 62500	.B1 00 00 00	.69140 62500	.F1 00 00 00	.94140 62500
.32 00 00 00	.19531 25000	.72 00 00 00	.44531 25000	.B2 00 00 00	.69531 25000	.F2 00 00 00	.94531 25000
.33 00 00 00	.19921 87500	.73 00 00 00	.44921 87500	.B3 00 00 00	.69921 87500	.F3 00 00 00	.94921 87500
.34 00 00 00	.20312 50000	.74 00 00 00	.45312 50000	.B4 00 00 00	.70312 50000	.F4 00 00 00	.95312 50000
.35 00 00 00	.20703 12500	.75 00 00 00	.45703 12500	.B5 00 00 00	.70703 12500	.F5 00 00 00	.95703 12500
.36 00 00 00	.21093 75000	.76 00 00 00	.46093 75000	.B6 00 00 00	.71093 75000	.F6 00 00 00	.96093 75000
.37 00 00 00	.21484 37500	.77 00 00 00	.46484 37500	.B7 00 00 00	.71484 37500	.F7 00 00 00	.96484 37500
.38 00 00 00	.21875 00000	.78 00 00 00	.46875 00000	.B8 00 00 00	.71875 00000	.F8 00 00 00	.96875 00000
.39 00 00 00	.22265 62500	.79 00 00 00	.47265 62500	.B9 00 00 00	.72265 62500	.F9 00 00 00	.97265 62500
.3A 00 00 00	.22656 25000	.7A 00 00 00	.47656 25000	.BA 00 00 00	.72656 25000	.FA 00 00 00	.97656 25000
.3B 00 00 00	.23046 87500	.7B 00 00 00	.48046 87500	.BB 00 00 00	.73046 87500	.FB 00 00 00	.98046 87500
.3C 00 00 00	.23437 50000	.7C 00 00 00	.48437 50000	.BC 00 00 00	.73437 50000	.FC 00 00 00	.98437 50000
.3D 00 00 00	.23828 12500	.7D 00 00 00	.48828 12500	.BD 00 00 00	.73828 12500	.FD 00 00 00	.98828 12500
.3E 00 00 00	.24218 75000	.7E 00 00 00	.49218 75000	.BE 00 00 00	.74218 75000	.FE 00 00 00	.99218 75000
.3F 00 00 00	.24609 37500	.7F 00 00 00	.49609 37500	.BF 00 00 00	.74609 37500	.FF 00 00 00	.99609 37500

HEXADECIMAL-DECIMAL FRACTION CONVERSION TABLE (cont.)

Hexadecimal	Decimal	Hexadecimal	Decimal	Hexadecimal	Decimal	Hexadecimal	Decimal
.00 00 00 00	.00000 00000	.00 40 00 00	.00097 65625	.00 80 00 00	.00195 31250	.00 C0 00 00	.00292 96875
.00 01 00 00	.00001 52587	.00 41 00 00	.00099 18212	.00 81 00 00	.00196 83837	.00 C1 00 00	.00294 49462
.00 02 00 00	.00003 05175	.00 42 00 00	.00100 70800	.00 82 00 00	.00198 36425	.00 C2 00 00	.00296 02050
.00 03 00 00	.00004 57763	.00 43 00 00	.00102 23388	.00 83 00 00	.00199 89013	.00 C3 00 00	.00297 54638
.00 04 00 00	.00006 10351	.00 44 00 00	.00103 75976	.00 84 00 00	.00201 41601	.00 C4 00 00	.00299 07226
.00 05 00 00	.00007 62939	.00 45 00 00	.00105 28564	.00 85 00 00	.00202 94189	.00 C5 00 00	.00300 59814
.00 06 00 00	.00009 15527	.00 46 00 00	.00106 81152	.00 86 00 00	.00204 46777	.00 C6 00 00	.00302 12402
.00 07 00 00	.00010 68115	.00 47 00 00	.00108 33740	.00 87 00 00	.00205 99365	.00 C7 00 00	.00303 64990
.00 08 00 00	.00012 20703	.00 48 00 00	.00109 86328	.00 88 00 00	.00207 51953	.00 C8 00 00	.00305 17578
.00 09 00 00	.00013 73291	.00 49 00 00	.00111 38916	.00 89 00 00	.00209 04541	.00 C9 00 00	.00306 70166
.00 0A 00 00	.00015 25878	.00 4A 00 00	.00112 91503	.00 8A 00 00	.00210 57128	.00 CA 00 00	.00308 22753
.00 0B 00 00	.00016 78466	.00 4B 00 00	.00114 44091	.00 8B 00 00	.00212 09716	.00 CB 00 00	.00309 75341
.00 0C 00 00	.00018 31054	.00 4C 00 00	.00115 96679	.00 8C 00 00	.00213 62304	.00 CC 00 00	.00311 27929
.00 0D 00 00	.00019 83642	.00 4D 00 00	.00117 49267	.00 8D 00 00	.00215 14892	.00 CD 00 00	.00312 80517
.00 0E 00 00	.00021 36230	.00 4E 00 00	.00119 01855	.00 8E 00 00	.00216 67480	.00 CE 00 00	.00314 33105
.00 0F 00 00	.00022 88818	.00 4F 00 00	.00120 54443	.00 8F 00 00	.00218 20068	.00 CF 00 00	.00315 85693
.00 10 00 00	.00024 41406	.00 50 00 00	.00122 07031	.00 90 00 00	.00219 72656	.00 D0 00 00	.00317 38281
.00 11 00 00	.00025 93994	.00 51 00 00	.00123 59619	.00 91 00 00	.00221 25244	.00 D1 00 00	.00318 90869
.00 12 00 00	.00027 46582	.00 52 00 00	.00125 12207	.00 92 00 00	.00222 77832	.00 D2 00 00	.00320 43457
.00 13 00 00	.00028 99169	.00 53 00 00	.00126 64794	.00 93 00 00	.00224 30419	.00 D3 00 00	.00321 96044
.00 14 00 00	.00030 51757	.00 54 00 00	.00128 17382	.00 94 00 00	.00225 83007	.00 D4 00 00	.00323 48632
.00 15 00 00	.00032 04345	.00 55 00 00	.00129 69970	.00 95 00 00	.00227 35595	.00 D5 00 00	.00325 01220
.00 16 00 00	.00033 56933	.00 56 00 00	.00131 22558	.00 96 00 00	.00228 88183	.00 D6 00 00	.00326 53808
.00 17 00 00	.00035 09521	.00 57 00 00	.00132 75146	.00 97 00 00	.00230 40771	.00 D7 00 00	.00328 06396
.00 18 00 00	.00036 62109	.00 58 00 00	.00134 27734	.00 98 00 00	.00231 93359	.00 D8 00 00	.00329 58984
.00 19 00 00	.00038 14697	.00 59 00 00	.00135 80322	.00 99 00 00	.00233 45947	.00 D9 00 00	.00331 11572
.00 1A 00 00	.00039 67285	.00 5A 00 00	.00137 32910	.00 9A 00 00	.00234 98535	.00 DA 00 00	.00332 64160
.00 1B 00 00	.00041 19873	.00 5B 00 00	.00138 85498	.00 9B 00 00	.00236 51123	.00 DB 00 00	.00334 16748
.00 1C 00 00	.00042 72460	.00 5C 00 00	.00140 38085	.00 9C 00 00	.00238 03710	.00 DC 00 00	.00335 69335
.00 1D 00 00	.00044 25048	.00 5D 00 00	.00141 90673	.00 9D 00 00	.00239 56298	.00 DD 00 00	.00337 21923
.00 1E 00 00	.00045 77636	.00 5E 00 00	.00143 43261	.00 9E 00 00	.00241 08886	.00 DE 00 00	.00338 74511
.00 1F 00 00	.00047 30224	.00 5F 00 00	.00144 95849	.00 9F 00 00	.00242 61474	.00 DF 00 00	.00340 27099
.00 20 00 00	.00048 82812	.00 60 00 00	.00146 48437	.00 A0 00 00	.00244 14062	.00 E0 00 00	.00341 79687
.00 21 00 00	.00050 35400	.00 61 00 00	.00148 01025	.00 A1 00 00	.00245 66650	.00 E1 00 00	.00343 32275
.00 22 00 00	.00051 87988	.00 62 00 00	.00149 53613	.00 A2 00 00	.00247 19238	.00 E2 00 00	.00344 84863
.00 23 00 00	.00053 40576	.00 63 00 00	.00151 06201	.00 A3 00 00	.00248 71826	.00 E3 00 00	.00346 37451
.00 24 00 00	.00054 93164	.00 64 00 00	.00152 58789	.00 A4 00 00	.00250 24414	.00 E4 00 00	.00347 90039
.00 25 00 00	.00056 45751	.00 65 00 00	.00154 11376	.00 A5 00 00	.00251 77001	.00 E5 00 00	.00349 42626
.00 26 00 00	.00057 98339	.00 66 00 00	.00155 63964	.00 A6 00 00	.00253 29589	.00 E6 00 00	.00350 95214
.00 27 00 00	.00059 50927	.00 67 00 00	.00157 16552	.00 A7 00 00	.00254 82177	.00 E7 00 00	.00352 47802
.00 28 00 00	.00061 03515	.00 68 00 00	.00158 69140	.00 A8 00 00	.00256 34765	.00 E8 00 00	.00354 00390
.00 29 00 00	.00062 56103	.00 69 00 00	.00160 21728	.00 A9 00 00	.00257 87353	.00 E9 00 00	.00355 52978
.00 2A 00 00	.00064 08691	.00 6A 00 00	.00161 74316	.00 AA 00 00	.00259 39941	.00 EA 00 00	.00357 05566
.00 2B 00 00	.00065 61279	.00 6B 00 00	.00163 26904	.00 AB 00 00	.00260 92529	.00 EB 00 00	.00358 58154
.00 2C 00 00	.00067 13867	.00 6C 00 00	.00164 79492	.00 AC 00 00	.00262 45117	.00 EC 00 00	.00360 10742
.00 2D 00 00	.00068 66455	.00 6D 00 00	.00166 32080	.00 AD 00 00	.00263 97705	.00 ED 00 00	.00361 63330
.00 2E 00 00	.00070 19042	.00 6E 00 00	.00167 84667	.00 AE 00 00	.00265 50292	.00 EE 00 00	.00363 15917
.00 2F 00 00	.00071 71630	.00 6F 00 00	.00169 37255	.00 AF 00 00	.00267 02880	.00 EF 00 00	.00364 68505
.00 30 00 00	.00073 24218	.00 70 00 00	.00170 89843	.00 B0 00 00	.00268 55468	.00 F0 00 00	.00366 21093
.00 31 00 00	.00074 76806	.00 71 00 00	.00172 42431	.00 B1 00 00	.00270 08056	.00 F1 00 00	.00367 73681
.00 32 00 00	.00076 29394	.00 72 00 00	.00173 95019	.00 B2 00 00	.00271 60644	.00 F2 00 00	.00369 26269
.00 33 00 00	.00077 81982	.00 73 00 00	.00175 47607	.00 B3 00 00	.00273 13232	.00 F3 00 00	.00370 78857
.00 34 00 00	.00079 34570	.00 74 00 00	.00177 00195	.00 B4 00 00	.00274 65820	.00 F4 00 00	.00372 31445
.00 35 00 00	.00080 87158	.00 75 00 00	.00178 52783	.00 B5 00 00	.00276 18408	.00 F5 00 00	.00373 84033
.00 36 00 00	.00082 39746	.00 76 00 00	.00180 05371	.00 B6 00 00	.00277 70996	.00 F6 00 00	.00375 36621
.00 37 00 00	.00083 92333	.00 77 00 00	.00181 57958	.00 B7 00 00	.00279 23583	.00 F7 00 00	.00376 89208
.00 38 00 00	.00085 44921	.00 78 00 00	.00183 10546	.00 B8 00 00	.00280 76171	.00 F8 00 00	.00378 41796
.00 39 00 00	.00086 97509	.00 79 00 00	.00184 63134	.00 B9 00 00	.00282 28759	.00 F9 00 00	.00379 94384
.00 3A 00 00	.00088 50097	.00 7A 00 00	.00186 15722	.00 BA 00 00	.00283 81347	.00 FA 00 00	.00381 46972
.00 3B 00 00	.00090 02685	.00 7B 00 00	.00187 68310	.00 BB 00 00	.00285 33935	.00 FB 00 00	.00382 99560
.00 3C 00 00	.00091 55273	.00 7C 00 00	.00189 20898	.00 BC 00 00	.00286 86523	.00 FC 00 00	.00384 52148
.00 3D 00 00	.00093 07861	.00 7D 00 00	.00190 73486	.00 BD 00 00	.00288 39111	.00 FD 00 00	.00386 04736
.00 3E 00 00	.00094 60449	.00 7E 00 00	.00192 26074	.00 BE 00 00	.00289 91699	.00 FE 00 00	.00387 57324
.00 3F 00 00	.00096 13037	.00 7F 00 00	.00193 78662	.00 BF 00 00	.00291 44287	.00 FF 00 00	.00389 09912

HEXADECIMAL-DECIMAL FRACTION CONVERSION TABLE (cont.)

Hexadecimal	Decimal	Hexadecimal	Decimal	Hexadecimal	Decimal	Hexadecimal	Decimal
.00 00 00 00	.00000 00000	.00 00 40 00	.00000 38146	.00 00 80 00	.00000 76293	.00 00 C0 00	.00001 14440
.00 00 01 00	.00000 00596	.00 00 41 00	.00000 38743	.00 00 81 00	.00000 76889	.00 00 C1 00	.00001 15036
.00 00 02 00	.00000 01192	.00 00 42 00	.00000 39339	.00 00 82 00	.00000 77486	.00 00 C2 00	.00001 15633
.00 00 03 00	.00000 01788	.00 00 43 00	.00000 39935	.00 00 83 00	.00000 78082	.00 00 C3 00	.00001 16229
.00 00 04 00	.00000 02384	.00 00 44 00	.00000 40531	.00 00 84 00	.00000 78678	.00 00 C4 00	.00001 16825
.00 00 05 00	.00000 02980	.00 00 45 00	.00000 41127	.00 00 85 00	.00000 79274	.00 00 C5 00	.00001 17421
.00 00 06 00	.00000 03576	.00 00 46 00	.00000 41723	.00 00 86 00	.00000 79870	.00 00 C6 00	.00001 18017
.00 00 07 00	.00000 04172	.00 00 47 00	.00000 42319	.00 00 87 00	.00000 80466	.00 00 C7 00	.00001 18613
.00 00 08 00	.00000 04768	.00 00 48 00	.00000 42915	.00 00 88 00	.00000 81062	.00 00 C8 00	.00001 19209
.00 00 09 00	.00000 05364	.00 00 49 00	.00000 43511	.00 00 89 00	.00000 81658	.00 00 C9 00	.00001 19805
.00 00 0A 00	.00000 05960	.00 00 4A 00	.00000 44107	.00 00 8A 00	.00000 82254	.00 00 CA 00	.00001 20401
.00 00 0B 00	.00000 06556	.00 00 4B 00	.00000 44703	.00 00 8B 00	.00000 82850	.00 00 CB 00	.00001 20997
.00 00 0C 00	.00000 07152	.00 00 4C 00	.00000 45299	.00 00 8C 00	.00000 83446	.00 00 CC 00	.00001 21593
.00 00 0D 00	.00000 07748	.00 00 4D 00	.00000 45895	.00 00 8D 00	.00000 84042	.00 00 CD 00	.00001 22189
.00 00 0E 00	.00000 08344	.00 00 4E 00	.00000 46491	.00 00 8E 00	.00000 84638	.00 00 CE 00	.00001 22785
.00 00 0F 00	.00000 08940	.00 00 4F 00	.00000 47087	.00 00 8F 00	.00000 85234	.00 00 CF 00	.00001 23381
.00 00 10 00	.00000 09536	.00 00 50 00	.00000 47683	.00 00 90 00	.00000 85830	.00 00 D0 00	.00001 23977
.00 00 11 00	.00000 10132	.00 00 51 00	.00000 48279	.00 00 91 00	.00000 86426	.00 00 D1 00	.00001 24573
.00 00 12 00	.00000 10728	.00 00 52 00	.00000 48875	.00 00 92 00	.00000 87022	.00 00 D2 00	.00001 25169
.00 00 13 00	.00000 11324	.00 00 53 00	.00000 49471	.00 00 93 00	.00000 87618	.00 00 D3 00	.00001 25765
.00 00 14 00	.00000 11920	.00 00 54 00	.00000 50067	.00 00 94 00	.00000 88214	.00 00 D4 00	.00001 26361
.00 00 15 00	.00000 12516	.00 00 55 00	.00000 50663	.00 00 95 00	.00000 88810	.00 00 D5 00	.00001 26957
.00 00 16 00	.00000 13113	.00 00 56 00	.00000 51259	.00 00 96 00	.00000 89406	.00 00 D6 00	.00001 27553
.00 00 17 00	.00000 13709	.00 00 57 00	.00000 51856	.00 00 97 00	.00000 90003	.00 00 D7 00	.00001 28149
.00 00 18 00	.00000 14305	.00 00 58 00	.00000 52452	.00 00 98 00	.00000 90599	.00 00 D8 00	.00001 28746
.00 00 19 00	.00000 14901	.00 00 59 00	.00000 53048	.00 00 99 00	.00000 91195	.00 00 D9 00	.00001 29342
.00 00 1A 00	.00000 15497	.00 00 5A 00	.00000 53644	.00 00 9A 00	.00000 91791	.00 00 DA 00	.00001 29938
.00 00 1B 00	.00000 16093	.00 00 5B 00	.00000 54240	.00 00 9B 00	.00000 92387	.00 00 DB 00	.00001 30534
.00 00 1C 00	.00000 16689	.00 00 5C 00	.00000 54836	.00 00 9C 00	.00000 92983	.00 00 DC 00	.00001 31130
.00 00 1D 00	.00000 17285	.00 00 5D 00	.00000 55432	.00 00 9D 00	.00000 93579	.00 00 DD 00	.00001 31726
.00 00 1E 00	.00000 17881	.00 00 5E 00	.00000 56028	.00 00 9E 00	.00000 94175	.00 00 DE 00	.00001 32322
.00 00 1F 00	.00000 18477	.00 00 5F 00	.00000 56624	.00 00 9F 00	.00000 94771	.00 00 DF 00	.00001 32918
.00 00 20 00	.00000 19073	.00 00 60 00	.00000 57220	.00 00 A0 00	.00000 95367	.00 00 E0 00	.00001 33514
.00 00 21 00	.00000 19669	.00 00 61 00	.00000 57816	.00 00 A1 00	.00000 95963	.00 00 E1 00	.00001 34110
.00 00 22 00	.00000 20265	.00 00 62 00	.00000 58412	.00 00 A2 00	.00000 96559	.00 00 E2 00	.00001 34706
.00 00 23 00	.00000 20861	.00 00 63 00	.00000 59008	.00 00 A3 00	.00000 97155	.00 00 E3 00	.00001 35302
.00 00 24 00	.00000 21457	.00 00 64 00	.00000 59604	.00 00 A4 00	.00000 97751	.00 00 E4 00	.00001 35898
.00 00 25 00	.00000 22053	.00 00 65 00	.00000 60200	.00 00 A5 00	.00000 98347	.00 00 E5 00	.00001 36494
.00 00 26 00	.00000 22649	.00 00 66 00	.00000 60796	.00 00 A6 00	.00000 98943	.00 00 E6 00	.00001 37090
.00 00 27 00	.00000 23245	.00 00 67 00	.00000 61392	.00 00 A7 00	.00000 99539	.00 00 E7 00	.00001 37686
.00 00 28 00	.00000 23841	.00 00 68 00	.00000 61988	.00 00 A8 00	.00001 00135	.00 00 E8 00	.00001 38282
.00 00 29 00	.00000 24437	.00 00 69 00	.00000 62584	.00 00 A9 00	.00001 00731	.00 00 E9 00	.00001 38878
.00 00 2A 00	.00000 25033	.00 00 6A 00	.00000 63180	.00 00 AA 00	.00001 01327	.00 00 EA 00	.00001 39474
.00 00 2B 00	.00000 25629	.00 00 6B 00	.00000 63776	.00 00 AB 00	.00001 01923	.00 00 EB 00	.00001 40070
.00 00 2C 00	.00000 26226	.00 00 6C 00	.00000 64373	.00 00 AC 00	.00001 02519	.00 00 EC 00	.00001 40666
.00 00 2D 00	.00000 26822	.00 00 6D 00	.00000 64969	.00 00 AD 00	.00001 03116	.00 00 ED 00	.00001 41263
.00 00 2E 00	.00000 27418	.00 00 6E 00	.00000 65565	.00 00 AE 00	.00001 03712	.00 00 EE 00	.00001 41859
.00 00 2F 00	.00000 28014	.00 00 6F 00	.00000 66161	.00 00 AF 00	.00001 04308	.00 00 EF 00	.00001 42455
.00 00 30 00	.00000 28610	.00 00 70 00	.00000 66757	.00 00 B0 00	.00001 04904	.00 00 F0 00	.00001 43051
.00 00 31 00	.00000 29206	.00 00 71 00	.00000 67353	.00 00 B1 00	.00001 05500	.00 00 F1 00	.00001 43647
.00 00 32 00	.00000 29802	.00 00 72 00	.00000 67949	.00 00 B2 00	.00001 06096	.00 00 F2 00	.00001 44243
.00 00 33 00	.00000 30398	.00 00 73 00	.00000 68545	.00 00 B3 00	.00001 06692	.00 00 F3 00	.00001 44839
.00 00 34 00	.00000 30994	.00 00 74 00	.00000 69141	.00 00 B4 00	.00001 07288	.00 00 F4 00	.00001 45435
.00 00 35 00	.00000 31590	.00 00 75 00	.00000 69737	.00 00 B5 00	.00001 07884	.00 00 F5 00	.00001 46031
.00 00 36 00	.00000 32186	.00 00 76 00	.00000 70333	.00 00 B6 00	.00001 08480	.00 00 F6 00	.00001 46627
.00 00 37 00	.00000 32782	.00 00 77 00	.00000 70929	.00 00 B7 00	.00001 09076	.00 00 F7 00	.00001 47223
.00 00 38 00	.00000 33378	.00 00 78 00	.00000 71525	.00 00 B8 00	.00001 09672	.00 00 F8 00	.00001 47819
.00 00 39 00	.00000 33974	.00 00 79 00	.00000 72121	.00 00 B9 00	.00001 10268	.00 00 F9 00	.00001 48415
.00 00 3A 00	.00000 34570	.00 00 7A 00	.00000 72717	.00 00 BA 00	.00001 10864	.00 00 FA 00	.00001 49011
.00 00 3B 00	.00000 35166	.00 00 7B 00	.00000 73313	.00 00 BB 00	.00001 11460	.00 00 FB 00	.00001 49607
.00 00 3C 00	.00000 35762	.00 00 7C 00	.00000 73909	.00 00 BC 00	.00001 12056	.00 00 FC 00	.00001 50203
.00 00 3D 00	.00000 36358	.00 00 7D 00	.00000 74505	.00 00 BD 00	.00001 12652	.00 00 FD 00	.00001 50799
.00 00 3E 00	.00000 36954	.00 00 7E 00	.00000 75101	.00 00 BE 00	.00001 13248	.00 00 FE 00	.00001 51395
.00 00 3F 00	.00000 37550	.00 00 7F 00	.00000 75697	.00 00 BF 00	.00001 13844	.00 00 FF 00	.00001 51991

HEXADECIMAL-DECIMAL FRACTION CONVERSION TABLE (cont.)

Hexadecimal	Decimal	Hexadecimal	Decimal	Hexadecimal	Decimal	Hexadecimal	Decimal
.00 00 00 00	.00000 00000	.00 00 00 40	.00000 00149	.00 00 00 80	.00000 00298	.00 00 00 C0	.00000 00447
.00 00 00 01	.00000 00002	.00 00 00 41	.00000 00151	.00 00 00 81	.00000 00300	.00 00 00 C1	.00000 00449
.00 00 00 02	.00000 00004	.00 00 00 42	.00000 00153	.00 00 00 82	.00000 00302	.00 00 00 C2	.00000 00451
.00 00 00 03	.00000 00006	.00 00 00 43	.00000 00155	.00 00 00 83	.00000 00305	.00 00 00 C3	.00000 00454
.00 00 00 04	.00000 00009	.00 00 00 44	.00000 00158	.00 00 00 84	.00000 00307	.00 00 00 C4	.00000 00456
.00 00 00 05	.00000 00011	.00 00 00 45	.00000 00160	.00 00 00 85	.00000 00309	.00 00 00 C5	.00000 00458
.00 00 00 06	.00000 00013	.00 00 00 46	.00000 00162	.00 00 00 86	.00000 00311	.00 00 00 C6	.00000 00461
.00 00 00 07	.00000 00016	.00 00 00 47	.00000 00165	.00 00 00 87	.00000 00314	.00 00 00 C7	.00000 00463
.00 00 00 08	.00000 00018	.00 00 00 48	.00000 00167	.00 00 00 88	.00000 00316	.00 00 00 C8	.00000 00465
.00 00 00 09	.00000 00020	.00 00 00 49	.00000 00169	.00 00 00 89	.00000 00318	.00 00 00 C9	.00000 00467
.00 00 00 0A	.00000 00023	.00 00 00 4A	.00000 00172	.00 00 00 8A	.00000 00321	.00 00 00 CA	.00000 00470
.00 00 00 0B	.00000 00025	.00 00 00 4B	.00000 00174	.00 00 00 8B	.00000 00323	.00 00 00 CB	.00000 00472
.00 00 00 0C	.00000 00027	.00 00 00 4C	.00000 00176	.00 00 00 8C	.00000 00325	.00 00 00 CC	.00000 00474
.00 00 00 0D	.00000 00030	.00 00 00 4D	.00000 00179	.00 00 00 8D	.00000 00328	.00 00 00 CD	.00000 00477
.00 00 00 0E	.00000 00032	.00 00 00 4E	.00000 00181	.00 00 00 8E	.00000 00330	.00 00 00 CE	.00000 00479
.00 00 00 0F	.00000 00034	.00 00 00 4F	.00000 00183	.00 00 00 8F	.00000 00332	.00 00 00 CF	.00000 00481
.00 00 00 10	.00000 00037	.00 00 00 50	.00000 00186	.00 00 00 90	.00000 00335	.00 00 00 D0	.00000 00484
.00 00 00 11	.00000 00039	.00 00 00 51	.00000 00188	.00 00 00 91	.00000 00337	.00 00 00 D1	.00000 00486
.00 00 00 12	.00000 00041	.00 00 00 52	.00000 00190	.00 00 00 92	.00000 00339	.00 00 00 D2	.00000 00488
.00 00 00 13	.00000 00044	.00 00 00 53	.00000 00193	.00 00 00 93	.00000 00342	.00 00 00 D3	.00000 00491
.00 00 00 14	.00000 00046	.00 00 00 54	.00000 00195	.00 00 00 94	.00000 00344	.00 00 00 D4	.00000 00493
.00 00 00 15	.00000 00048	.00 00 00 55	.00000 00197	.00 00 00 95	.00000 00346	.00 00 00 D5	.00000 00495
.00 00 00 16	.00000 00051	.00 00 00 56	.00000 00200	.00 00 00 96	.00000 00349	.00 00 00 D6	.00000 00498
.00 00 00 17	.00000 00053	.00 00 00 57	.00000 00202	.00 00 00 97	.00000 00351	.00 00 00 D7	.00000 00500
.00 00 00 18	.00000 00055	.00 00 00 58	.00000 00204	.00 00 00 98	.00000 00353	.00 00 00 D8	.00000 00502
.00 00 00 19	.00000 00058	.00 00 00 59	.00000 00207	.00 00 00 99	.00000 00356	.00 00 00 D9	.00000 00505
.00 00 00 1A	.00000 00060	.00 00 00 5A	.00000 00209	.00 00 00 9A	.00000 00358	.00 00 00 DA	.00000 00507
.00 00 00 1B	.00000 00062	.00 00 00 5B	.00000 00211	.00 00 00 9B	.00000 00360	.00 00 00 DB	.00000 00509
.00 00 00 1C	.00000 00065	.00 00 00 5C	.00000 00214	.00 00 00 9C	.00000 00363	.00 00 00 DC	.00000 00512
.00 00 00 1D	.00000 00067	.00 00 00 5D	.00000 00216	.00 00 00 9D	.00000 00365	.00 00 00 DD	.00000 00514
.00 00 00 1E	.00000 00069	.00 00 00 5E	.00000 00218	.00 00 00 9E	.00000 00367	.00 00 00 DE	.00000 00516
.00 00 00 1F	.00000 00072	.00 00 00 5F	.00000 00221	.00 00 00 9F	.00000 00370	.00 00 00 DF	.00000 00519
.00 00 00 20	.00000 00074	.00 00 00 60	.00000 00223	.00 00 00 A0	.00000 00372	.00 00 00 E0	.00000 00521
.00 00 00 21	.00000 00076	.00 00 00 61	.00000 00225	.00 00 00 A1	.00000 00374	.00 00 00 E1	.00000 00523
.00 00 00 22	.00000 00079	.00 00 00 62	.00000 00228	.00 00 00 A2	.00000 00377	.00 00 00 E2	.00000 00526
.00 00 00 23	.00000 00081	.00 00 00 63	.00000 00230	.00 00 00 A3	.00000 00379	.00 00 00 E3	.00000 00528
.00 00 00 24	.00000 00083	.00 00 00 64	.00000 00232	.00 00 00 A4	.00000 00381	.00 00 00 E4	.00000 00530
.00 00 00 25	.00000 00086	.00 00 00 65	.00000 00235	.00 00 00 A5	.00000 00384	.00 00 00 E5	.00000 00533
.00 00 00 26	.00000 00088	.00 00 00 66	.00000 00237	.00 00 00 A6	.00000 00386	.00 00 00 E6	.00000 00535
.00 00 00 27	.00000 00090	.00 00 00 67	.00000 00239	.00 00 00 A7	.00000 00388	.00 00 00 E7	.00000 00537
.00 00 00 28	.00000 00093	.00 00 00 68	.00000 00242	.00 00 00 A8	.00000 00391	.00 00 00 E8	.00000 00540
.00 00 00 29	.00000 00095	.00 00 00 69	.00000 00244	.00 00 00 A9	.00000 00393	.00 00 00 E9	.00000 00542
.00 00 00 2A	.00000 00097	.00 00 00 6A	.00000 00246	.00 00 00 AA	.00000 00395	.00 00 00 EA	.00000 00544
.00 00 00 2B	.00000 00100	.00 00 00 6B	.00000 00249	.00 00 00 AB	.00000 00398	.00 00 00 EB	.00000 00547
.00 00 00 2C	.00000 00102	.00 00 00 6C	.00000 00251	.00 00 00 AC	.00000 00400	.00 00 00 EC	.00000 00549
.00 00 00 2D	.00000 00104	.00 00 00 6D	.00000 00253	.00 00 00 AD	.00000 00402	.00 00 00 ED	.00000 00551
.00 00 00 2E	.00000 00107	.00 00 00 6E	.00000 00256	.00 00 00 AE	.00000 00405	.00 00 00 EE	.00000 00554
.00 00 00 2F	.00000 00109	.00 00 00 6F	.00000 00258	.00 00 00 AF	.00000 00407	.00 00 00 EF	.00000 00556
.00 00 00 30	.00000 00111	.00 00 00 70	.00000 00260	.00 00 00 B0	.00000 00409	.00 00 00 F0	.00000 00558
.00 00 00 31	.00000 00114	.00 00 00 71	.00000 00263	.00 00 00 B1	.00000 00412	.00 00 00 F1	.00000 00561
.00 00 00 32	.00000 00116	.00 00 00 72	.00000 00265	.00 00 00 B2	.00000 00414	.00 00 00 F2	.00000 00563
.00 00 00 33	.00000 00118	.00 00 00 73	.00000 00267	.00 00 00 B3	.00000 00416	.00 00 00 F3	.00000 00565
.00 00 00 34	.00000 00121	.00 00 00 74	.00000 00270	.00 00 00 B4	.00000 00419	.00 00 00 F4	.00000 00568
.00 00 00 35	.00000 00123	.00 00 00 75	.00000 00272	.00 00 00 B5	.00000 00421	.00 00 00 F5	.00000 00570
.00 00 00 36	.00000 00125	.00 00 00 76	.00000 00274	.00 00 00 B6	.00000 00423	.00 00 00 F6	.00000 00572
.00 00 00 37	.00000 00128	.00 00 00 77	.00000 00277	.00 00 00 B7	.00000 00426	.00 00 00 F7	.00000 00575
.00 00 00 38	.00000 00130	.00 00 00 78	.00000 00279	.00 00 00 B8	.00000 00428	.00 00 00 F8	.00000 00577
.00 00 00 39	.00000 00132	.00 00 00 79	.00000 00281	.00 00 00 B9	.00000 00430	.00 00 00 F9	.00000 00579
.00 00 00 3A	.00000 00135	.00 00 00 7A	.00000 00284	.00 00 00 BA	.00000 00433	.00 00 00 FA	.00000 00582
.00 00 00 3B	.00000 00137	.00 00 00 7B	.00000 00286	.00 00 00 BB	.00000 00435	.00 00 00 FB	.00000 00584
.00 00 00 3C	.00000 00139	.00 00 00 7C	.00000 00288	.00 00 00 BC	.00000 00437	.00 00 00 FC	.00000 00586
.00 00 00 3D	.00000 00142	.00 00 00 7D	.00000 00291	.00 00 00 BD	.00000 00440	.00 00 00 FD	.00000 00589
.00 00 00 3E	.00000 00144	.00 00 00 7E	.00000 00293	.00 00 00 BE	.00000 00442	.00 00 00 FE	.00000 00591
.00 00 00 3F	.00000 00146	.00 00 00 7F	.00000 00295	.00 00 00 BF	.00000 00444	.00 00 00 FF	.00000 00593

TABLE OF POWERS OF TWO

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125
1 099 511 627 776	40	0.000 000 000 000 909 494 701 772 928 237 915 039 062 5
2 199 023 255 552	41	0.000 000 000 000 454 747 350 886 464 118 957 519 531 25
4 398 046 511 104	42	0.000 000 000 000 227 373 675 443 232 059 478 759 765 625
8 796 093 022 208	43	0.000 000 000 000 113 686 837 721 616 029 739 379 882 812 5
17 592 186 044 416	44	0.000 000 000 000 056 843 418 860 808 014 869 689 941 406 25
35 184 372 088 832	45	0.000 000 000 000 028 421 709 430 404 007 434 844 970 703 125
70 368 744 177 664	46	0.000 000 000 000 014 210 854 715 202 003 717 422 485 351 562 5
140 737 488 355 328	47	0.000 000 000 000 007 105 427 357 601 001 858 711 242 675 781 25
281 474 976 710 656	48	0.000 000 000 000 003 552 713 678 800 500 929 355 621 337 890 625
562 949 953 421 312	49	0.000 000 000 000 001 776 356 839 400 250 464 677 810 668 945 312 5
1 125 899 906 842 624	50	0.000 000 000 000 000 888 178 419 700 125 232 338 905 334 472 656 25
2 251 799 813 685 248	51	0.000 000 000 000 000 444 089 209 850 062 616 169 452 667 236 328 125
4 503 599 627 370 496	52	0.000 000 000 000 000 222 044 604 925 031 308 084 726 333 618 164 062 5
9 007 199 254 740 992	53	0.000 000 000 000 000 111 022 302 462 515 654 042 363 166 809 082 031 25
18 014 398 509 481 984	54	0.000 000 000 000 000 055 511 151 231 257 827 021 181 583 404 541 015 625
36 028 797 018 963 968	55	0.000 000 000 000 000 027 755 575 615 628 913 510 590 791 702 270 507 812 5
72 057 594 037 927 936	56	0.000 000 000 000 000 013 877 787 807 814 456 755 295 395 851 135 253 906 25
144 115 188 075 855 872	57	0.000 000 000 000 000 006 938 893 903 907 228 377 647 697 925 567 626 953 125
288 230 376 151 711 744	58	0.000 000 000 000 000 003 469 446 951 953 614 188 823 848 962 783 813 476 562 5
576 460 752 303 423 488	59	0.000 000 000 000 000 001 734 723 475 976 807 094 411 924 481 391 906 738 281 25
1 152 921 504 606 846 976	60	0.000 000 000 000 000 000 867 361 737 988 403 547 205 962 240 695 953 369 140 625
2 305 843 009 213 693 952	61	0.000 000 000 000 000 000 433 680 868 994 201 773 602 981 120 347 976 684 570 312 5
4 611 686 018 427 387 904	62	0.000 000 000 000 000 000 216 840 434 497 100 886 801 490 560 173 988 342 285 156 25
9 223 372 036 854 775 808	63	0.000 000 000 000 000 000 108 420 217 248 550 443 400 745 280 086 994 171 142 578 125

MATHEMATICAL CONSTANTS

Constant	Decimal Value	Hexadecimal Value
π	3.14159 26535 89793	3.243F 6A89
π^{-1}	0.31830 98861 83790	0.517C C1B7
$\sqrt{\pi}$	1.77245 38509 05516	1.C5BF 891C
$\ln \pi$	1.14472 98858 49400	1.250D 048F
e	2.71828 18284 59045	2.B7E1 5163
e^{-1}	0.36787 94411 71442	0.5E2D 58D9
$\sqrt{e}$	1.64872 12707 00128	1.A612 98E2
$\log_{10} e$	0.43429 44819 03252	0.6F2D EC55
$\log_2 e$	1.44269 50408 88963	1.7154 7653
γ	0.57721 56649 01533	0.93C4 67E4
$\ln \gamma$	-0.54953 93129 81645	-0.8CAE 9BC1
$\sqrt{2}$	1.41421 35623 73095	1.6A09 E668
$\ln 2$	0.69314 71805 59945	0.B172 17F8
$\log_{10} 2$	0.30102 99956 63981	0.4D10 4D42
$\sqrt{10}$	3.16227 76601 68379	3.298B 075C
$\ln 10$	2.30258 40929 94046	2.4D76 3777

APPENDIX B. INSTRUCTION TIMING

INSTRUCTION TIMES

The instruction times for the Xerox 530 instructions are calculated by adding the preparation time and the instruction execution time. These times are shown in Table B-1. For those instructions where a range of numbers is shown for the preparation time, the preparation time is a function of the specific effective address mode as shown in note ① to Table B-1. All of the times shown in the table are accurate within ±0.1%, assume no input/output or memory interference, and assume that all instructions and operands are in the same 8K memory module.

Table B-1. Instruction Preparation and Execution Times (in μsecs.)

Instruction/Mnemonic	Preparation Time①	Instruction Execution Time	Minimum	Maximum
Memory Reference Instructions				
LDA	0-.80	1.92	1.92	2.72
STA	0-.80	2.24	2.24	3.04
LDX (Normal)	0-.80	1.92	1.92	2.72
LDX (Interrupt Exit) See "Interrupt Exit" under "Direct Control Instructions"				
ADD	0-.80	1.92	1.92	2.72
SUB	0-.80	1.92	1.92	2.72
AND	0-.80	1.92	1.92	2.72
IM	0-.80	2.72	2.72	3.52
CP	0-.80	1.92	1.92	2.72
MUL	0-.80	8.00	8.00	8.80
DIV	0-.80	13.12-13.76	13.12	14.56
Branch Instructions				
B	0-.80	1.12	1.12	1.92
BAN (no Branch)	0	.80	.80	.80
BAN (Branch)	0	1.12	1.12	1.12
BAZ (no Branch)	0	.80	.80	.80
BAZ (Branch)	0	1.12	1.12	1.12
BEN (no Branch)	0	.80	.80	.80
BEN (Branch)	0	1.12	1.12	1.12
BNC (no Branch)	0	.80	.80	.80
BNC (Branch)	0	1.12	1.12	1.12
BNO (no Branch)	0	.80	.80	.80
BNO (Branch)	0	1.12	1.12	1.12
BIX (no Branch)	0	.80	.80	.80
BIX (Branch)	0	1.12	1.12	1.12
BXNO (no Branch)	0	.80	.80	.80
BXNO (Branch)	0	1.12	1.12	1.12
BXNC (no Branch)	0	.80	.80	.80
BXNC (Branch)	0	1.12	1.12	1.12

Table B-1. Instruction Preparation and Execution Times (in μsecs.) (cont.)

Instruction/Mnemonic	Preparation Time (1)	Instruction Execution Time	Minimum	Maximum
Shift Instructions				
SARS	0-.80	2.56 + .32N (2)	2.56	8.16
SARD	0-.80	2.88 + .32N (3)	2.88	13.60
SALS	0-.80	2.56 + .32N (2)	2.56	8.16
SALD	0-.80	2.56 + .32N (3)	2.56	13.28
SCRS	0-.80	2.56 + .32N (2)	2.56	8.16
SCRD	0-.80	2.56 + .32N (3)	2.56	13.28
SCLS	0-.80	2.56 + .32N (2)	2.56	8.16
SCLD	0-.80	2.88 + .32N (3)	2.56	13.60
Normalize Shift	0-.80	3.52 + .32N (4)	3.52	14.24
Copy Instructions				
RCPY, RCPYI, RCPYC, RADD, RADDI, RADDC, RCLA, RCLAI, RCLAC, ROR, RORI, RORC, RAND, REOR	0	.96 (5)	.96	1.44
RANDI, RANDC, REORI, REORC	0	1.28 (5)	1.28	1.76
Direct Control Instructions				
RD - Internal Mode (Except for the following)	0-.80	2.88 - 4.16	2.88	4.96
I/O Reset	0-.80	83.80 (12)	83.80	84.60
IOP Registers	0-.80	5.44 (13)	5.44	-
IOP Instructions	0-.80	10.88 + .32 FSL (14)	11.20	-
RD - Interrupt Mode	0-.80	8.96 (15)	8.96	15.52
RD - External Mode	0-.80	3.52 + .32 FSA (16)	3.84	-
WD - Internal Mode (Except for the following)	0-.80	2.24 - 4.32	2.24	5.12
Interrupt Exit (6)	0-.80	11.52 (17)	11.52	16.26
IOP Registers	0-.80	5.12 (13)	5.12	-
WD - Interrupt Mode	0-.80	9.28 (15)	9.28	15.84
WD - External Mode	0-.80	3.84 + .32 FSA (16)	4.16	-
General Register Instructions				
LW	0-.80 (7)	3.84 (8)	3.84	4.64
STW	0-.80 (7)	4.48 (8)	4.48	5.28
AW	0-.80 (7)	3.84 (8)	3.84	4.64
SW	0-.80 (7)	3.84 (8)	3.84	4.64
AND	0-.80 (7)	3.84 (8)	3.84	4.64
CW	0-.80 (7)	4.48 (8)	4.48	5.28
Multiple Register Instructions				
LDM	0-.80 (7)	4.64 - 7.68 (8)(9)(10)	4.64	8.48
LDD	0-.80 (7)	4.00 (8)	4.00	4.80

Table B-1. Instruction Preparation and Execution Times (in μsecs.) (cont.)

Instruction/Mnemonic	Preparation Time (1)	Instruction Execution Time	Minimum	Maximum
Multiple Register Instructions (cont.)				
STM	0-.80	5.28 - 8.32 (8)(9)(10)	5.28	9.12
STD	0-.80	4.16	4.16	4.96
DAD	0-.80 (7)	4.00 (8)	4.00	4.80
DSB	0-.80	4.00	4.00	4.80
CPD	0-.80	4.32	4.32	5.12
Floating Point Instructions				
FLD	0-.80	3.68	3.68	4.48
FST	0-.80	3.84	3.84	4.64
FAD	0-.80	8.80 + .32J + .96K (18)	8.80	38.40
FSB (11)	0-.80	8.80 + .32J + .96K (18)	8.80	38.40
FMP	0-.80	32.96 Typical (19)	31.36	35.36
FDV	0-.80	77.56 Typical (19)	8.32	101.92
FCP	0-.80	5.96 Typical	3.52	20.36
Field Addressing Instructions				
LLF (11)				
16-bit field (starting on bit 0) (20)	0-.80	8.64 (8)(23)	8.64	13.92
8-bit field (starting on bit 0 or 8) (21)	0-.80	9.28 - 9.60 (8)(23)	9.28	14.88
4-bit field (starting on bit 0, 4, 8, or 12) (22)	0-.80	9.28 - 9.60 (8)(23)	9.28	14.88
1-bit field	0-.80 (7)	10.88 (8)(23)	10.88	16.16
Other fields – contained within one word	0-.80	11.20 - 15.68 (8)(23)	11.20	20.96
Other fields – contained within two words	0-.80	11.52 - 16.00 (8)(23)	11.52	21.28
LAF (11)				
16-bit field (starting on bit 0) (20)	0-.80	8.64 (8)(23)	8.64	13.92
8-bit field (starting on bit 0 or 8) (21)	0-.80	10.24 - 10.56 (8)(23)	10.24	15.84
4-bit field (starting on bit 0, 4, 8, or 12) (22)	0-.80	10.24 - 10.56 (8)(23)	10.24	15.84
1-bit field	0-.80 (7)	10.88 (8)(23)	10.88	16.16
Other fields – contained within one word	0-.80	11.20 - 15.68 (8)(23)	11.20	20.96
Other fields – contained within two words	0-.80	11.52 - 16.00 (8)(23)	11.52	21.28
CLF (11)				
16-bit field (starting on bit 0) (20)	0-.80	8.96 (8)(23)	8.96	14.24
8-bit field (starting on bit 0 or 8) (21)	0-.80	9.60 - 9.92 (8)(23)	9.60	15.20
4-bit field (starting on bit 0, 4, 8, or 12) (22)	0-.80	9.60 - 9.92 (8)(23)	9.60	15.20
1-bit field	0-.80 (7)	11.20 (8)(23)	11.20	16.48
Other fields – contained within one word	0-.80	11.52 - 16.00 (8)(23)	11.52	21.28
Other fields – contained within two words	0-.80	11.84 - 16.32 (8)(23)	11.84	21.60

Table B-1. Instruction Preparation and Execution Times (in μsecs.) (cont.)

Instruction/Mnemonic	Preparation Time (1)	Instruction Execution Time	Minimum	Maximum
Field Addressing Instructions (cont.)				
CAF (11)				
16-bit field (starting on bit 0) (20)	0-.80 (7)	8.64 (8) (23)	8.64	13.92
8-bit field (starting bit 0 or 8) (21)	0-.80	10.24 - 10.56 (8) (23)	10.24	15.84
4-bit field (starting on bit 0, 4, 8, or 12) (22)	0-.80	10.24 - 10.56 (8) (23)	10.24	15.84
1-bit field	0-.80	10.88 (8) (23)	10.88	16.16
Other fields – contained within one word	0-.80	11.20 - 15.68 (8) (23)	11.20	20.96
Other fields – contained within two words	0-.80	11.52 - 16.00 (8) (23)	11.52	21.28
SLF (11)				
16-bit field (starting on bit 0) (20)	0-.80 (7)	10.88 (8) (23)	10.88	16.16
8-bit field (starting on bit 0 or 8) (21)	0-.80	10.56 - 10.88 (8) (23)	10.56	16.16
4-bit field (starting on bit 0, 4, 8, or 12) (22)	0-.80	10.56 - 10.88 (8) (23)	10.56	16.16
1-bit field	0-.80	11.20 (8) (23)	11.20	16.48
Other fields – contained within one word	0-.80	11.52 - 16.00 (8) (23)	11.52	21.28
Other fields – contained within two words	0-.80	11.82 - 16.32 (8) (23)	11.82	21.60
SOF (11)				
16-bit field (starting on bit 0) (20)	0-.80 (7)	9.44 (8) (23)	9.44	14.72
8-bit field (starting on bit 0 or 8) (21)	0-.80	10.08 (8) (23)	10.08	15.36
4-bit field (starting on bit 0, 4, 8, or 12) (22)	0-.80	10.08 (8) (23)	10.08	15.36
1-bit field	0-.80	11.68 (8) (23)	11.68	16.96
Other fields – contained within one word	0-.80	11.68 (8) (23)	11.68	16.96
Other fields – contained within two words	0-.80	14.24 (8) (23)	14.24	19.52
SZF (11)				
16-bit field (starting on bit 0) (20)	0-.80 (7)	9.12 (8) (23)	9.12	14.40
8-bit field (starting on bit 0 or 8) (21)	0-.80	9.76 (8) (23)	9.76	15.04
4-bit field (starting on bit 0, 4, 8, or 12) (22)	0-.80	9.76 (8) (23)	9.76	15.04
1-bit field	0-.80	11.36 (8) (23)	11.36	16.64
Other fields – contained within one word	0-.80	11.36 (8) (23)	11.36	16.64
Other fields – contained within two words	0-.80	13.60 (8) (23)	13.60	18.88
STF (11)				
16-bit field (starting on bit 0) (20)	0-.80 (7)	10.40 (8) (23)	10.40	15.68
8-bit field (starting on bit 0 or 8) (21)	0-.80	10.72 - 11.04 (8) (23)	10.72	16.32
4-bit field (starting on bit 0, 4, 8, or 12) (22)	0-.80	10.72 - 11.36 (8) (23)	10.72	16.64
1-bit field	0-.80	13.28 - 18.08 (8) (23)	13.28	23.36
Other fields – contained within one word	0-.80	13.28 - 17.76 (8) (23)	13.28	23.04
Other fields – contained within two words	0-.80	16.16 - 20.64 (8) (23)	16.16	25.92

Table B-1. Instruction Preparation and Execution Times (in μsecs.) (cont.)

Notes:

① Preparation time depends on effective address mode according to the following table:

R	I	X	S	Effective Address†	Preparation Timing (μsec.)
0	0	0	0	D	0
0	0	0	1	D + (B)	0
0	0	1	0	D + (X)	0
0	0	1	1	D + (B) + (X)	.32
0	1	0	0	(D)	.80
0	1	0	1	(D + (B))	.80
0	1	1	0	(D) + (X)	.80
0	1	1	1	(D + (B)) + (X)	.80
1	0	0		(W) + SD	0
1	0	1		(W) + SD + (X)	.32
1	1	0		((W) + SD)	.80
1	1	1		((W) + SD) + (X)	.80

†Refer to "Effective Address Computation", Chapter 2, for definition of symbols.

② For single register shift instructions, N is the number of bit positions shifted ($0 \leq N \leq 15$).

③ For double register shift instructions, N is the number of bit positions shifted ($0 \leq N \leq 31$).

④ For normalize shift instructions, N is the number of bit positions registers E and A must be shifted to generate a normalized number ($0 \leq N \leq 31$).

⑤ For copy instructions, add .48 μsec if the destination register is general register 1 (P).

⑥ Time shown for interrupt exit is for both the set exit WD instruction and the LDX instruction.

⑦ Preparation time varies according to the addressing mode specified by the R, I, X, and S bits of the second word of the two-word instruction. Preparation times are the same as shown in note ① above.

⑧ Instruction execution time is for the entire two-word instruction.

⑨ Two-register LDM and STM instructions which operate specifically on the E and A registers have the same execution times as LDD and STD instructions, respectively.

⑩ LDM and STM instruction execution time depends on the number of registers to be stored according to the following table:

Number of Registers	LDM Instruction Execution Time	STM Instruction Execution Time
2	4.64	5.28
3	5.44	6.08
4	6.08	6.72
5	6.88	7.52
6	7.68	8.32

⑪ Optional Instruction.

⑫ Generates approximately 81 μsec reset pulse.

Table B-1. Instruction Preparation and Execution Times (in μsecs.) (cont.)

Notes: (cont.)

(13) Assumes IOP is not busy. If IOP is busy, CPU must wait until the end of the current IOP service cycle.

(14) Assumes external device controller. FSL is Function Strobe Leading Acknowledge.

(15) Assumes standard interrupts. Add 2.88 μsec for first group of 12 interrupt levels. Add 5.76 μsec for second group of 12 optional interrupt levels.

(16) FSA is Function Strobe Acknowledge from external device.

(17) Assumes standard interrupts. Add 1.92 μsec for first group of 12 optional interrupt levels. Add 3.94 μsec for second group of optional interrupt levels. Also assumes not reentering floating point mode.

(18) J is the number of prealignment shifts required. K is the number of postnormalization shifts required.

(19) FMP and FDV instruction execution times are typical times based on a random distribution of ones and zeros in the operands.

(20) Time shown is for 16-bit field starting on bit 0. For a 16-bit field starting on some other bit position, use the "other fields – contained within two words" execution times.

(21) Time shown is for 8-bit field starting on bit 0 or 8. For an 8-bit field starting on some other bit position, use the appropriate "other fields" execution times, depending on whether the 8-bit field is contained within one memory word or two memory words.

(22) Time shown is for 4-bit field starting on bit 0, 4, 8, or 12. For a 4-bit field starting on some other bit position, use the appropriate "other fields" execution times, depending on whether the 4-bit field is contained within one memory word or two memory words.

(23) Add 0.32 μsec if register indexing is specified by the RX field in the instruction. Add 3.52 μsec if self-incrementing is specified by the SX field in the instruction. Add 4.16 if self-decrementing is specified by the SX field in the instruction. See the description of "Field Addressing Instructions" in Chapter 3.

APPENDIX C. READ/WRITE (MODE 0) INSTRUCTIONS

This appendix is comprised of eight tables. Tables C-1 through C-4 show the 256 different function values for a READ DIRECT (Mode 0) instruction. Tables C-5 through C-8 show the 256 different function values for a WRITE DIRECT (Mode 0) instruction. Each table consists of 64 function values (assigned and unassigned). Unassigned function values are reserved and must not be used. Attempting to use an unassigned function value results in a Machine Fault Interrupt.

Table C-1. READ DIRECT (Mode 0) Instruction, Function Values X'00'-X'3F'

Hexadecimal Value of Bits 8-11 of Function Field	Hexadecimal Value of Bits 12-15 of Function Field															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		← Copy contents of specified general register into A →							← Copy contents of specified I/O channel register into A →							
		P	L	T	X	B	E	A	X'08'	X'09'	X'0A'	X'0B'	X'0C'	X'0D'	X'0E'	X'0F'
1	← Copy contents of specified I/O channel register into A →															
	X'10'	X'11'	X'12'	X'13'	X'14'	X'15'	X'16'	X'17'	X'18'	X'19'	X'1A'	X'1B'	X'1C'	X'1D'	X'1E'	X'1F'
2	← Copy contents of specified I/O channel register into A →															
	X'20'	X'21'	X'22'	X'23'	X'24'	X'25'	X'26'	X'27'	X'28'	X'29'	X'2A'	X'2B'	X'2C'	X'2D'	X'2E'	X'2F'
3	← Copy contents of specified I/O channel register into A →															
	X'30'	X'31'	X'32'	X'33'	X'34'	X'35'	X'36'	X'37'	X'38'	X'39'	X'3A'	X'3B'	X'3C'	X'3D'	X'3E'	X'3F'

Note: Except for function value X'00', which is unassigned, each of the function values designates a general register or I/O channel register whose contents are copied into the A register (accumulator). Function values X'08' and X'09' are associated with the first I/O channel and function values X'3E' and X'3F' are associated with the last I/O channel.

Table C-2. READ DIRECT (Mode 0) Instruction, Function Values X'40'-X'7F'

Hexadecimal Value of Bits 8-11 of Function Field	Hexadecimal Value of Bits 12-15 of Function Field															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4		SIO	TIO		TDV				HIO							
5	AIO															
6	I/O RESET															
7																

Note: Each of the assigned function values within this group is labeled with an appropriate mnemonic representing an I/O instruction. See Chapter 4 for further details.

Table C-3. READ DIRECT (Mode 0) Instruction, Function Values X'80'-X'BF'

Hexadecimal Value of Bits 8-11 of Function Field	Hexadecimal Value of Bits 12-15 of Function Field															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	Copy (DATA) into A								FA RX=1 SX=0	FA RX=1 SX=1	GR=2	GR=3	GR=4	GR=5	GR=6	FA RX=1 SX=7
9	FA RX=2 SX=0	FA RX=2 SX=1	LDM or STM X=2 Y=2	LDM or STM X=2 Y=3	LDM or STM X=2 Y=4	LDM or STM X=2 Y=5	LDD, STD, DAD, DSB, or CPD X=2 Y=6	FA RX=2 SX=7	FA RX=3 SX=0	FA RX=3 SX=1	LDM or STM X=3 Y=2	LDM or STM X=3 Y=3	LDM or STM X=3 Y=4	LDM or STM X=3 Y=5	SFM	FA RX=3 SX=7
A	FA RX=4 SX=0	FA RX=4 SX=1	LDM or STM X=4 Y=2	LDM or STM X=4 Y=3	LDM or STM X=4 Y=4			FA RX=4 SX=7	FA RX=5 SX=0	FA RX=5 SX=1	LDM or STM X=5 Y=2	LDM or STM X=5 Y=3				FA RX=5 SX=7
B	FA RX=6 SX=0	FA RX=6 SX=1	LDM or STM X=6 Y=2					FA RX=6 SX=7	FA RX=7 SX=0	FA RX=7 SX=1						FA RX=7 SX=7

Note: Except for function values X'80' (copy contents of DATA switches into A register) and X'9E' (SFM; Set Floating Mode), which are one-word instructions, each of the other assigned function values within this group is the first word of a two-word instruction sequence. These function values are identified and described with the following symbols and mnemonics:

CPD A doubleword compare instruction.

DAD A doubleword add instruction.

DSB A doubleword subtract instruction.

FA A field addressing instruction.

GR A general register instruction. The accompanying value (2-6) indicates the general register that is used as the accumulator.

LDD A doubleword load instruction.

LDM A multiple register (other than doubleword) load instruction.

RX For field addressing instructions, RX=1 signifies no indexing register, RX=2 through RX=7 signify the indexing register.

STD A doubleword store instruction.

STM A multiple register (other than doubleword) store instruction.

SX For field addressing instructions, SX=0 signifies no self-indexing, SX=1 signifies self-incrementing, SX=7 signifies self-decrementing.

X For multiple register and doubleword instructions, the X value signifies the number of registers.

Y For multiple register and doubleword instructions, the Y value signifies the first register.

Table C-4. READ DIRECT (Mode 0) Instruction, Function Values X'C0'-X'FF'

Hexadecimal Value of Bits 8-11 of Function Field	Hexadecimal Value of Bits 12-15 of Function Field															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
C	Copy (PSW1)															
D																
E	Copy (PSW1)				Copy (PSW1); reset EI				Copy (PSW1); reset II				Copy (PSW1); reset EI & II			
F	Copy (PSW1)				Copy (PSW1); set EI				Copy (PSW1); set II				Copy (PSW1); set EI & II			

Note: Each of the assigned function values causes the contents of the first word of the Program Status Doubleword (PSW1) to be copied into the A register (accumulator); zeros are copied into those positions that do not correspond to a program status indicator. In addition, function values X'E4', X'E8', X'EC', X'F4', X'F8', and X'FC' permit the EI and II bits to be set or reset, as indicated.

Table C-5. WRITE DIRECT (Mode 0) Instruction, Function Values X'00'-X'3F'

Hexadecimal Value of Bits 8-11 of Function Field	Hexadecimal Value of Bits 12-15 of Function Field															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		← Copy (A) into specified general register →							← Copy (A) into specified I/O channel register →							
		P	L	T	X	B	E	A	X'08'	X'09'	X'0A'	X'0B'	X'0C'	X'0D'	X'0E'	X'0F'
1	← Copy (A) into specified I/O channel register →															
	X'10'	X'11'	X'12'	X'13'	X'14'	X'15'	X'16'	X'17'	X'18'	X'19'	X'1A'	X'1B'	X'1C'	X'1D'	X'1E'	X'1F'
2	← Copy (A) into specified I/O channel register →															
	X'20'	X'21'	X'22'	X'23'	X'24'	X'25'	X'26'	X'27'	X'28'	X'29'	X'2A'	X'2B'	X'2C'	X'2D'	X'2E'	X'2F'
3	← Copy (A) into specified I/O channel register →															
	X'30'	X'31'	X'32'	X'33'	X'34'	X'35'	X'36'	X'37'	X'38'	X'39'	X'3A'	X'3B'	X'3C'	X'3D'	X'3E'	X'3F'

Note: Except for function value X'00', which is unassigned, each of the function values specifies either a general or I/O channel register that is to receive a word of information, as copied from the A register. (These copy instructions are the converse of READ DIRECT (Mode 0) instructions with the same function values.)

Table C-6. WRITE DIRECT (Mode 0) Instruction, Function Values X'40'-X'7F'

Hexadecimal Value of Bits 8-11 of Function Field	Hexadecimal Value of Bits 12-15 of Function Field															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4		← See Note 2 →							← See Note 3 →							
		P	L	T	X	B	E	A	X'08'	X'09'	X'0A'	X'0B'	X'0C'	X'0D'	X'0E'	X'0F'
5	← See Note 3 →															
	X'10'	X'11'	X'12'	X'13'	X'14'	X'15'	X'16'	X'17'	X'18'	X'19'	X'1A'	X'1B'	X'1C'	X'1D'	X'1E'	X'1F'
6	← See Note 3 →															
	X'20'	X'21'	X'22'	X'23'	X'24'	X'25'	X'26'	X'27'	X'28'	X'29'	X'2A'	X'2B'	X'2C'	X'2D'	X'2E'	X'2F'
7	← See Note 3 →															
	X'30'	X'31'	X'32'	X'33'	X'34'	X'35'	X'36'	X'37'	X'38'	X'39'	X'3A'	X'3B'	X'3C'	X'3D'	X'3E'	X'3F'

Notes:
1. Except for function value X'40', which is unassigned, the function value for a specified register is X'40' greater than the address of the specified register.
2. Copy bit 0 of specified general register into Overflow; then reset bit 0 of specified general register to 0.
3. Copy bit 0 of specified I/O channel register into Overflow; then reset bit 0 of specified I/O channel register to 0.

Table C-7. WRITE DIRECT (Mode 0) Instruction, Function Values X'80-X'BF'

Value of Bits 8-11 of Function Field	Hexadecimal Value of Bits 12-15 of Function Field															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	← Copy (A) into specified protection register →															
	X'0'	X'1'	X'2'	X'3'	X'4'	X'5'	X'6'	X'7'	X'8'	X'9'	X'A'	X'B'	X'C'	X'D'	X'E'	X'F'
9																
A																
B																

Note: Except for function values X'80'-X'8F', all function values are unassigned.

Table C-8. WRITE DIRECT (Mode 0) Instruction, Function Values X'C0'-X'FF'

Hexadecimal Value of Bits 8-11 of Function Field	Hexadecimal Value of Bits 12-15 of Function Field															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
C			Diagnose IOP-1	Diagnose IOP-2												
D	Set Wait flip-flop								Set Exit condition							
E					Reset EI to 0				Reset II to 0				Reset EI & II to 0			
F					Set EI to 1				Set II to 1				Set EI & II to 1			

Note: Except for the ten function values shown above, all function values within this group are unassigned.

Xerox Corporation
701 South Aviation Boulevard
El Segundo, California 90245

XEROX

Reader Comment Form

We would appreciate your comments and suggestions for improving this publication.

Publication No.	Rev. Letter	Title	Current Date

How did you use this publication?

☐ Learning ☐ Installing ☐ Sales
☐ Reference ☐ Maintaining ☐ Operating

Is the material presented effectively?

☐ Fully Covered ☐ Well Illustrated ☐ Well Organized ☐ Clear

What is your overall rating of this publication?

☐ Very Good ☐ Fair ☐ Very Poor
☐ Good ☐ Poor

What is your occupation?

Your other comments may be entered here. Please be specific and give page, column, and line number references where applicable. To report errors, Please use the Xerox Software Improvement or Difficulty Report (1188) instead of this form.

Your Name & Return Address

2190(12/72)

Thank You For Your Interest. (fold & fasten as shown on back, no postage needed if mailed in U.S.A.)

Staple

Staple

Fold

First Class
Permit No. 229
El Segundo,
California

BUSINESS REPLY MAIL

No postage stamp necessary if mailed in the United States

Postage will be paid by

Xerox Corporation
701 South Aviation Boulevard
El Segundo, California 90245

Attn: Programming Publications

Fold

XEROX 530 INSTRUCTIONS (OPERATION CODES)

Format†		Syntax††			
First Word	Second Word	Command	Argument	Instruction Name	Page
0000 RIXS D		WD	*a, x, b	Write Direct	29
0001 RIXS D		RD	*a, x, b	Read Direct	32
0001 0000 0100 0001		SIO		Start Input/Output	57
0001 0000 0100 0010		TIO		Test Input/Output	57
0001 0000 0100 0100		TDV		Test Device	58
0001 0000 0100 1000		HIO		Halt Input/Output	58
0001 0000 0101 0000		AIO		Acknowledge I/O Interrupt	58
0001 0000 10 RX SX	0101 RIXS D	CLF, rx, sx	*a, x, b	Compare Logical Field (optional)	50
0001 0000 10 RX SX	1000 RIXS D	LLF, rx, sx	*a, x, b	Load Logical Field (optional)	49
0001 0000 10 RX SX	1001 RIXS D	LAF, rx, sx	*a, x, b	Load Arithmetic Field (optional)	49
0001 0000 10 RX SX	1010 RIXS D	STF, rx, sx	*a, x, b	Store Field (optional)	49
0001 0000 10 RX SX	1011 RIXS D	STZ, rx, sx	*a, x, b	Store Zero Field (optional)	50
0001 0000 10 RX SX	1101 RIXS D	CAF, rx, sx	*a, x, b	Compare Arithmetic Field (optional)	51
0001 0000 10 RX SX	1100 RIXS D	SOF, rx, sx	*a, x, b	Store Ones Field (optional)	50
0001 0000 10 RX SX	1111 RIXS D	SLF, rx, sx	*a, x, b	Sense Left Bit of Field (optional)	51
0001 0000 10001 GR	1000 RIXS D	LW, r	*a, x, b	Load Word	34
0001 0000 10001 GR	1001 RIXS D	AND, r	*a, x, b	Logical And (two-word instruction)	36
0001 0000 10001 GR	1010 RIXS D	AW, r	*a, x, b	Add Word	35
0001 0000 10001 GR	1011 RIXS D	SW, r	*a, x, b	Subtract Word	35
0001 0000 10001 GR	1101 RIXS D	CW, r	*a, x, b	Compare Word	36
0001 0000 10001 GR	1110 RIXS D	STW, r	*a, x, b	Store Word	35
0001 0000 10 XXX YYY	1000 RIXS D	LDM	*a, x, b, fr, nr	Load Multiple	37
0001 0000 10 XXX YYY	1110 RIXS D	STM	*a, x, b, fr, nr	Store Multiple	37
0001 0000 10 010 110	1000 RIXS D	LDD	*a, x, b	Load Double	37
0001 0000 10 010 110	1010 RIXS D	DAD	*a, x, b	Double Add	38
0001 0000 10 010 110	1011 RIXS D	DSB	*a, x, b	Double Subtract	38
0001 0000 10 010 110	1101 RIXS D	CPD	*a, x, b	Compare Double	38
0001 0000 10 010 110	1110 RIXS D	STD	*a, x, b	Store Double	37
0001 0000 1001 1110		SFM		Set Floating Mode (optional)	41
0010 RIXS D		S	*a, x, b	Shift	22
0010 0000 000 count		SARS	c, x, b	Shift Arithmetic Right Single	22
0010 0000 100 count		SARD	c, x, b	Shift Arithmetic Right Double	22
0010 0000 001 count		SALS	c, x, b	Shift Arithmetic Left Single	23
0010 0000 101 count		SALD	c, x, b	Shift Arithmetic Left Double	23
0010 0000 010 count		SCRS	c, x, b	Shift Circular Right Single	23
0010 0000 110 count		SCRD	c, x, b	Shift Circular Right Double	23
0010 0000 011 count		SCLS	c, x, b	Shift Circular Left Single	24
0010 0000 111 count		SCLD	c, x, b	Shift Circular Left Double	24
0011 RIXS D		MUL or FMP	*a, x, b	Multiply or Floating Multiply (optional) if FM bit is set	28, 43
0100 RIXS D		B	*a, x, b	Branch	24
0101 RIXS D		DIV or FDV	*a, x, b	Divide, or Floating Divide (optional) if FM bit is set	28, 43
0110 000S D		BNO	a	Branch if No Overflow	25
0110 001S D		BNC	a	Branch if No Carry	25
0110 010S D		BAZ	a	Branch if Accumulator Zero	25
0110 011S D		BIX	a	Branch on Incrementing Index	25
0110 100S D		BXNO	a	Branch on Incrementing Index and No Overflow	25
0110 101S D		BXNC	a	Branch on Incrementing Index and No Carry	25
0110 110S D		BEN	a	Branch if Extended Accumulator Negative	25
0110 111S D		BAN	a	Branch if Accumulator Negative	24
0111 0000 0 DR I/S SR		RAND	*s, d	Register AND	27
0111 0001 0 DR I/S SR		RANDI	*s, d	Register AND and Increment	27
0111 0010 0 DR I/S SR		RANDC	*s, d	Register AND and Carry	28
0111 0100 0 DR I/S SR		ROR	*s, d	Register OR	27
0111 0100 1 DR I/S SR		RCPY	*s, d	Register Copy	26
0111 0101 0 DR I/S SR		RORI	*s, d	Register OR and Increment	27
0111 0101 1 DR I/S SR		RCPYI	*s, d	Register Copy and Increment	27
0111 0110 0 DR I/S SR		RORC	*s, d	Register OR and Carry	28
0111 0110 1 DR I/S SR		RCPYC	*s, d	Register Copy and Carry	27
0111 1000 0 DR I/S SR		REOR	*s, d	Register Exclusive OR	27
0111 1001 0 DR I/S SR		REORI	*s, d	Register Exclusive OR and Increment	27
0111 1010 0 DR I/S SR		REORC	*s, d	Register Exclusive OR and Carry	28
0111 1100 0 DR I/S SR		RADD	*s, d	Register Add	26
0111 1100 1 DR I/S SR		RCLA	*s, d	Register Clear and Add	28
0111 1101 0 DR I/S SR		RADDI	*s, d	Register Add and Increment	27
0111 1101 1 DR I/S SR		RCLAI	*s, d	Register Clear, Add and Increment	28
0111 1110 0 DR I/S SR		RADDC	*s, d	Register Add and Carry	27
0111 1110 1 DR I/S SR		RCLAC	*s, d	Register Clear, Add, and Carry	28
1000 RIXS D		LDA or FLD	*a, x, b	Load Register A, or Floating Load (optional) if FM bit is set	20, 42
1001 RIXS D		AND	*a, x, b	And, Logical (one-word instruction)	21
1010 RIXS D		ADD or FAD	*a, x, b	Add, or Floating Add (optional) if FM bit is set	21, 42
1011 RIXS D		SUB or FSB	*a, x, b	Subtract, or Floating Subtract (optional) if FM bit is set	21, 42
1100 RIXS D		LDX	*a, x, b	Load Index	21
1101 RIXS D		CP or FCP	*a, x, b	Compare, or Floating Compare (optional) if FM bit is set	24, 43
1110 RIXS D		STA or FST	*a, x, b	Store Register A, or Floating Store (optional) if FM bit is set	21, 42
1111 RIXS D		IM	*a, x, b	Increment Memory	21

†Except for using binary notation (rather than hexadecimal) to represent fixed fields, the format is the same as described in Chapters 3 and 4.

††Refer to the Xerox Extended Symbol/LN, OPS Reference Manual, 90 10 52, for further information on symbolic notation.

701 South Aviation Boulevard
El Segundo, California 90245
213 679-4511

XEROX