SPERRY RAND

UNIVAC

9200/9300

SERIES

PROCESSOR AND STORAGE

PROGRAMMER REFERENCE

UP-7546 Rev. 1

PAGE STATUS SUMMARY

ISSUE: UP-7546 Rev. 1

Section	Page Number	Update Level
Cover/Disclaimer		
PSS	1	
Contents	1 thru 5	
1	1 thru 17	
2	1 thru 34	
3	1 thru 21	
Appendix A	1 thru 3	
Appendix B	1 thru 3	
Appendix C	1 thru 4	
Appendix D	1	
Index	1 thru 11	

Section	Page Number	Update Level

Section	Page Number	Update Level

CONTENTS

FIGURES

1. INTRODUCTION

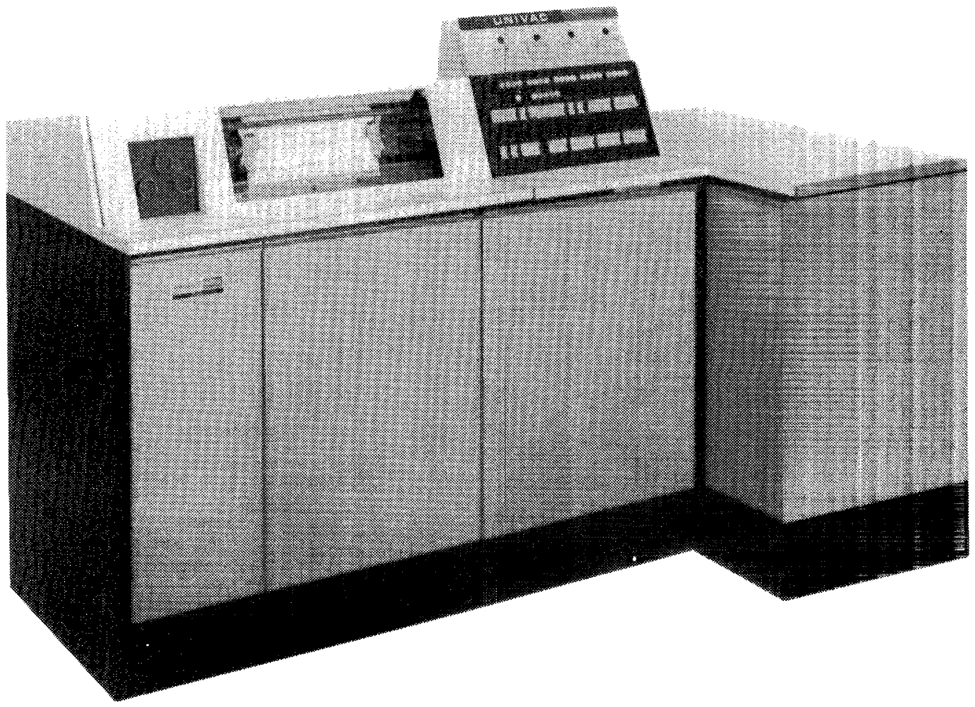


Figure 1–1. UNIVAC 9200/9200 II/9300/9300 II Systems Processor

1.1. GENERAL

This document contains a description of the UNIVAC 9200/9200 II/9300/9300 II Systems processor with optional features pertaining to expansion, internal operation of the processor, data and program information presentation, instruction repertoire and coding, and constant and storage definitions.

It should be noted that only one processor is detailed herein with the specific differences between each noted as required. The four system processors are identical in appearance. The configuration is shown in Figure 1–1.

This manual is divided into the following basic sections:

- Introduction
- Processor Unit
- Instructions
- Appendix

The appendix contains tables, charts, and diagrams as a convenience to the programmer and field engineer. The fact that many examples and explanations are given in terms of punched card input does not imply that any other of the usual input methods may not be used.

The UNIVAC 9200/9200 II/9300/9300 II Systems processor shown in Figure 1–1, is a byte-oriented data processor integrated with an attached printer and an optional externally-connected card punch and card reader (basic peripherals). A wide range of other peripheral devices may be incorporated either initially by replacing one or more of these basic peripheral devices, or later, by adding to the system.

Two major functions of the processor are decoding and storage. The decoding function analyzes each instruction to determine required operations and the location of needed information. An instruction containing labels of data in storage is decoded to find the addresses and lengths of the operands (data to be processed), as well as the particular operation that is to be performed on the data. The processor then responds to the instruction by using generated control signals. The storage (memory) portion of the processor stores the data and instructions required by the program in addressable locations that are easily accessible to the program.

The major components of the UNIVAC 9200/9200 II/9300/9300 II Systems processor are shown in Figure 1–2.

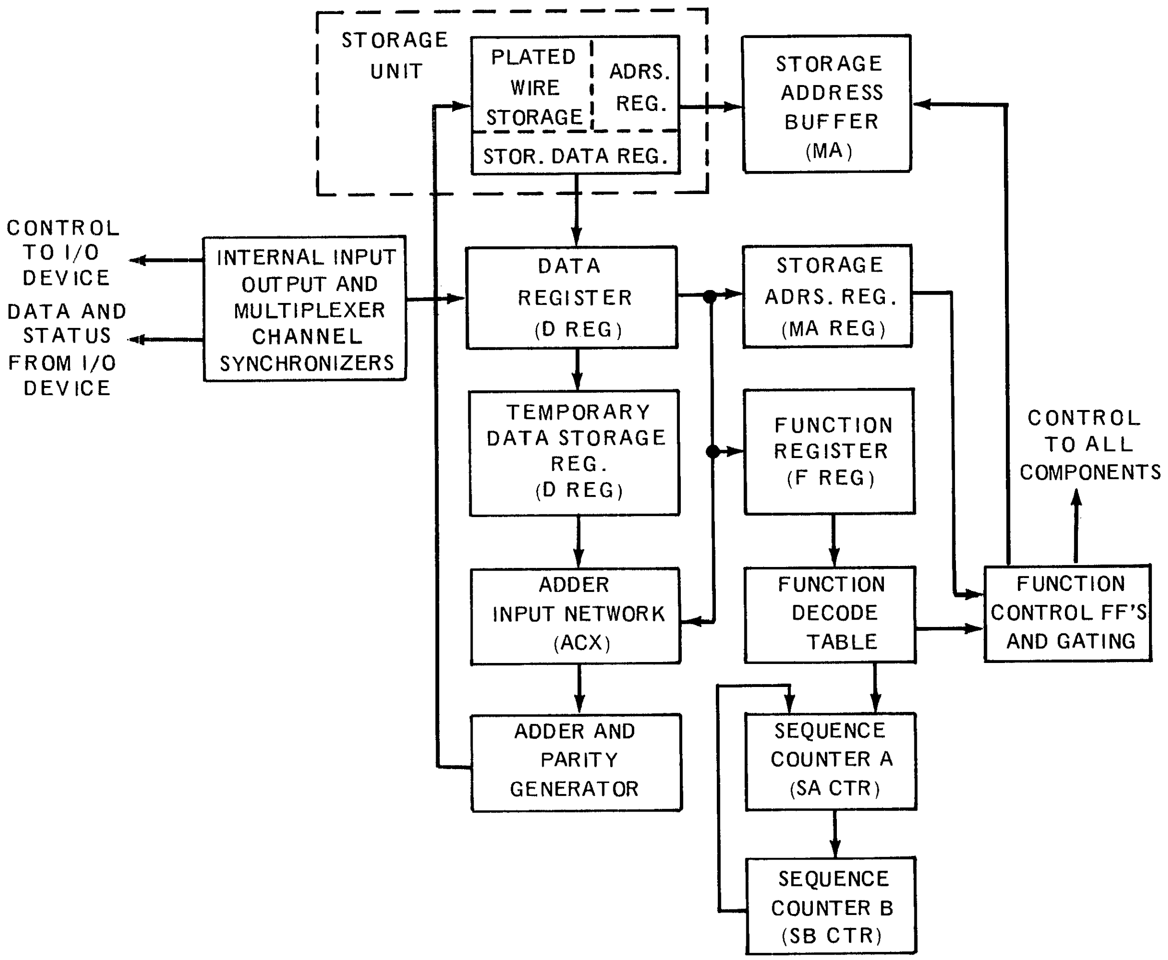

STORAGE UNIT
PLATED WIRE STORAGE
ADRS. REG.
STOR. DATA REG.
STORAGE ADDRESS BUFFER (MA)
CONTROL TO I/O DEVICE
DATA AND STATUS FROM I/O DEVICE
INTERNAL INPUT OUTPUT AND MULTIPLEXER CHANNEL SYNCHRONIZERS
DATA REGISTER (D REG)
STORAGE ADRS. REG. (MA REG)
TEMPORARY DATA STORAGE REG. (D REG)
FUNCTION REGISTER (F REG)
CONTROL TO ALL COMPONENTS
ADDER INPUT NETWORK (ACX)
FUNCTION DECODE TABLE
FUNCTION CONTROL FF'S AND GATING
ADDER AND PARITY GENERATOR
SEQUENCE COUNTER A (SA CTR)
SEQUENCE COUNTER B (SB CTR)

Figure 1–2. UNIVAC 9200/9200 II/9300/9300 II Systems Processor Block Diagram

They are controlled and coordinated by means of internally stored programs which are derived from a standard instruction repertoire. The function of each of the components is as follows:

■ Storage Unit	– Comprises registers used in the processing and storage of all instructions and data that is to be, or has been, processed.
■ Data Register (D Reg)	– Provides temporary storage for data that is currently being processed. All transfer of data between the processor or peripheral units and the storage unit is accomplished through the data register.
■ Temporary Data Storage Register (B Reg)	– Provides additional temporary storage for data currently being processed, and also for various special codes required for process control and addressing.
■ Adder Input Network (ACX)	– Gates, modifies, and generates data involved in processing program data or special control codes.
■ Adder and Parity Generator	– Performs various processes specified by the instructions, such as addition, subtraction, and the formation of AND- and OR- products. Also generates parity.
■ Storage (Memory) Address Buffer (MA)	– Addresses of data being processed.
■ Storage (Memory) Address Register (MA Reg)	– Stores the addresses of data currently being processed.
■ Function Register (F Reg)	– Stores the partial decoded function code specified in an instruction.
■ Function Decode Table	– Interprets instructions and generates the control signals required for the initiation of the required process.
■ Sequence Counters A and B (SA CTR, SB CTR)	– Controls sequencing of the various control signals required to complete a process specified by an instruction. Although outputs of both counters control the generating of function signals, sequence counter A is used as a first-stage input to the second-stage sequence counter B.
■ Function Control Flip-flops and Gating	– Contains all of the secondary storage and gating circuitry for generation of the control signals necessary to initiate the various subfunctions of a specified process.

1.2. SYSTEM CONFIGURATION

The UNIVAC 9200/9200 II/9300/9300 II Systems processor is housed in two separate cabinets. The larger of the two cabinets contains an operator's control panel, a printer, and the integrated logic circuit packages. The second cabinet is the electronics cabinet; it contains the storage (memory) circuitry and the power supplies and power distribution panel for the entire processor.

Table 1–1 lists the basic and optional processor equipment for each of the four systems (UNIVAC 9200, 9200 II, 9300, 9300 II). Figures 1–3, 1–4, 1–5, and 1–6 illustrate the system configurations for these systems.

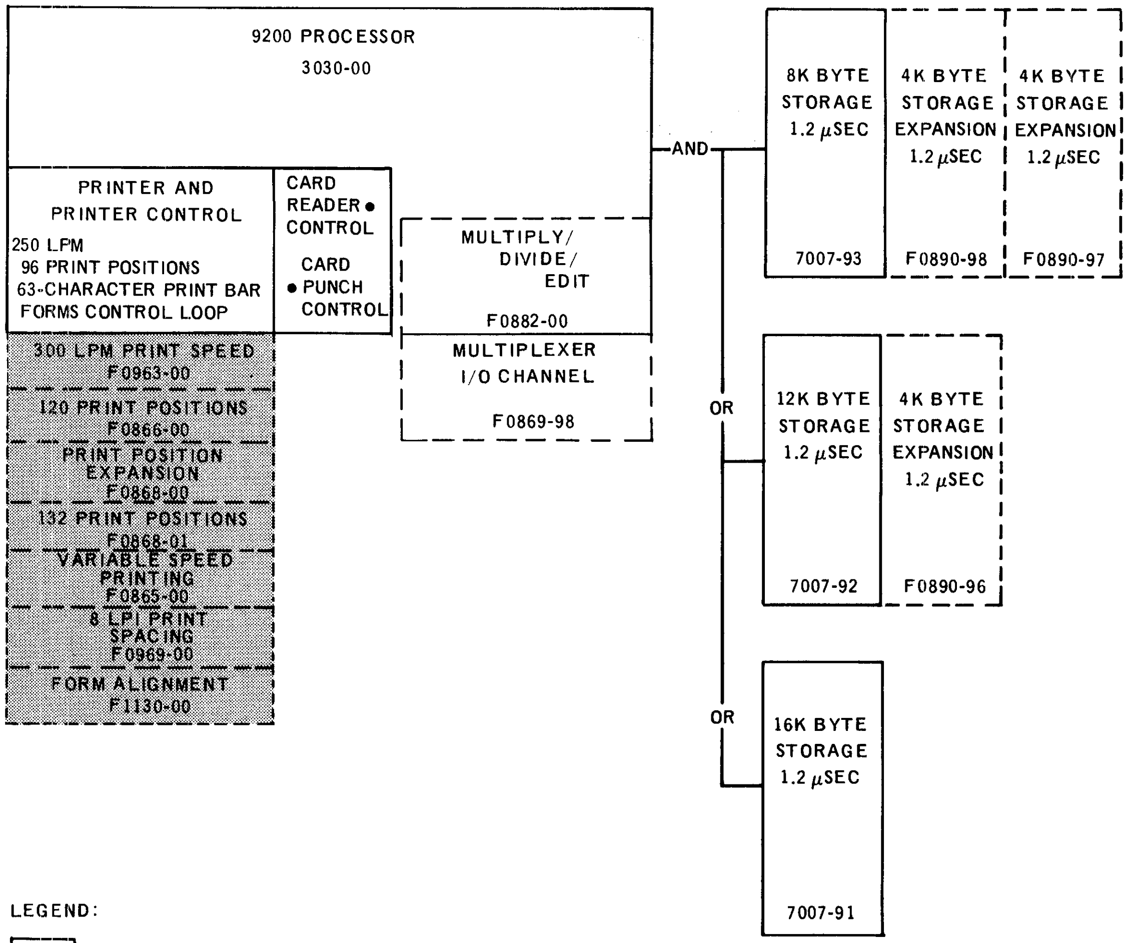
UNIVAC SYSTEMS PROCESSORS			
9200	9200 II	9300	9300 II
Printer Processor	Printer Processor	Printer Processor	Printer Processor
o Control	o Control	o Control	o Control
o Printer	o Printer	o Printer	o Printer
o Form control loop	o Form control loop	o Form control loop	o Form control loop
o 250 LPM bar printer 96 print positions 63 character print bar	o 250 LPM bar printer 96 print positions 63 character print bar	o 600 LPM bar printer 120 print positions 63 character print bar	o 600 LPM bar printer 120 print positions 63 character print bar
o 8K-byte storage Expandable to 16K bytes 1.2 μsec. cycle time	o 8K-byte storage Expandable to 32K bytes 1.2 μsec. cycle time	o 8K-byte storage Expandable to 32K bytes 0.6 μsec. cycle time	o 16K-byte storage Expandable to 32K bytes 0.6 μsec. cycle time
o Arithmetic Control Control for printer, punch, and reader	o Arithmetic Control Control for printer, punch, and reader	o Arithmetic Control Control for printer, punch, and reader	o Arithmetic/Control Control for printer, punch, and reader
* Multiplexer channel; accesses up to 8 subsystems or another processor	o Multiplexer channel; accesses up to 8 subsystems or another processor	* Multiplexer channel; accesses up to 8 subsystems or another processor	o Multiplexer channel; accesses up to 8 subsystems or another processor
* Multiply, divide, edit	* Multiply, divide, edit	o Multiply, divide, edit	o Multiply, divide, edit
	* Selector channel		o Selector channel
† 300 LPM print speed	† 300 LPM print speed		
† 120 print positions	† 120 print positions		
† Print position expansion	† Print position expansion	† Print position expansion	† Print position expansion
† 132 print positions	† 132 print positions		
† Variable speed printing	† Variable speed printing	† High speed numeric print	† High speed numeric print
† 8 LPI print spacing	† 8LPI print spacing	† 8 LPI print spacing	† 8 LPI print spacing
† Form alignment	† Form alignment		

LEGEND:

o Basic Equipment
* Processor Optional Features
† Printer Optional Features

Table 1–1. Configurations for UNIVAC 9200 and 9300 Systems Processors

UNIVAC 9200 SYSTEM PROCESSOR



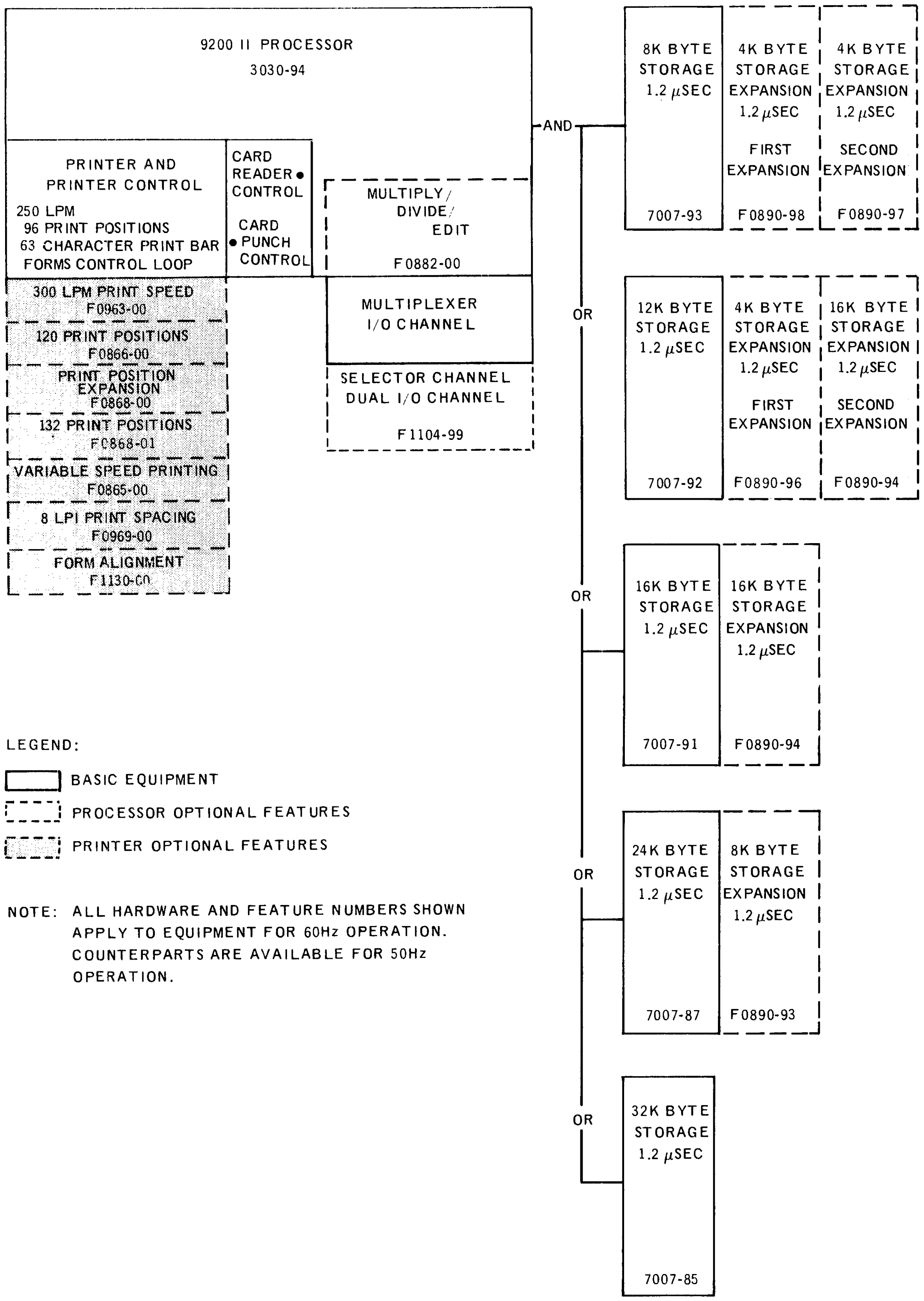
LEGEND:

☐	BASIC EQUIPMENT
⬚	PROCESSOR OPTIONAL FEATURES
▨	PRINTER OPTIONAL FEATURES

NOTE: ALL HARDWARE AND FEATURE NUMBERS SHOWN APPLY TO EQUIPMENT FOR 60Hz OPERATION. COUNTERPARTS ARE AVAILABLE FOR 50Hz OPERATION.

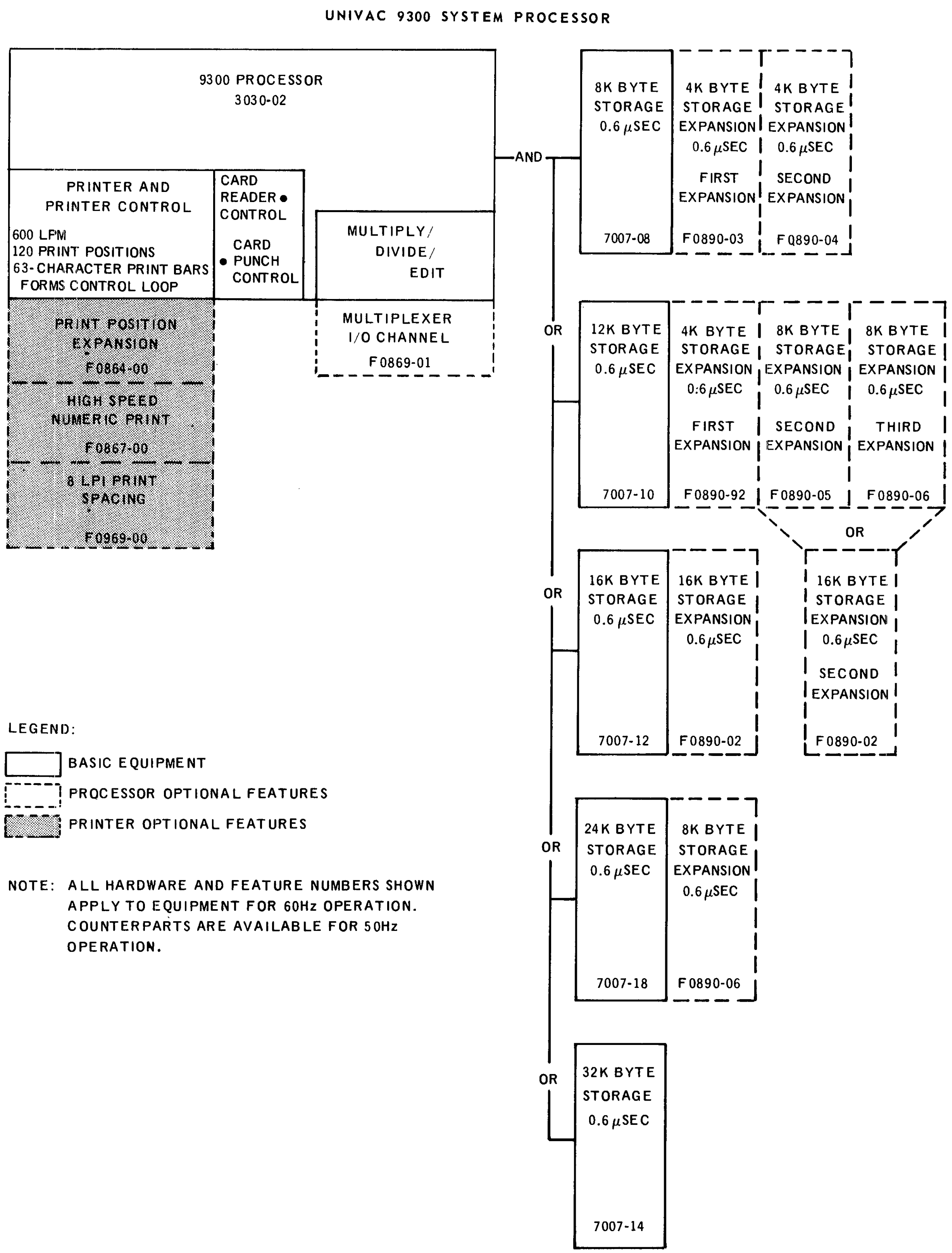
Figure 1–3. Configuration for UNIVAC 9200 System Processor

UNIVAC 9200 II SYSTEM PROCESSOR



NOTE: ALL HARDWARE AND FEATURE NUMBERS SHOWN APPLY TO EQUIPMENT FOR 60Hz OPERATION. COUNTERPARTS ARE AVAILABLE FOR 50Hz OPERATION.

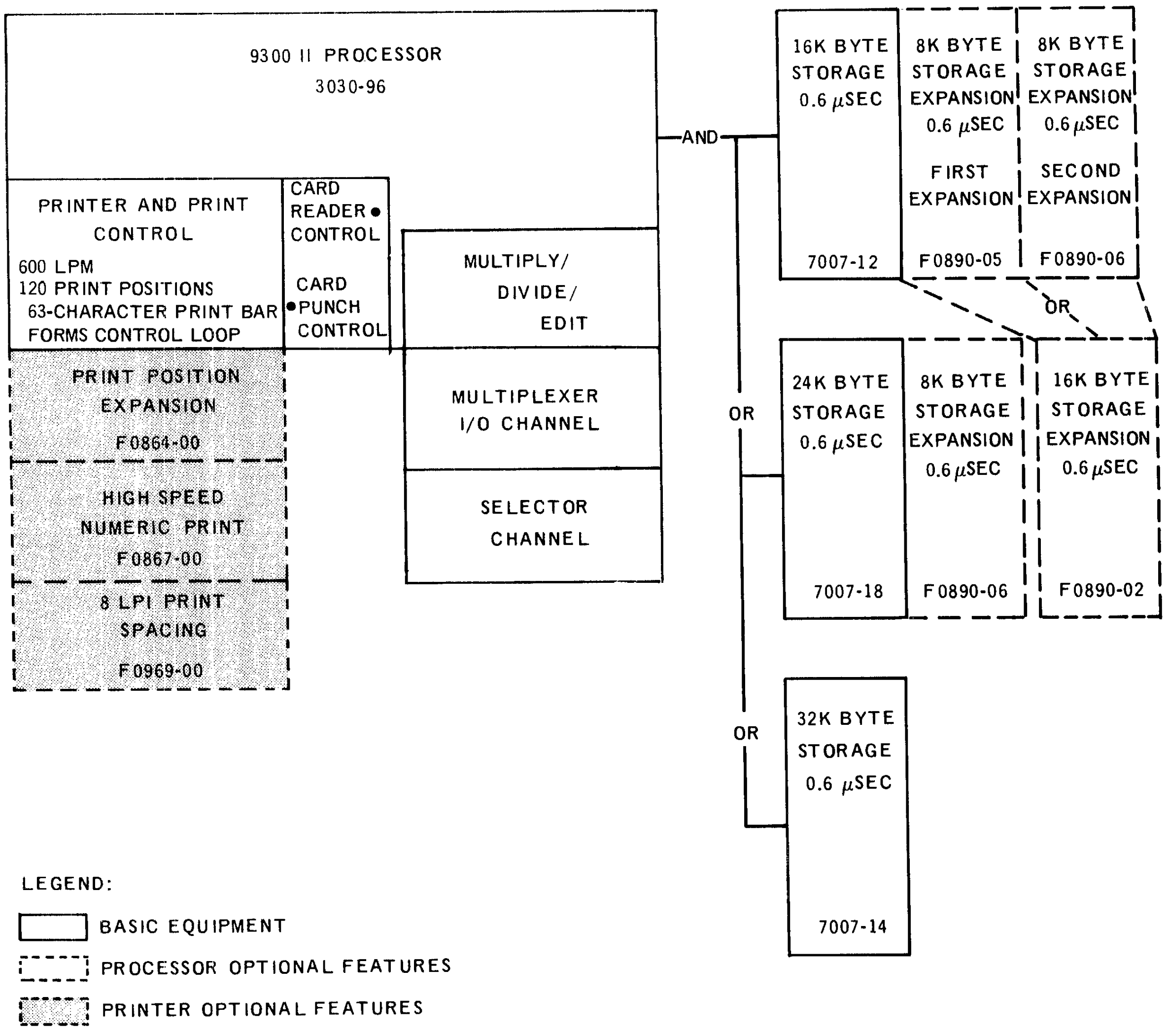
Figure 1–4. Configuration for UNIVAC 9200 II System Processor



UNIVAC 9300 SYSTEM PROCESSOR
9300 PROCESSOR 3030-02
PRINTER AND PRINTER CONTROL
600 LPM
120 PRINT POSITIONS
63-CHARACTER PRINT BARS
FORMS CONTROL LOOP
CARD READER ● CONTROL
CARD ● PUNCH CONTROL
MULTIPLY/ DIVIDE/ EDIT
MULTIPLEXER I/O CHANNEL F0869-01
PRINT POSITION EXPANSION F0864-00
HIGH SPEED NUMERIC PRINT F0867-00
8 LPI PRINT SPACING F0969-00
AND
OR
8K BYTE STORAGE 0.6 μSEC 7007-08
4K BYTE STORAGE EXPANSION 0.6 μSEC FIRST EXPANSION F0890-03
4K BYTE STORAGE EXPANSION 0.6 μSEC SECOND EXPANSION F0890-04
12K BYTE STORAGE 0.6 μSEC 7007-10
4K BYTE STORAGE EXPANSION 0.6 μSEC FIRST EXPANSION F0890-92
8K BYTE STORAGE EXPANSION 0.6 μSEC SECOND EXPANSION F0890-05
8K BYTE STORAGE EXPANSION 0.6 μSEC THIRD EXPANSION F0890-06
16K BYTE STORAGE 0.6 μSEC 7007-12
16K BYTE STORAGE EXPANSION 0.6 μSEC F0890-02
16K BYTE STORAGE EXPANSION 0.6 μSEC SECOND EXPANSION F0890-02
24K BYTE STORAGE 0.6 μSEC 7007-18
8K BYTE STORAGE EXPANSION 0.6 μSEC F0890-06
32K BYTE STORAGE 0.6 μSEC 7007-14
LEGEND:
BASIC EQUIPMENT
PROCESSOR OPTIONAL FEATURES
PRINTER OPTIONAL FEATURES
NOTE: ALL HARDWARE AND FEATURE NUMBERS SHOWN APPLY TO EQUIPMENT FOR 60Hz OPERATION. COUNTERPARTS ARE AVAILABLE FOR 50Hz OPERATION.

Figure 1–5. Configuration for UNIVAC 9300 System Processor

UNIVAC 9300 II SYSTEM PROCESSOR



9300 II PROCESSOR
3030-96
AND
16K BYTE STORAGE 0.6 μSEC 7007-12
8K BYTE STORAGE EXPANSION 0.6 μSEC FIRST EXPANSION F0890-05
8K BYTE STORAGE EXPANSION 0.6 μSEC SECOND EXPANSION F0890-06
OR
PRINTER AND PRINT CONTROL
600 LPM
120 PRINT POSITIONS
63-CHARACTER PRINT BAR
FORMS CONTROL LOOP
CARD READER ● CONTROL
CARD ● PUNCH CONTROL
MULTIPLY/ DIVIDE/ EDIT
MULTIPLEXER I/O CHANNEL
SELECTOR CHANNEL
PRINT POSITION EXPANSION F0864-00
HIGH SPEED NUMERIC PRINT F0867-00
8 LPI PRINT SPACING F0969-00
OR
24K BYTE STORAGE 0.6 μSEC 7007-18
8K BYTE STORAGE EXPANSION 0.6 μSEC F0890-06
16K BYTE STORAGE EXPANSION 0.6 μSEC F0890-02
OR
32K BYTE STORAGE 0.6 μSEC 7007-14
LEGEND:
BASIC EQUIPMENT
PROCESSOR OPTIONAL FEATURES
PRINTER OPTIONAL FEATURES

NOTE: ALL HARDWARE AND FEATURE NUMBERS SHOWN APPLY TO EQUIPMENT FOR 60Hz OPERATION. COUNTERPARTS ARE AVAILABLE FOR 50Hz OPERATION.

Figure 1–6. Configuration for UNIVAC 9300 II System Processor

1.3. COMPONENT DESCRIPTION

Components for the different systems are generally rather similar. For this reason, the descriptions which follow apply, with exceptions noted, regardless of the system in which the component is used.

1.3.1. Processor

The major portions of the UNIVAC 9200/9200 II/9300/9300 II Systems processor are the main storage, control, arithmetic, and input and output.

1.3.1.1. Main Storage

The main storage portions of the processor is a separate free standing unit connected to the printer processor cabinet. The storage elements are of the plated wire, cylindrical thin film type. The storage unit operates either in a read, write, or lockout mode at a cycle rate of 1.2 microseconds for the UNIVAC 9200 and 9200 II Systems processor; and at a rate of 0.6 microseconds for the UNIVAC 9300 and 9300 II Systems processor. Reading is nondestructive; that is, the data is not erased from storage by the read process.

The main storage is used to hold data received from input peripherals, results of processing, data to be distributed to output peripherals, programmed instructions, and control information.

The minimum basic storage is 8,192 bytes of nine bits each (eight data bits and one parity bit), except that the UNIVAC 9300 II Systems processor has a minimum of 16,384 bytes. The basic storage units may be expanded to 12,288, 16,384, 24,576, or 32,768 bytes except for the UNIVAC 9200 Systems processor, which is limited to 16,384 bytes.

1.3.1.2. Control

The control portion of the processor controls the sequence, interpretation, and execution of each instruction. The cycling of main storage is initiated by this section. All of the hardware aspects of interrupt handling, error checking, and protection are performed by the control section. The control section maintains the program address location counter and provides for the different processor modes of operation.

1.3.1.3. Arithmetic

The arithmetic portion of the processor performs data manipulations including binary and decimal arithmetic operations, and logical operations. The basic UNIVAC 9200 and 9200 II Systems processor is limited to machine addition only. Subtraction is performed by converting the subtrahend to its two's complement and adding. To multiply and divide, a suitable subroutine must be inserted into the user's program. The Multiply, Divide, and Edit capability (see 1.3.1.3.1) is available as an option.

1.3.1.3.1. Multiply, Divide, and Edit

The UNIVAC 9300 and 9300 II Systems processor has a multiply/divide/edit instruction set which includes decimal multiplication and division, and additional editing features. This capability is optional for the UNIVAC 9200 and 9200 II Systems processor.

1.3.1.3.2. Subtraction by Two's Complement Method

Because the fundamental arithmetic operation of the processor is addition, to perform subtraction it is necessary to add the two's complement of the subtrahend to the minuend. Let M and S be the absolute values of the minuend and subtrahend respectively. Then:

$$\left.\begin{matrix}(+M)-(+S)\\(+M)-(-S)\\(-M)-(+S)\\(-M)-(-S)\end{matrix}\right\}\text{is converted to}\left\{\begin{matrix}(+M)+(+S)'\\(+M)+(-S)'\\(-M)+(+S)'\\(-M)+(-S)'\end{matrix}\right.$$

where ()' denotes the two's complement of the original value.

By definition, the two's complement of a binary number (N) with n digit positions is:

2^n-N

Thus for eight-bit binary number (N), the two's complement is 2^8-N or, in binary notation, 100000000−N.

The two's complement of the binary number 00111001, for example, is:

```
 100000000
-00111001
 ---------
 11000111
```

Note that actual subtraction is not required, since the two's complement can be obtained by inspection of the number. Each bit of the number is simply inverted, that is, a 1 is changed to a 0, and a 0 is changed to a 1; a 1 is then added to the least significant bit at right. Thus the

```
binary number  00111001
is inverted    11000110
1 is added     +      1
               --------
               11000111 = two's complement of 00111001,
```

which agrees with the result obtained above.

Example:

It is desired to subtract +58 from another number. This is done by adding the two's complement of 58. What is the value?

```
Binary equivalent  0011 1010
One's complement   1100 0101
                      +    1
                   ---------
Two's complement   1100 0110  This is the desired value.
```

Suppose 58 were to be subtracted from 17, where the representation of 17 is 0001 0001.

```
The addition is represented 0001 0001
                            1100 0110
                            ---------
                            1101 0111
```

The 1 in the most significant bit position indicates a negative value; therefore the two's complement is required. This is –(0010 1000+1) = –(0010 1001) which is –41.

1.3.1.4. Input/Output

The input/output portion of the processor, through the use of input/output instructions, provides the means of initiating the operation of all peripheral devices associated with the processor and of determining the status of each device. This portion of the processor also directs the transfer of data between main storage and the peripheral system. After control of the input/output function has been transferred to the control unit for a particular device, data transfer is performed concurrently with the processor functions.

1.3.1.4.1. Multiplexer Channel

The multiplexer channel (standard with the UNIVAC 9200 II and 9300 II Systems processor and optional with the UNIVAC 9200 and 9300 Systems processor) provides an interface for devices other than the basic peripheral devices, card punch, and card reader. The multiplexer channel accepts I/O instructions from the processor and sends I/O requests to the connected peripheral devices, one at a time. The multiplexer channel places the device address and all signals needed to ascertain the status of the device on the multiplexer channel output line (bus) to the device. The peripheral device responds with a byte of information containing its status. The multiplexer channel decodes this status byte and generates generates a condition code for the processor. If there is no traffic to be executed, or if the device is not at that time ready to handle traffic, the multiplexer channel tests the next peripheral device in a predetermined order; if there is no traffic to be executed, the sequence specified by the processor's program is executed.

The multiplexer channel is asynchronous; it depends on the processor and the control unit of the peripheral device for instructions. The channel generates the necessary sequences to respond to sequences initiated by the control unit, making use of the processor's flip-flop registers and arithmetic circuits on a time-sharing basis. This permits the multiplexer channel to work several peripheral devices in sequence by assigning the multiplexer channel interface to the first peripheral device long enough to transfer one or a few bytes of information. When the processor finishes its operation with the first peripheral device, it operates similarly with the other peripheral devices before returning to the first. The processor must test each peripheral device for availability before an information transfer can take place.

The maximum transfer rate for one control unit in burst mode is 98K bytes per second. To achieve this rate, the control unit must respond with the leading edge of Service In at the input to the channel within 1.3 microseconds following the trailing edge of Service Out at the output of the channel. If the control unit response time is greater than 1.3 microseconds the maximum transfer rate is reduced as follows:

- If greater than 1.3 microseconds but less than 2.45 microseconds, the maximum transfer rate is 87.7K bytes per second.

- If greater than 2.45 microseconds but less than 3.65 microseconds, the maximum transfer rate is 79.5K bytes per second.

The maximum transfer rate for one control unit in multiplex mode is 75.8K bytes per second. To achieve this rate the control unit must respond with the leading edge of Address In at the input to the channel within 350 nanoseconds following the leading edge of Select Out at the output of the channel. This time must include all delays caused by propagation of the Select signal, cable lengths, control unit receivers and drivers, and control unit logic. Ordinarily, this can be achieved only by the highest priority control unit. The typical maximum rate for a control unit, other than the highest priority control unit is 72.5K bytes per second. To maintain these two transfer rates (75.8K and 72.5K bytes per second) the control unit must have the Request In signal active at the channel 1.2 microseconds following the trailing edge of the Service Out signal at the output of the channel.

For a control unit that does not meet the Request In requirements, the maximum transfer rate drops to 57.2K bytes per second. This rate uses the average processor latency time of 8.5 microseconds. Again to achieve this transfer rate of 57.2 bytes per second, the control unit must respond with the leading edge of the Request In signal at the input to the channel within 4.5 microseconds following the trailing edge of Service Out signal at the output of the channel.

The maximum transfer rate, for more than one control unit in multiplex mode when connected in such a configuration as to keep Request In signal active continuously is 72.5K bytes per second. This rate considers the typical response times from the control units, the Select Out propagation time through higher priority control units and the fact that the channel interface bus cable can be connected to eight control units.

1.3.1.4.2. Selector Channel

The selector channel, standard with UNIVAC 9300 II Systems processor and optional with UNIVAC 9200 II System processor, provides additional I/O capability for eight subsystems such as a disc subsystem. Maximum transfer rate of the selector channel is 350K bytes per second, including command chaining when operating in burst mode.

The selector channel operates with a higher priority than the multiplexer channel, thereby permitting the selector channel to gain access to the storage area at any time.

1.3.2. Printer

A bar printer with a 63-character print bar is included in the main cabinet of the processor. For the basic UNIVAC 9200/9200 II equipment, printing speed is 250 lines per minute; there are 96 print positions per line. For the UNIVAC 9300 and 9300 II equipment, printing speed is 600 lines per minute and there are 120 print positions. See Appendix A for additional type bars available for the printer.

Optional features available for the printer are described in the subsections as follows.

1.3.2.1. 300 LPM Print Speed

The 300 LPM print speed feature, optional with the UNIVAC 9200 and 9300 II Systems processor, increases the print speed of the printer from 250 LPM to 300 LPM. When used with variable speed printing feature, the variable speed is increased from 250/500 LPM to 300/600 LPM.

1.3.2.2. 120 Print Positions

The 120 print position feature, optional with the UNIVAC 9200 and 9200 II Systems processor and standard with the 9300 and 9300 II Systems processor, adds 24 additional print positions to the basic 96, to provide 120 print positions.

1.3.2.3. Print Position Expansion

The print position expansion feature, optional with the UNIVAC 9200, 9200 II, 9300, 9300 II Systems processors, permits a 120-position printer to be expanded up to 132 print positions. This option can be added after equipment installation.

1.3.2.4. 132 Print Positions

The 132 print position feature, optional only with the UNIVAC 9200 and 9200 II Systems processor, permits a 96-position printer to be expanded up to 132 print positions. This option can be adopted after equipment installation.

1.3.2.5. Variable Speed Printing

Variable speed printing, optional only with UNIVAC 9200 and 9200 II Systems processor, converts the 63 character, 250 LPM printer to a printer capable of printing a 48 character set at 250 LPM, and through program control automatically switches to 500 LPM if the line to be printed contains only characters of the 16 character numeric set. This allows automatic switching between 250 LPM and 500 LPM depending on characters to be printed on a line. After installation of the feature, 63 character and 48/16 character bars are interchangeable.

1.3.2.6. 8 LPI Print Spacing

Print spacing at eight line per inch, optional with UNIVAC 9200, 9200 II, 9300, 9300 II Systems processor, permits the operator to select vertical print spacing of either 6 or 8 lines per inch.

1.3.2.7. Form Alignment

Form alignment, optional with UNIVAC 9200 and 9200 II Systems printers, provides vernier tractors and a single turn phasing knob, which are standard with the UNIVAC 9300 and 9300 II Systems processor printers.

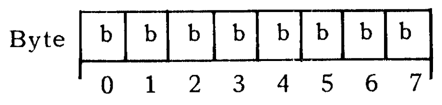
1.3.2.8. High Speed Numeric Print

High speed numeric print, optional only with the UNIVAC 9300 and 9300 II Systems processor, permits conversion of the 63 character 600 LPM printer portion of the processor to a 16 character, 1200 LPM numeric set. After installation of the feature, 63 character and 16 character bars are interchangeable.

1.4. DATA FORMATS AND CODES

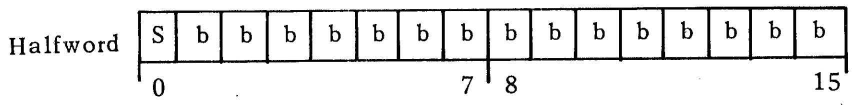
The data formats used in storage are as follows:

One byte consists of eight data bits numbered 0 to 7, left to right, as shown:



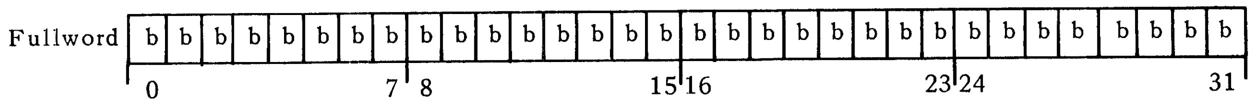
where b denotes a bit.

Halfword data format consist of two consecutive bytes, starting at an even address such as 0, 2, 4, the binary representation being within 16 bits and having the leftmost bit as the sign (S) bit of the data:



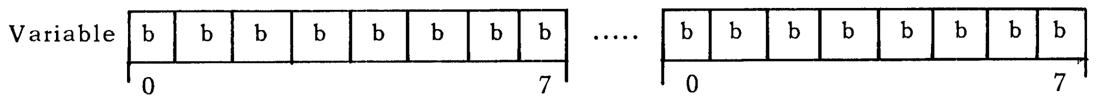
It is possible to store 256 different bit combinations in 2 bytes.

Fullword data formats consist of four consecutive bytes if the most significant byte address is divisible by four.

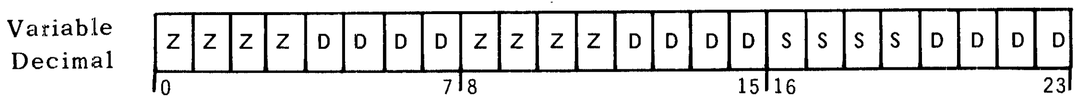

Fullword
b b b b b b b b b b b b b b b b b b b b b b b b b b b b b b b b
0
7
8
15
16
23
24
31

Instructions use 4 to 6 bytes and are restricted to the halfword boundary.

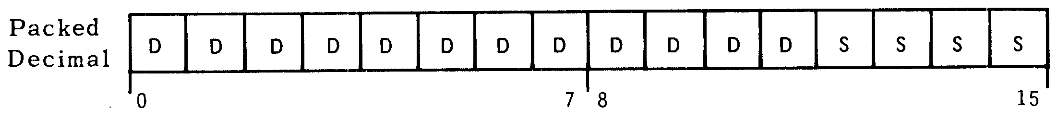
Variable data formats have a variable number of consecutive bytes:



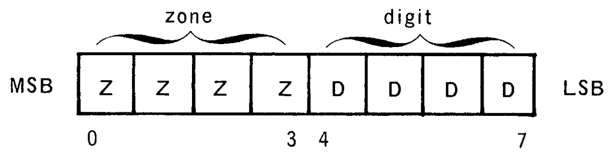
The data format of unpacked decimal representation is variable in length. A three digit decimal number, for example, is stored in three bytes as shown in the sketch, where ZZZZ = zone, DDDD = digit, SSSS = sign.



Packed decimal representation is also variable in length, A three digit decimal number is stored in a two byte packed decimal field when DDDD = digit, SSSS = sign.



Data can be represented in various forms by the programmer; but certain restrictions are imposed if the data is to be printed or processed arithmetically. The contents of a byte can be considered as a binary number, a decimal number, or two packed decimal numbers, an alphabetic or symbolic character, or logical information. A field used to represent a binary number uses all of the bit positions (except the sign bit) to contain the value. However, each byte in a field representing a decimal number, alphabetic character, or symbol is considered to be divided into zone and digit portions. The zone portion is the most significant four bits; the digit portion is the least significant four bits.



1.4.1. Binary Number Representation

Binary numbers in the registers make use of the most significant bit (MSB) as the sign indicator. A 0 in the most significant position indicates a positive number and a 1 indicates a negative number. It should be noted that since the MSB is the sign bit, a binary value of only ±32K may be expressed. Negative numbers are expressed by the two's complement value of the number.

1.4.2. Hexadecimal Representation

Hexadecimal numbers are those having a radix or base 16. There are 16 digits from 0 through F(15). A hexadecimal digit is used to denote a particular four-bit pattern in the zone or digit portion of a byte representing either a decimal number or an alphabetic or symbolic character. Hexadecimal digits and their decimal equivalents up to 65,535 are given in Table 1–2 along with an example of how to derive the decimal equivalent to a hexadecimal number.

HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
0	0	0	0	0	0	0	0
1000	4096	100	256	10	16	1	1
2000	8192	200	512	20	32	2	2
3000	12288	300	768	30	48	3	3
4000	16384	400	1,024	40	64	4	4
5000	20480	500	1,280	50	80	5	5
6000	24576	600	1,536	60	96	6	6
7000	28672	700	1,792	70	112	7	7
8000	32768	800	2,048	80	128	8	8
9000	36864	900	2,304	90	144	9	9
A000	40960	A00	2,560	A0	160	A	10
B000	45056	B00	2,816	B0	176	B	11
C000	49152	C00	3,072	C0	192	C	12
D000	53248	D00	3,328	D0	208	D	13
E000	57344	E00	3,584	E0	224	E	14
F000	61440	F00	3,840	F0	240	F	15

EXAMPLE:

Hexadecimal 39DB to Decimal

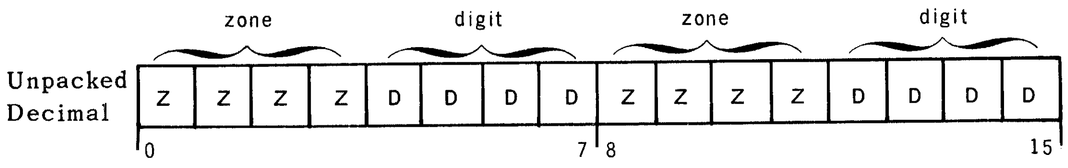
```
3000 = 12,288
 900 =  2,304
  D0 =    208
   B =     11
-------------
39DB = 14,811
```

Table 1–2. Hexadecimal – Decimal Conversion

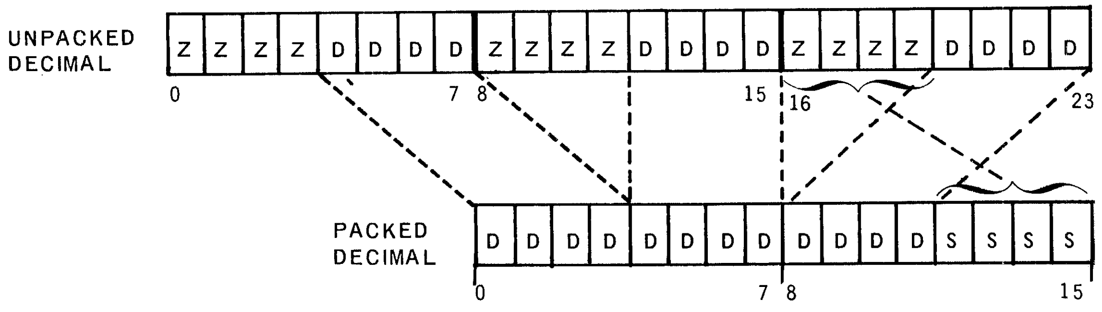
The expansion of Table 1–2, and tables for the powers of 2 and 16, are given in Appendix A.

1.4.3. Decimal Number Representation

Decimal numbers are represented either in unpacked form (one digit per byte) or in packed form (two digits per byte). In unpacked form, the byte is divided into zone and digit positions.



The zone portion usually contains a hexadecimal F (1111), which is ignored except in the least significant byte; the zone portion of the least significant byte is interpreted as the sign of the number. In packed form, digits are contained in both halves of a byte, except the least significant half byte of the field, which is interpreted as the sign of the decimal number.



UNPACKED DECIMAL
Z Z Z Z D D D D Z Z Z Z D D D D Z Z Z Z D D D D
0
7
8
15
16
23
PACKED DECIMAL
D D D D D D D D D D D D S S S S
0
7
8
15

The sign of the decimal number is represented by hexadecimal digits A through F. Any other bit configuration is an invalid sign code, which could produce unpredictable results.

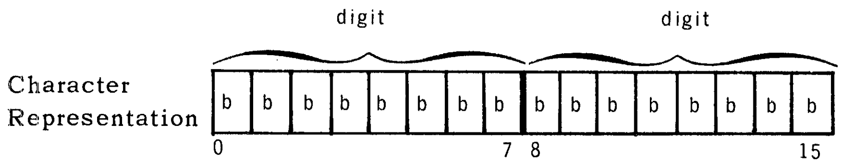
1.4.3.1. Sign Bits

The binary values of the sign bits are interpreted as follows:

0000 through 1000	positive number
1001	negative number
1010	positive number – generated by instructions in ASCII mode
1011	negative number – generated by instructions in ASCII mode
1100	positive number – generated by instructions in EBCDIC mode
1101	negative number – generated by instructions in EBCDIC mode
1110	positive number
1111	positive number

1.4.4. Character Representation

An alphabetic symbolic character uses eight bits of a byte and has no sign.



1.4.5. Parity Verification

The UNIVAC 9200/9200 II/9300/9300 II Systems processor uses odd parity for error checking. All data read from storage is given a parity check.

1.4.6. OP Codes

The OP code designations are expressed by two hexadecimal digits, in the 8-bit code shown, as follows:

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

1.4.7. Logical Information

Logical information consists of alphabetic or numeric character codes. This information is used in operations such as comparison, translation, editing, bit setting, and bit testing. Logical information is handled as fixed- or variable-length data and is processed from left to right, one byte at a time. Units of logical information may contain up to 256 bytes.

2. PROCESSOR UNIT

2.1. MAIN STORAGE

The maximum number of addressable locations (after installation of the expansion option) is 16,384 for the UNIVAC 9200 System processor, and 32,768 for the UNIVAC 9200 II, 9300, 9300 II System processors. Each addressable location consists of a byte of information containing eight data bits. The first 260 locations are called the privileged or low order storage. Of these, the first 96 byte locations are associated with either the processor program state control or the I/O (executive) program state control and are restricted. The next 32 byte locations are reserved for the buffer control words. The final 132 byte locations of low order storage are reserved for print storage. Main storage byte locations are used to store the following kinds of information: data received from input peripheral units, results of processed data, data to be distributed to output peripherals, programmed instructions, and control information. Figure 2–1 shows the main storage organization, identifying the byte locations by giving typical addresses in both decimal and (in parentheses) hexadecimal notation.

Each byte contains nine bits, which comprise eight information bits and one parity bit. The bit combination, as interpreted by the various instructions, represents alphabetic, decimal, binary, or logical data.

2.1.1. Privileged and Low Order Storage

The first 260 bytes of storage, called the low order storage, are reserved for specific operation information. Information stored in this area is accessible as needed during the execution of the functions. The first 128 locations, consisting of 64 locations called privileged storage and 64 locations for buffer control words, are reserved for program control functions. The remaining 132 locations are reserved for print storage, storing the data required for each line of print for the bar printer.

The privileged storage area contains 16 general registers, eight processor program registers, and eight input/output program registers, each holding two bytes. The processor can reference either of these two sets of eight registers in low order storage. This capability greatly reduces the interrupt handling time because the normal interrupt processing required, when only one set of general registers is provided, is unnecessary. Also the time required for storing the contents of the processor program registers and loading instructions and data into the registers is greatly reduced by the use of two sets of registers.

Control of the program access to either set of registers are functions of the respective program state control words in low order storage. The processor program state control word is at byte location 0 (0) through 3 (03) and input/output program state control word is at location 16 (10) through 19 (13). It should be noted that when the processor is in the processor program state, the first 64 byte locations of the privileged storage cannot be accessed because they are restricted to machine use only. An attempt to access these locations when the processor is in any state but the input/output program state results in a processor-abnormal condition.

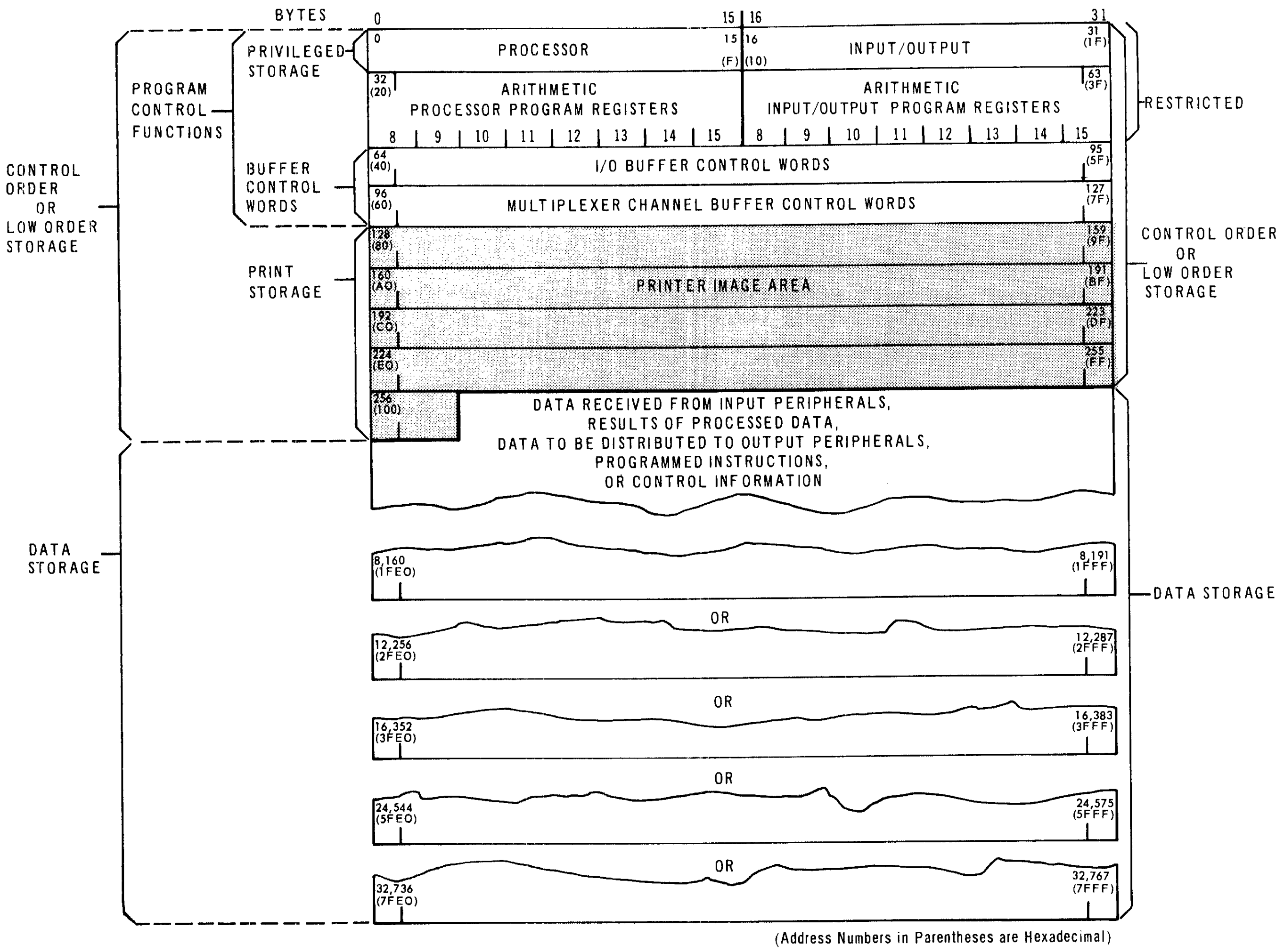


Figure 2-1. Main Storage Organization

2.1.2. Storage Boundaries

Bytes may be addressed separately or in groups. Two consecutive bytes constitute a halfword when the more significant byte is found at an even number address. Instructions occupy 4 or 6 bytes and are restricted to halfword boundaries. For instructions which imply a fixed length field of more than one byte, the operands are again restricted to halfword boundaries. If the field lengths are variable, there are no boundary restrictions. All fixed and variable length storage operands (operands being the address portion of an instruction) are addressed at the most significant byte position, regardless of whether the instruction is processed from left to right or from right to left. For variable size operands, the length is specified as a binary number which is one less than the actual count, thus a length count of one is expressed by all zeros in the length field.

Bytes in any storage are consecutively numbered starting at 0, up to a number which is less by one than the storage size, according to the following:

Maximum Address Number	Available in			
	9200	9200 II	9300	9300 II
8,191	X	X	X	
12,287	X	X	X	
16,383	X	X	X	X
24,575		X	X	X
32,767		X	X	X

2.1.3. Parity Checking

Each byte of storage has, in addition to the eight information bits, a ninth, or parity bit. The parity bit is set to 0 or 1 as may be necessary to make an odd number of 1 bits in the byte. This parity check bit is checked as data are read out of storage and is regenerated when data are altered as a result of processing. Parity is also generated when data are introduced into the processor from peripherals or other sources that do not already have a parity bit included in their data format. In other parts of this description of the processor's operation, the parity bit will not be mentioned except as it may be related to a possible error condition. If a processor parity error occurs, the processor will stop, and all input/output devices in use will continue to normal termination. A parity error that is detected during the transfer of data to and from the attached peripherals is considered a peripheral parity error and is discussed in 2.3.8.

2.1.4. Interrupts

An interrupt is the stopping of a process (in the processor) in such a way that it can be resumed; the reason for an interrupt is to afford opportunity for the required execution of another process external to the interrupted process but related to it. Interrupts are inhibited if processing is in the input/output state or if an inhibit interrupt is specified by programmed instruction. An interrupt can occur as a result of any of the following conditions:

- normal completion of an input or output function
- input or output function ended due to abnormal condition
- processing of a supervisor request call (SRC) instruction
- interval time (normally one second) exceeded.

An interrupt from a peripheral device can occur only if priority is granted to the device requesting the interrupt. When the request is granted, the following steps occur:

(1) The interrupt request is stored in appropriate indicators.

(2) The status code for the device is loaded into location 66 (42) and the device number is loaded into location 67 (43).

(3) At end of the instruction in process, the program state is set to the input/output mode.

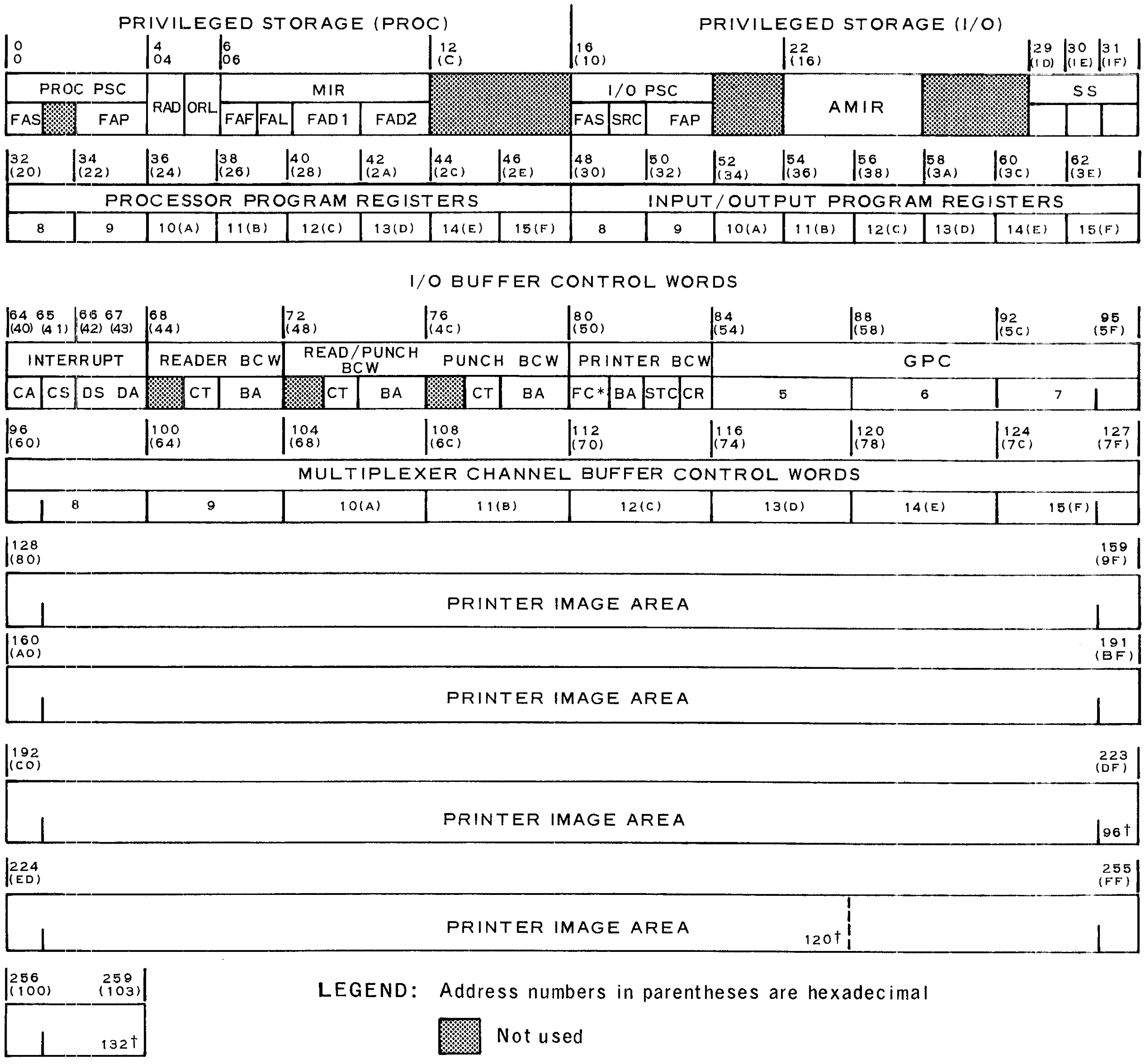
2.2. PROCESSOR CONTROL

The low-order storage area, locations 0 through 259 (103), is used for temporary storage of data required for processing an instruction and for the instruction and for the instruction itself. This storage is divided into two major parts (processor, and input/output) and several minor parts, as illustrated in Figure 2–2. The processor and the input/output portions of privileged storage are used separately; at any given time the one used depends on the Program State Control. The fields within the low-order storage and the adjacent buffer control word areas are as follows:

PROC PSC	Processor Program State Control
FAS	Fixed Address Status (Condition Code)
FAP	Fixed Address Pointer (address of first byte of next instruction)
RAD	Restricted Alter/Display
ORL	Operator Request Location
MIR	Machine Instruction Register
FAF	Fixed Address Function (Operation Code)
FAL	Operand Length/Register Address/Immediate Data
FAD1	Address of Operand 1
FAD2	Address of Operand 2
I/O PSC	Input/Output Program State Control
FAS	Fixed Address Status
SRC	Supervisor Request Call
FAP	Fixed Address Pointer
AMIR	Restart Instruction Register
SS	Special Status Storage for General Purpose Channel
INTERRUPT	Interrupt Entry for General Purpose Channel
CA	Device Address (Selector Channel Only)
CS	Channel Status (Selector Channel Only)
DS	Device Status
DA	Device Address
BCW	Buffer Control Word (Reader, Read/Punch, Punch)
CT	Count (number of columns to be read or punched)
BA	Base Address (address of least significant byte of input or output data)

PRINTER BCW

FC	Forms Control
BA	Base Address
STC	Starting Code
CR	Code Register
GPC	General Purpose Channel (reserved for multiplexer channel buffer control words in UNIVAC 9300/9300 II Systems processor.)


PRIVILEGED STORAGE (PROC)
PRIVILEGED STORAGE (I/O)
0 0 | 4 04 | 6 06 | 12 (C) | 16 (10) | 22 (16) | 29 (1D) | 30 (1E) | 31 (1F)
PROC PSC
FAS
FAP
RAD
ORL
MIR
FAF
FAL
FAD1
FAD2
I/O PSC
FAS
SRC
FAP
AMIR
SS
32 (20) | 34 (22) | 36 (24) | 38 (26) | 40 (28) | 42 (2A) | 44 (2C) | 46 (2E) | 48 (30) | 50 (32) | 52 (34) | 54 (36) | 56 (38) | 58 (3A) | 60 (3C) | 62 (3E)
PROCESSOR PROGRAM REGISTERS
INPUT/OUTPUT PROGRAM REGISTERS
8 | 9 | 10(A) | 11(B) | 12(C) | 13(D) | 14(E) | 15(F) | 8 | 9 | 10(A) | 11(B) | 12(C) | 13(D) | 14(E) | 15(F)
I/O BUFFER CONTROL WORDS
64 65 (40) (41) | 66 67 (42) (43) | 68 (44) | 72 (48) | 76 (4C) | 80 (50) | 84 (54) | 88 (58) | 92 (5C) | 95 (5F)
INTERRUPT
CA CS DS DA
READER BCW
CT BA
READ/PUNCH BCW
CT BA
PUNCH BCW
CT BA
PRINTER BCW
FC* BA STC CR
GPC
5 | 6 | 7
96 (60) | 100 (64) | 104 (68) | 108 (6C) | 112 (70) | 116 (74) | 120 (78) | 124 (7C) | 127 (7F)
MULTIPLEXER CHANNEL BUFFER CONTROL WORDS
8 | 9 | 10(A) | 11(B) | 12(C) | 13(D) | 14(E) | 15(F)
128 (80) | 159 (9F)
PRINTER IMAGE AREA
160 (A0) | 191 (BF)
PRINTER IMAGE AREA
192 (C0) | 223 (DF)
PRINTER IMAGE AREA
96†
224 (E0) | 255 (FF)
PRINTER IMAGE AREA
120†
256 (100) | 259 (103)
132†

LEGEND: Address numbers in parentheses are hexadecimal

▨ Not used

† Print positions

* FC = Bits 4567
0001 space one line
0010 space two lines
1xxx paper loop control

Figure 2–2. Organization of First 260 Bytes of Storage

2.2.1. Program State Control

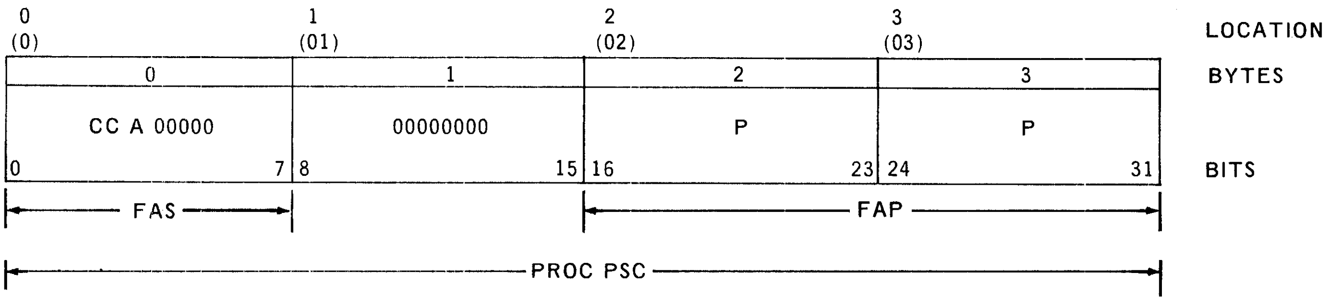
Program state control is a means of changing from one program state (processor or input/output) to the other. The state control determines which portion of privileged storage (processor or input/output) is used for processing instructions. The addressing of privileged storage under processor state control results in an error stop. Privileged storage may, however, be addressed under input/output state control without an error stop. Input/output devices may read out data from privileged storage without an error stop, but only during initial load may an input/output device write into privileged storage.

The processor and input/output states differ significantly in the following respects:

Processor State	Input/Output State
Processor portion of privileged storage used for processing instruction.	Input/output portion of privileged storage used for processing instructions.
Selected only by special instruction.	Selected by conditions such as an interrupt from a peripheral unit or general clear.
Interrupt inhibited only by program instruction.	Interrupt always inhibited.
Accessing privileged storage by an instruction generates processor abnormal (address error) condition.	Privileged storage can be accessed by an instruction with no (privileged storage) address error.

2.2.1.1. Processor Program State Control Word (Bytes 0–3)

The processor program state control (PPSC) word has the following organization:

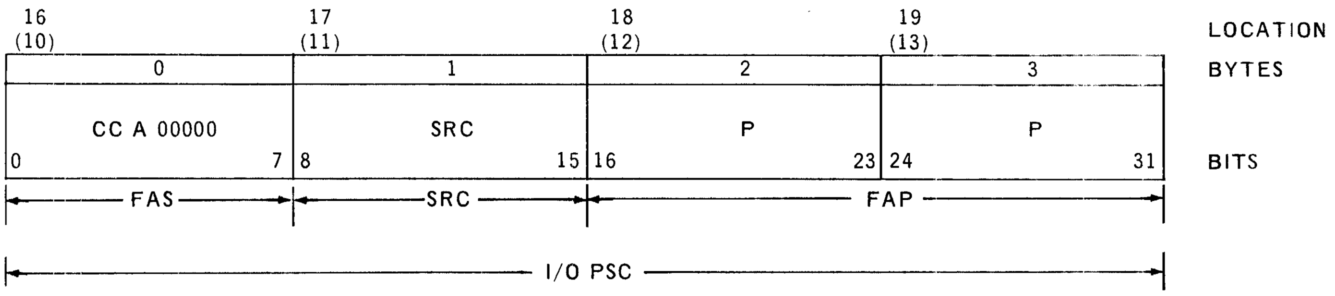


where: CC = condition code
A = ASCII control bit
1 for ASCII
0 for EBCDIC
0 = unspecified
P = address of first byte of next instruction

The user's program is normally executed in the processor program state. In this state, an interrupt is never inhibited except by program instruction.

2.2.1.2. Input/Output Program State Control Word (Bytes 16–19)

The input/output program state control (I/O PSC) word has the following organization:



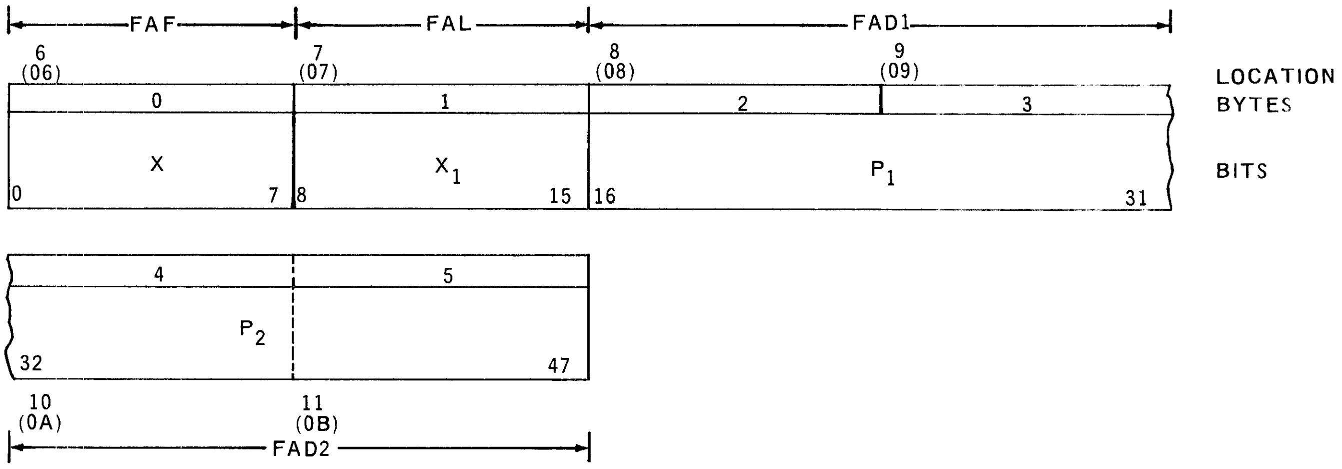
where: CC = condition
A = ASCII control bit
1 for ASCII
0 for EBCDIC
0 = unspecified
SRC = supervisor request call byte
P = address of first byte of next instruction

2.2.2. Restricted Alter/Display and Operator Request (Bytes 4, 5)

The restricted alter/display (RAD) location and the operator request location (ORL) are storage areas in low order storage used for entering information through the console of the computer. This function is performed in connection with certain software routines.

2.2.3. Machine Instruction Register (Bytes 6–11)

The machine instruction register (MIR) is loaded with the first instruction to be executed when a program is loaded; thereafter, the register will contain the current instruction in the program.



where: X = operation code

X_1 = operation length/register address/immediate data

P_1 = address of Operand 1

P_2 = address of Operand 2

2.2.4. Restart Instruction Register (Bytes 22–25)

The restart instruction register (AMIR) is used to store a branch instruction. It is the first instruction executed when a restart procedure is initiated by the operation of the first the CLEAR switch and then the START switch at the control console.

LOCATION	22 (16)	23 (17)	24 (18)	25 (19)
BYTES	0	1	2	3
	P_1	P_2	P_3	P_4
BITS	0 7	8 15	16 23	24 31

where: P_1 = present program address

P_2 = register of next instruction in logical sequence

$\left.\begin{matrix} P_3 \\ P_4 \end{matrix}\right\}$ = address to which instruction branches

2.2.5. Special Status (Bytes 29–31)

This area of storage is reserved for use by the Univac customer engineer. See 2.3.8.

2.2.6. Processor Program Register (Bytes 32–47) and Input/Output Program Registers (Bytes 48–63)

A separate group of eight registers is reserved in storage for each program state. The registers for any particular program state can be accessed only when the system is in the corresponding program state. The addresses of the registers are 1000_2 through 1111_2. Each register is two bytes long.

2.2.7. Device Status (Byte 66) and Device Address (Byte 67)

The device status (DS) and device address (DA) locations are storage areas in lower memory to receive device status (condition) and address fields during input/output operations from the peripheral devices.

LOCATION	64 (40)	65 (41)	66 (42)	67 (43)
BYTES	0	1	2	3
	0000	0000	P	P_1
BITS	0 7	8 15	16 23	24 31
	← CA →	← CS →	← DS →	← DA →

where: 0 = for selector channel use only, otherwise unspecified

P = device status

P_1 = device address

Location 64 (40) and 65 (41) are used to store the device address and channel status respectively when interrupt occurs for devices operating exclusively by way of the selector channel.

2.2.8. Buffer Control Words (Bytes 68–127)

Each buffer control word is described in the section dealing with the appropriate input/output operation.

2.2.8.1. Multiplexer Subchannel BCW

Each subchannel requires a four-byte buffer control word (BCW) in main storage. The BCW's contain data counts, data addresses, and control bits. Eleven BCW's are reserved for shared channels (byte locations 84 (54) – 127 (7F). The maximum number of BCW's (11) is greater than the maximum number of control units allowed (8) because some devices require more than one BCW. Sixteen additional BCW's (byte location 512 (40) – 575 (4F)) are for nonshared subchannel devices.

2.2.8.2. Device Control Subchannel Numbering

Control units for multiplexer channel peripherals are assigned an address or group of addresses. The channel recognizes two device address formats, one for shared subchannels and one for nonshared subchannels. Devices that share a control unit and cannot transfer data concurrently may share a subchannel. A subchannel may be shared by as many as eight devices. A shared control unit with more than eight devices would require at least two subchannels.

Devices that share subchannels have addresses with the following binary format:

1ssssbbb

where: ssss = the subchannel number

bbb = a particular device number

Devices operating through nonshared subchannels have addresses with the following binary format:

00xxssss

where: ssss = the subchannel address

xx = device number

2.2.8.3. Buffer Control Word Location

For devices with the addresses shown above, the location of the most significant byte (BCW 00) of subchannel n is storage location $64_{10} + 4N$, where N = ssss (see 2.3.2).

2.2.8.4. Additional Nonshared Subchannel Device Addresses

An additional set of addresses for devices operating through nonshared subchannels has been provided in the form:

0100ssss

For devices with such an address, the location of the most significant byte of the associated BCW is 512_{10} + 4n, where n equals number of subchannels.

2.2.9. Printer Image Area (Bytes 128–259)

The printer image area is a buffer area used in output operations to the systems printer. This buffer area is an interface between the processor and printer. All print data must be temporarily stored in the printer image area until accepted and printed by the printer. The number of print positions used, 96, 120, or 132, depends on the number of characters per line on the particular print bar installed in the printer. The printer image area must not be read from, or written to, during a print buffer scan.

2.2.10. Printer Control

The printer prints first and then advances paper. To allow the maximum amount of time to prepare the next line of data and to store the data in the specified print area, an interrupt is generated before the paper advance operation is completed. Thus, the functions overlap since the next execute input/output function (XIOF) instruction can be issued before the paper advance is completed for the last print instruction. If the interrupt were not generated until after the paper advance, one cycle of the print bar would be skipped after double spacing. Printing starts when the print bar is at its extreme position, either left or right. Printing for the UNIVAC 9200/9200 II Systems processor printer requires one complete cycle of bar movement, back and forth. An advance of as many as two lines can then be made without missing a print bar cycle. For the UNIVAC 9300/9300 II Systems processor printer, printing requires 1/2 cycle of bar movement, that is, one complete movement in either direction.

2.2.10.1. Printer Instructions

For the UNIVAC 9200/9200 II Systems processor, there are two valid print instructions: print and control. The print bar selection modifies these codes and is effective only if the bar print option has been included as part of the active system. Any other codes cause invalid operations. A print instruction may be given with or without paper advance. A control instruction is used for paper feeding without printing.

For the UNIVAC 9300/9300 II Systems processor, there are also two valid instructions: print and advance paper, and advance paper only. Any other codes cause invalid operations. In both cases the amount of paper spaced is controlled by the value placed in the forms control portion of the printer buffer control word; this may indicate paper advanced 0, 1, or 2 lines, or paper advanced under forms loop control. The selection of the 16 character print code is effective only if this feature has been included as part of the printer.

For the UNIVAC 9200/9200 II/9300/9300 II Systems processor, processing continues during all I/O instructions. If the H bit (see Execute I/O Function that follows) is set to one, all interrupts from the device are inhibited. The TIO instruction is then used to determine the status of the device.

The Execute I/O Function (XIOF) instruction follows:

0	1	2		3		BYTES
OP CODE A4	DA 00000011	B1	0000	BNOH	00X1	
0 7	8 15	16 19	20 23	24 27	28 31	BITS

where: X = 0 for a print instruction

X = 1 for a control instruction

B = 0 standard 63-character bar

B = 1 optional 48-character bar (9200, 9200 II only)

B = 1 optional 16-character bar (9300, 9300 II only)

N = 1 print numeric if 48 or 16 character bar is activated

H = 1 inhibit interrupt

NOTE: In a control XIOF, B and N are not significant.

On a system that has a printer with less than 132 print positions per line, data can be stored in the positions of the print image area for which there are no print hammers (locations 224 through 259 for a 96-position printer; or locations 248 through 259 for 120-position printer). Such data is not altered by, nor does it affect, the operation of the printer.

2.2.10.2. Printer Buffer Control Word

The buffer control word for the printer contains the following data:

0	1	2	3	BYTES
FC 0000LXXX	BA	STC	CR	
0 7	8 15	16 19	20 31	BITS

where: BA = base address

STC = starting code

CR = code register

FC = forms control

BA, STC, and CR are under complete hardware control. If they are inadvertently changed by a program, printer control will probably be lost.

The forms control byte is loaded by the program once a Test I/O (TIO) or an interrupt determines that it is permissible. The forms control byte is not changed by the execution of a printer function. The four arrangements of bits, and their interpretations, follow:

L	X	X	X	
0	0	0	0	No line spacing
0	0	0	1	Space 1 line
0	0	1	0	Space 2 lines
1	X	X	X	Select any of 7 paper loop channel controls by matching holes in the paper loop to the 1-bits in the X position.

There are two paper loop conventions:

X	X	X	
1	1	1	for Home Paper
0	0	1	for Forms Overflow

Under paper loop control, if a hole combination is sought that is not punched on the tape, a runaway paper condition results.

2.2.10.3. Issue and Execute

''Issue'' refers to the decoding of an XIOF by the processor and the forwarding of the command information to the printer control. At the time of issue, the condition code (CC) is generated and made available to the program. ''Execute'' refers to response by the printer control to the command information forwarded by the processor. In some instances, ''execute'' may follow ''issue'' by a considerable period of time.

2.2.10.4. Status Register

The print control area of storage contains a status register which stores the various status indications until they are transferred to main storage by a test input/output (TIO) instruction or by an interrupt request acceptance by the processor. When an XIOF is in progress, the setting of any bit in the status register will terminate the operation and generate an interrupt request.

There are two types of error conditions. Type I errors set the status register directly when they occur. Type II error indications are stored in an intermediate error storage when they occur. The next time an XIOF is executed, these indications are transferred to the status register.

Type I errors are the following:

- Bar Check

  Bar check error occurs after an XIOF is executed, but before printing begins.

- Memory Overload

  Memory overload occurs during printing.

- Parity

  Parity error occurs after an XIOF is executed, but before paper advances.

- Abnormal

  Abnormal conditions can occur any time.

Type II errors are the following:

- Paper Low
- Forms Overflow
- Paper Runaway

Type II errors occur during paper advance.

When an offline error occurs, the status register is not set, but the printer appears to be busy to the processor. Any order in progress when the offline (OFF-LN) switch is pressed will be allowed to continue to completion.

2.2.10.5. Interrupt Requests

Interrupt requests occur at the following times:

- End of print before associated paper feed is started.
- Immediately following an accepted paper feed order before paper advancing has begun, unless a previously initiated paper feed order is in progress. In the latter case, the interrupt is delayed until the previously initiated paper feed order has been completed.
- Upon abortion of an order due to detection of low paper, forms overflow, or paper runaway conditions occurring as a result of a preceding order.
- Upon termination of an operation due to any other error condition.

2.2.10.6. Printer Status Byte

The status byte, byte 66 (42), contains information pertaining to the result of the last issued order or the next to last issued order. The status indications are as follows:

All zeros	No indicator set; function performed as specified.
Bit 7 set to 1	Paper is low, as a result of paper spacing. Until the paper condition is corrected, this indication will occur for each XIOF. Low paper is indicated when the bottom edge of the last form is 15 1/3 ± 1/3 inches from print line.
Bit 6 set to 1	Forms overflow. 001 sensed at paper loop station during single or double spacing. Forms overflow indication is set even if spacing does not stop on the 001 channel punch. Passing over the punch is sufficient.

Bit 5 set to 1	Interrupt request pending. This status bit is set only if the TIO function clears a pending interrupt before interrupt request is accepted. This status bit does not indicate an error.
Bit 4 set to 1	Instruction is not consistent with bar switch setting.
Bit 3 set to 1	Data parity or control parity error on last XIOF instruction. Printer stops immediately.
Bit 2 set to 1	Storage overload occurred on last XIOF instruction. Printer stops immediately. Paper is not advanced.
Bit 1 set to 1	Paper runaway – forms control lost. Further orders will not be accepted without operator intervention, since printer operation is abnormal.
Bit 0 set to 1	Abnormal or not ready.

Paper low, forms overflow, and paper runaway interrupts are recognized following the normal interrupt request. The previous function will therefore be properly completed except in the case of paper runaway where paper has been spaced improperly. If another XIOF has been accepted, it will be aborted and an interrupt will be generated. If the next XIOF is not issued until after detection of the condition, the order will be accepted, then aborted and an interrupt request will be generated. Any error that happens before paper is advanced will prevent the paper from advancing.

2.2.11. Input/Output Storage

Several sets of storage locations are assigned to each input/output unit. Each set, called a buffer control word, consists of four storage bytes and is used for storing data storage addresses, character counts, and other details of each input/output function.

Input/output control requires the following software steps:

(1) Load the proper BCW with information required by the control unit if the unit is not busy.

(2) Issue an input/output instruction which specifies the device address and the function to be performed.

(3) Check the condition code setting to determine if the instruction is accepted.

(4) Test the status of the device when the operation is completed (completion is usually indicated by the generation of an interrupt) to determine if the operation is successful.

Processing continues during the execution of all I/O instructions except where the device is connected to a selector channel (see 2.4.2). If the inhibit interrupt (H) bit is set, all interrupts for the device are inhibited. The Test I/O instruction is then used to determine the device status. An I/O interrupt can be made only at the end of a program instruction execution in the Processor Program State Control.

At the end of each instruction execution, the peripheral interrupt request line is examined. If an interrupt request is present, interrupt is granted, control is shifted to the I/O program state control, and the device address and device status are stored in fixed locations in memory by the hardware.

A BCW should not be altered during the execution of an input/output operation on the peripheral device to which it is assigned. To do so can cause unpredictable results.

2.2.12. Operator-Initiated Functions

Operator-initiated functions requiring control are the load cycle, and the alternate execute and staticize cycle. The function of each execute (instruction) cycle is given in 3.3.

2.2.12.1. Load Cycle

In the load cycle, the program instructions are transferred from a peripheral device to the storage unit. This function is controlled by special load instructions transferred directly from a peripheral unit to the privileged area of storage. When all program instruction is loaded, the operation of the processor is stopped by a programmed HALT instruction and an appropriate indication is displayed on the operator's control panel.

2.2.12.2. Alternate Execute and Staticize Cycle

The execution of the program consists of alternate execute and staticize cycles, which continue to occur (in the RUN mode) until stopped by a programmed halt, an error condition, or operator intervention. When the program operation is started, the first (program) execute cycle occurs, and the process specified by the instruction in the privileged storage is accomplished. At the completion of the specified process, the processor automatically initiates the first staticize cycle. The staticize cycle has three basic functions:

(1) to increment the instruction address stored in the FAP portion of privileged memory,

(2) to store the instruction in the CIR (Current Instruction Register) portion of privileged memory after indexing (when required) operand addresses,

(3) to store the function code of the instruction being staticized in the function register.

During the staticize cycle, the next sequential instruction is brought from storage and decoded. At the completion of the staticize cycle, the processor automatically initiates the execute cycle for the instruction just staticized. This alternating staticize execute operation continues until again stopped by a programmed halt, an error condition, or an operator intervention.

2.3. MULTIPLEXER CHANNEL CONTROL

Peripheral units connected to the processor by way of the multiplexer channel have specific control requirements. These requirements are described as follows.

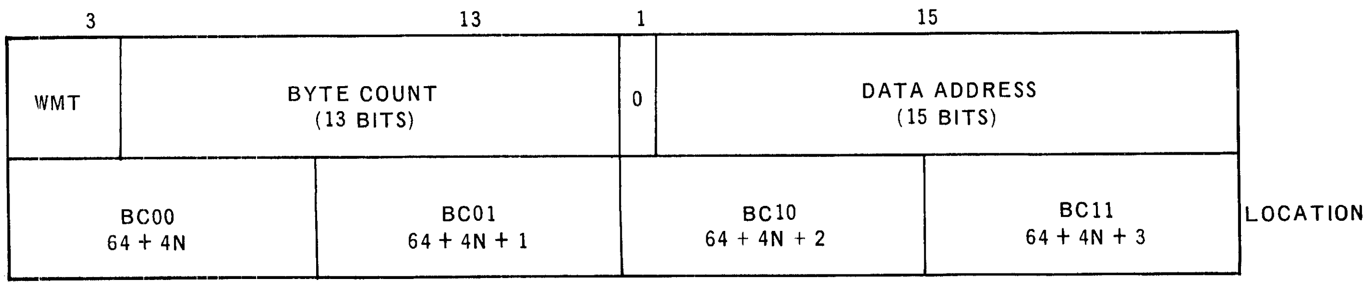
2.3.1. Multiplexer Channel Instructions

When execute or test I/O instructions are issued to devices other than the basic peripherals (device address 1, 2, or 3), the channel will attempt to execute the initial selection sequence or I/O command. The channel will reject the command if the addressed device is offline or does not exist. This will produce condition code 11; see 2.3.4.

2.3.2. Multiplexer Channel Buffer Control Word

When a subchannel is used, the proper BCW (see 2.2.8.1) must be loaded with the correct initial conditions before issuing an Execute I/O order to any subchannel. The multiplexer channel may also use buffer control words allotted to the basic I/O units even if the basic units are not present.

When a control unit initiates a sequence in order to request or present data or to present a status byte, the control unit presents a device address along with appropriate control signals. This address is placed in the multiplexer channel's device address register, where it is used to determine the location of the proper buffer control word. The action taken by the channel depends upon the contents of this location. The normal BCW format is as follows:



where:

W = Data Direction Bit

W = 1 for write (output) or "buffered" control operations

W = 0 for read (input) operations

M = Addressing Mode Bit

M = 0 for forward addressing sequence.

M = 1 for backward addressing sequence.

T = Termination Bit

If T = 1, no data will be transferred, the BCW will not be modified by the channel and the Terminate response will be given to data request.

The channel will set T = 1 after the transfer of a byte of data causes the byte count to go to zero. The channel will not reset the T bit to 0.

Byte Count: This field is decremented by the channel when a byte of data is transferred. An initial count of zero gives a block length of 8,192 bytes if T = 0.

A control unit may terminate an operation before the count becomes zero. Upon termination, the byte count field indicates the difference between the initial byte count and the number of bytes actually transferred.

Data Address: This field is fetched by the channel and used as the address for the current byte of data. The address is modified in the BCW under control of the M bit in preparation for the next byte. Upon termination, this field indicates where the next byte would have gone to or come from had the operation continued.

The W and M bits and the I/O command initiated through the subchannel must agree.

2.3.3. Multiplexer Channel Status Byte

At the time of initial selection during an execute I/O or test I/O instruction and also at the end of I/O operations, peripheral units present a status byte having the following format:

Bit 0 – Attention

1 – Status modifier

2 – Control unit end

3 – Busy

4 – Channel end

5 – Device end

6 – Unit check

7 – Exception

Normal initial selection status is not stored, but causes a condition code to be stored. The status byte is stored by a Test I/O instruction in a location specified by the program. When the channel is allowed to interrupt the program, the status byte is stored in location 66 (42).

2.3.4. Condition Code

When an execute I/O or test I/O instruction is issued through the multiplexer channel, the result of the operation is summarized in the condition code placed in the appropriate program state control area.

CC	EXECUTE I/O	TEST I/O
00	Command Accepted. No Channel Error. Status Byte equals 0000XX00.	Device Available. No Channel Error. Status Byte equals 00000000.
01	Command Rejected. No Channel Error. Status Byte does not equal 0X010000, or 0000XX00.	Nonzero Status Stored and cleared. No Channel Error. Status Byte does not equal 0X0X0000.
10	Device, Control Unit or Channel busy. Command Rejected. No Channel Error. Status Byte equals 0X010000.	Device, Control Unit, or Channel busy. No Channel Error. Status Byte equals 0X010000.
11	Device, Control Unit, or Channel not operational. Select in* or channel-detected error.	Device, Control Unit, or Channel not operational. Select in* or channel-detected error.

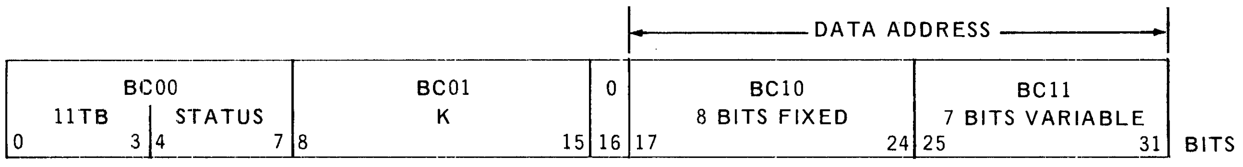
(X denotes that it has no significance)

''Select in'' is the signal sent by the multiplexer channel to the peripheral device to be selected.

On rare occasions, the channel may be momentarily busy, having been committed to a request in sequence, just as the I/O instruction was being staticized. Also, a programming error, for example, addressing a nonexistent device, can generate condition code 11.

2.3.5. Alternate BCW Format

An alternate buffer control word format (LT BCW format) is provided so that the system can handle devices that transfer a continuous stream of data at relatively slow speeds, such as communications line terminals for remote inquiry stations and computer-to-computer transmissions. The main difference between this and the normal buffer control word (see 2.3.2) is the fixed-length wraparound buffer addressing sequence, which can be considered as a limited form of data chaining. This operation is defined by the program, not by any device characteristic. The LT BCW format is specified when the W and M bits are both 1's, which would specify ''Write Backwards'' in the normal format. The LT BCW format follows:



T = Terminate Bit (position 2): If the terminate bit is a 1, no data will be transferred, the data address will not be modified, and the multiplexer channel will give the Terminate response to data requests. The channel will set the T bit to 1 when wraparound error occurs (see Buffer End Bit). The channel will not erase a T bit.

B = Buffer End Bit (position 3): When the address is modified to an integral multiple of 64, a carry from bit position 5 of the data address may be generated. In such case, the channel sets the B bit to 1 and generates an LT Summary Interrupt request. The B bit indicates to the program that a 64-byte buffer segment has ended. The program is expected to reset the B bit to zero when that buffer segment is again ready for use by the channel. If the channel finds a B bit set to 1 in the BCW when the end-of-buffer segment occurs again, the multiplexer channel sets the T bit to 1 so that the data will not be overlaid. This is the wraparound error situation. The channel will not reset a B bit; it must be removed by the program.

Data Address: This field contains the address of the next data byte to be transferred. The address modification in the LT format is always $A + 1 \rightarrow A$ (Mod 128). This sequence, with the B bit, gives the effect of alternating the use of two adjacent 64-byte buffer areas.

Status Field: If a device operating in the LT mode initiates a sequence to indicate the status, bits 4 to 7 of the status byte are transmitted by means of an OR function into this field. If the processor allows the interrupt, the entire status byte is also placed in the interrupt entry area and the LT summary interrupt request bit is set. If the interrupt is not allowed, the LT summary interrupt request bit is reset.

K Field (Address trap): If a device operating in the LT mode attempts to indicate the status, and bits 4 and 5 in the status field of the BCW were previously both 0, the eight least significant bits in the data address are transferred to the K field. If either bit 4 or bit 5 in the BCW was previously 1, the transfer does not occur.

2.3.5.1. Data Direction Control

No control bits for data direction are provided in the LT format. In this format, odd numbered devices are assumed to be executing input operations and even numbered devices are assumed to be executing output operations. The data direction is controlled by the least significant bit of the device address, which was transmitted at the beginning of the control unit initiated sequence.

2.3.6. Polling

The multiplexer channel will probe the interface every few microseconds, if the following three conditions exist:

- A printer scan is not in progress.
- A multiplexer channel is not involved in a previously initiated sequence.
- The processor has not staticized an I/O instruction.

Buffered devices, and any other devices that can wait indefinitely to transfer data, will not generate service requests when attached to this channel. (This is an option for all control units.) These devices can make a preemptive response when the channel polls, but cannot force the channel to poll and interrupt a print scan.

2.3.6.1. Priority of Interrupt

If the data to be transmitted by any device by way of the multiplexer channel would overload the data transfer storage capacity, the XIOF is rejected, the data late bit is set in the device status byte, and the unit check bit is set in the multiplexer channel status byte. If this condition exists, peripheral equipment will have requests processed in the following order (high or low) of priority: card reader, card punch, multiplexer channel, and printer.

If the punch, channel, and printer were working and the reader caused an anticipated storage overload condition, the printer would be interrupted first and then the multiplexer channel. If the reader stopped, the multiplexer channel would be interrupted followed by the printer. No matter which of the four levels presents the signal that would cause storage overload, this priority of interrupt occurs.

2.3.7. Special Channel Instructions and Interrupts

Certain device addresses are recognized or generated by the multiplexer channel itself. These dummy device numbers provide for:

(1) Operator interrupt

(2) Alternate (LT) summary interrupt

(3) One-second interrupt

(4) Diagnostic and self-checking features

Device numbers with the binary format 100xxxxx are dummy device addresses. These addresses are arrested by the channel on execute I/O function or test I/O status instructions. They are generated when certain interrupts occur and under some diagnostic conditions.

These subchannels do not have BCW's except in special diagnostic operations.

2.3.7.1. Operator Interrupt

The dummy device address for operator interrupt is hexadecimal 80, or binary 10000000. When the operator request (OP REQ) switch on the control console is pressed, the setting of the DATA ENTRY switches is stored in location 5 (05) and the operator request flip-flop is set. When interrupts are allowed, the multiplexer channel will interrupt the program and store 1000 0000 (80) in location 67 (43) and 0000 1100 (OC) in location 66 (42). Once the operator interrupt request indicator is set, further operator requests are inhibited and the OP REQ indicator remains off until the program issues a release operator request command by an execute I/O function instruction. An execute I/O function instruction may also be used to inhibit an operator request. The operator interrupt may be reset by a test I/O status instruction. When the CHANNEL CLEAR or master CLEAR switch is pressed at the control console, the inhibit operator request bit is set to the inhibit state.

2.3.7.2. Alternate (LT) Summary Interrupt

The LT summary interrupt dummy device address is hexadecimal 88, or 1000 1000. When a device operating in the LT format reaches a buffer end, or when a request to present status by such a device is rejected, the LT summary interrupt bit will request an interrupt. When the interrupt is allowed, 1000 1000 (88) is stored in location 67 (43) and 0000 1100 (OC) is stored in location 66 (42). Instructions are provided to inhibit or permit this interrupt. CHANNEL CLEAR or master CLEAR switch, when pressed at the control console, sets the inhibit bit to the inhibit state. This interrupt may be cleared by a test I/O instruction.

2.3.7.3. One-Second Interrupt

The one-second interrupt dummy address is 1001 0000 (90). This instruction is provided to set a one-second delay which will cause an interrupt after it recovers. When the interrupt is allowed, 1001 0000 (90) is stored in location 67 (43) and 0000 1100 (0C) is stored in location 66 (42). The interrupt request may be cleared by Test I/O instruction or by pressing the CHANNEL CLEAR or master CLEAR switch on the control console. The delay time caused by this interrupt is 1 ±0.50 second. Addressing this dummy device while the delay is set but not recovered results in busy condition code, 10_2, and 0001 0000 (10) status on a test I/O instruction.

2.3.7.4. Summary of Special Channel Instructions

Dummy Hexadecimal Device Address	XIOF Function Code	Function
80	13*	Inhibit Operator Request and extinguish Operator Request indicator.
80	23*	Permit Operator Requests and light Operator Request indicator.
80	00 (TIO)	Test and reset Operator Request bit.
88	13*	Inhibit LT Summary Interrupts.
88	23*	Permit LT Summary Interrupts.
88	00 (TIO)	Test and Reset LT Summary Interrupts.
90	23	Set one-second delay.
90	00 (TIO)	Test and reset One-Second Interrupt.

* These instructions are executed without regard to the resulting condition code, unless a channel error occurs.

2.3.8. Channel Checking

The multiplexer channel has three error flip-flops. When one of them is set, the processor will come to a stop unless a dummy device address is involved or the maintenance switches for use only by the Univac customer engineer are set to the time-out test (010) mode. When the channel detects an error, the I/O interface conditions at the time of the error are stored along with the error flip-flop settings, and the channel's device address register in locations 29 (1D), 30 (1E), and 31 (1F) in the control storage area.

The three error flip-flops and the conditions under which they are set are described as follows.

2.3.8.1. Interface Error Flip-Flop

This flip-flop is set when:

- More than one of the following inbound control lines is activated:

  ADDRESS IN
  STATUS IN
  SERVICE IN
  SELECT IN

- An interface sequence has been initiated and not completed within 70 microseconds.
- A control unit has held OPERATIONAL-IN active for more than 500 milliseconds without transferring data or status (burst mode time check).

2.3.8.2. Device Address Parity

This flip-flop is set if even parity is detected when a control unit has signaled the presence of a device address on the input line. If this flip-flop is set, the parity error flip-flop will also be set.

2.3.8.3. Parity Error Flip-Flop

This flip-flop is set if:

- Even parity is detected for input data, status bytes, or device addresses.
- Even parity is detected when a BCW byte is brought in from memory. The central processor parity error flip-flop will also be set.
- A memory addressing error is generated by the channel. The central processor address error flip-flop will also be set.

Addressing errors are generated when:

– The channel attempts to address a location beyond the capacity of the given system.

– The channel attempts to transfer data to or from the privileged area and Initial Load is not set.

– The channel attempts to write data into locations 128 (80) to 255 (FF) while the Print flip-flop is set.

2.4. SELECTOR CHANNEL CONTROL

The selector channel is a high speed channel designed to perform the following functions:

- Transfer data to and from UNIVAC 9200 II and 9300 II Systems processor in burst mode.
- Store status in a reserved storage location called the Channel Status Word.
- Interrupt the processor when error ending or ending status is stored.
- Execute a command chain sequence for disc operation.
- Operate with eight subsystems, each having no more than 16 devices per subsystem.
- Operate as an independent transfer mechanism queueing with the processor at the storage access interface.

2.4.1. Channel Addressing

No device on the selector channel may be assigned the same address as a device on the multiplexer channel. Device numbers 000X 111X and 1111 Xbbb may not be assigned the multiplexer channel, since the buffer control words would overlap the selector channel address word and diagnostic interrupt locations. (X = bits ignored by control unit, b = data bit).

2.4.1.1. Selector Channel Address Format

Selector channel address is of the 0b1bbbbb format. Interpretation of the b (data) bits depends upon the I/O subsystem. The selector channel does not distinguish bits assigned to the control unit from bits assigned to any particular device attached to that control unit. The channel handles the device address as a single number. For example, 00100XXX is the device address of the simulated control unit of the selector channel.

2.4.1.2. Multiplexer Channel Address Format

Devices operating through nonshared subchannels have addresses with the binary format:

000XSSSS where SSSS is the subchannel address.

Devices sharing subchannels have addresses with the binary format:

1ssssbbb where ssss is subchannel number and 1ssssbbb is a particular device number.

Devices with Alternate BCW formats (LT form) may address additional BCW's with the binary address: 0100bbbb in the LT BCW format given in 2.3.5, where b is the subchannel address.

2.4.2. Processor Lock-Out

The processor will not execute any instructions from the time it issues an execute I/O command to the selector channel until the channel receives termination status.

2.4.3. Concurrent Operations

When a card reader and a card punch are attached to the processor, the control for these units has a higher priority than the selector channel. The printer has a lower priority than the selector channel. Starting the selector channel, while a previously initiated print order is being executed by the printer control, may cause printer overrun since the selector channel has only one byte of data buffering.

Previously initiated multiplexer channel operations will run concurrently with the selector channel. Overrun depends in this case on data rates. The selector channel normally requires two memory cycles to process each byte of data. During command chaining sequences, the selector channel requires up to 36 consecutive memory cycles derived from the multiplexer channel.

2.4.4. Command Chaining

Command chaining is initiated when the chaining flag is set to 1 in the channel command word and the channel receives a normal termination status byte containing device end status. Chaining will not occur if an attention unit check, a unit exception, or a control unit end bit is set.

If a status modifier and a device end status are presented, and if chaining is indicated, the next channel word following the one skipped will be executed.

The stored status information modifies the chaining operations as follows:

- Attention, unit check, or unit exception status terminate the chaining operation.
- Status modifier along with the device end word status causes the channel to skip the next adjoining channel command word specified by the command address. The channel takes the next adjoining one (the channel command word following the skipped channel word) and performs the operation indicated.
- Channel end status without device end status causes the channel to indicate command chaining to the device. Multidevice subsystems must use this indication to link the appropriate device end status to this channel end status.
- Device end status, or channel end status and device end status, cause the channel to read the next adjacent channel command word (CCW) specified by the command address.

If a transfer-in-channel command (see 2.4.7) is decoded during a chain operation, the channel does not transfer this command to the device. The channel reads another channel command word using the data address along with the transfer-in-channel command, instead of the command address to locate the channel word. The channel performs the operation indicated and continues to chain if directed. Test commands are invalid in command chaining inasmuch as a zero status returned will terminate the chain.

2.4.5. Execute I/O and Test I/O Instructions

When an execute I/O or a test I/O is issued by the processor to devices other than the basic peripherals (device address 0, 1, 2, or 3) the channel executes the initial selection sequence which transfers the command byte in the I/O instruction to the selected device. The resulting condition code will be the same as the one specified for the multiplexer channel. The condition code indicates acceptance or rejection of the first command if command chaining had been requested.

2.4.6. I/O Commands

Command codes transmitted by the selector channel during initial selection sequence are as follows:

Command Codes	Bit Position 0 1 2 3 4 5 6 7
TEST I/O	XXXX 0 0 0 0
SENSE	DDDD 0 1 0 0
WRITE (output)	DDDDDD 0 1
READ (input)	DDDDDD 1 0
READ BACKWARD	DDDD 1 1 0 0
CONTROL	DDDDDD 1 1
TRANSFER IN CHANNEL	XXXX 1 0 0 0

where:

X = Ignored by Control Units
D = Command Detail Bits

- Test I/O – The test I/O command is sent to the device, and the resulting status is stored in the general register, byte 0.
- Sense – A sense command is sent to the device, and the subchannel is set up to transfer sense bytes from the device to storage.
- Write – A write command is sent to the device, and the subchannel is set up to transfer data from storage to the device when the device requests service.
- Read – A read command is sent to the device, and the subchannel is set up to transfer data to storage when the device requests service.
- Read Backward – A read backward command is sent to the device, and the subchannel places the bytes in storage in descending order.
- Control – A control command is sent to the device, and the subchannel is set up to transfer data from storage to the device when the device requests data.
- Transfer in Channel – The next channel command word is fetched from the location designated by the address field of the channel command word specifying transfer-in-channel. The transfer-in-channel command does not initiate an I/O operation at the channel.

2.4.7. Transfer-In-Channel (TIC) Command

The TIC command causes replacement of the address of the command address word (CAW) with the address field of the channel command word (CCW). The TIC (Channel "Jump" instruction) is not transmitted to the interface. The command byte of the Execute I/O instruction must not specify TIC.

TIC Code = XXXX1000

This command permits a search to be reinitiated.

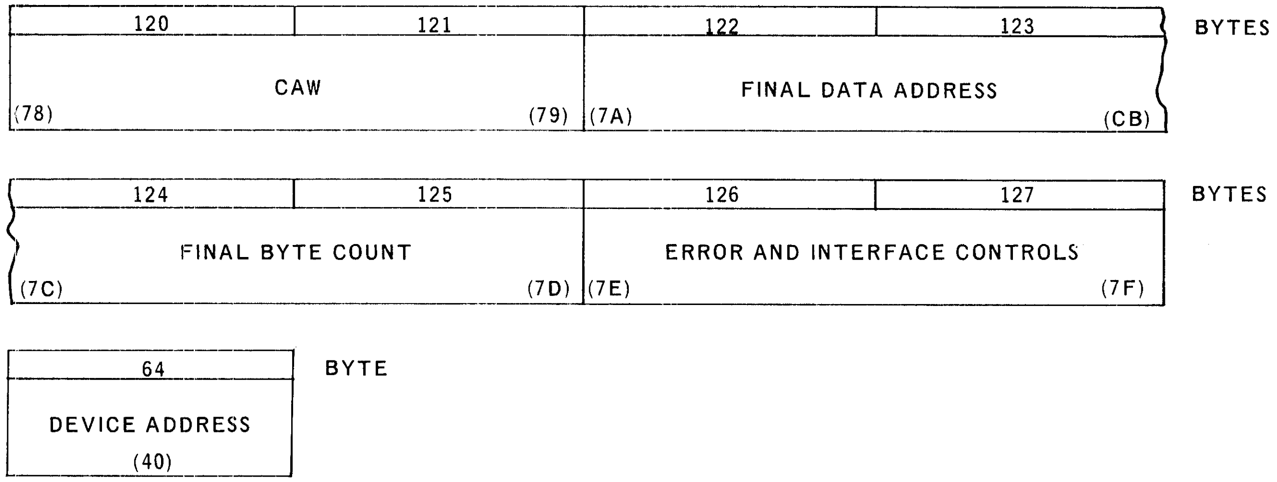
2.4.8. Condition Code (CC)

The Condition Code (CC) is stored in the appropriate Program State Control area; it summarizes the results of the XIOF (execute or test) instruction. CC for execute (XIOF) and test (TIOF) is as follows:

CC	XIOF	TIO
00	Command accepted	Device Available
01	Command rejected	Nonzero Status except 0X010000
10	Command rejected	Busy Device or Control Unit
11	Nonoperational, or channel error	Nonoperational, or channel error

2.4.9. Channel Address Word (CAW) and Diagnostic Interrupt

The CAW and Diagnostic Interrupt, a component of the multiplexer channel buffer control word (see Figure 2–2), is as shown:



(Numbers in parentheses are byte locations in hexadecimal code.)

The Command Address Word is set initially by the program to the address of the third byte of the first Channel Command Word (CCW) to be executed by the channel. The CAW is modified by the selector channel as it fetches the command(s). On termination of the channel operation, the CAW contains the address of the last command executed plus 8.

Storage byte locations 122 (7A) through 127 (7F) are the diagnostic interrupt locations. When an execute I/O instruction has been issued to the selector channel and the command on the chain of commands is terminated, the contents of the selector channel register, error indicators, and BUS IN are stored in fixed locations. This information is stored irrespective of the processor mode. If the processor is in processor mode, the channel then initiates a Status In sequence (interrupt).

The following byte locations are loaded by the selector channel on termination, normal or otherwise:

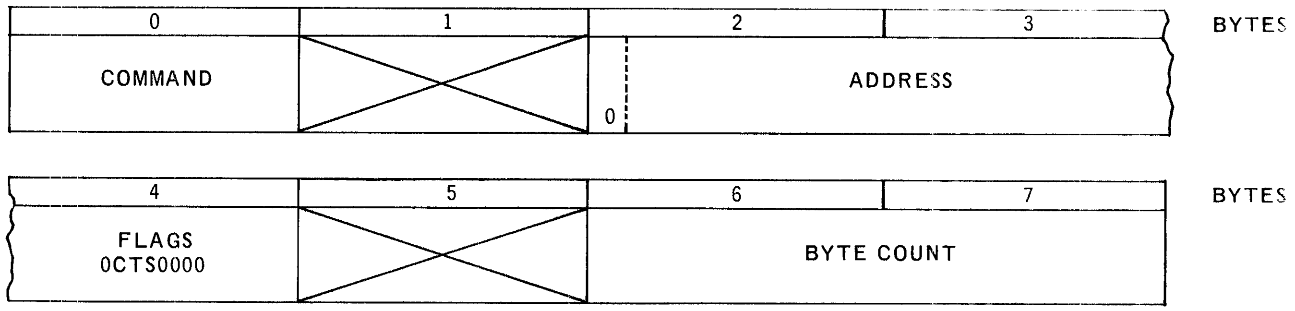
Byte Location	Contents
120 (78), 121 (79)	Incremented CAW
122 (7A), 123 (7B)	Next data address
124 (7C), 125 (7D)	Final byte count
126 (7E), 127 (7F)	Error FF's and interface controls
64 (40)	Device address

Bit contents for byte locations 126 (7E) and 127 (7F) are as follows:

Byte 126 (7E)		Byte 127 (7F)	
Bit	Contents	Bit	Contents
0	Interface error	0	STATUS IN
1	Address error	1	SERVICE OUT
2	Parity error	2	SERVICE IN
3	ADDRESS OUT	3	REQUEST IN
4	SELECT OUT	4	SUPPRESS OUT
5	OPERATIONAL IN	5	SELECT IN
6	ADDRESS IN	6	TERMINATE FF
7	COMMAND OUT	7	SJBO (always 1)

2.4.10. Channel Command Word (CCW)

The CCW is used during the execution of a start I/O instruction or command chain sequence to specify control information needed for operating the addressed I/O device. The CCW is located by the program on double word boundaries with the following format:



- The command byte of the first CCW is ignored by the channel, since the command is specified by the I/O instruction. The subsequent commands, if command chaining is indicated, are executed in order, provided no abnormal condition occurs (status is 0X00X100).

  The TIC operation may refer to the first CCW only if the program has previously loaded the command byte for reinitiation of a search.

- The address field is a 15-bit field that specifies the memory address of the first data byte, except in the TIC command, where it specifies the address of the first byte of the next CCW.

- Flag bits have the format OCTS 0000.

  (1) The command chaining bit (C) determines whether or not command chaining is indicated.

  Command chaining bit C

  C = 1 Command chaining is indicated. The command in the next CCW will be issued on normal completion of the current operation. The channel will wait for Device End, then initiate the reselect sequence.

  C = 0 No chaining will occur.

  (2) The suppress length indication bit (S) determines whether chaining will or will not continue when incorrect word length is indicated.

  S = 1 Incorrect length indicator in channel status word is suppressed and chaining will not be terminated.

  S = 0 Chaining will be terminated when the incorrect length indicator is set. This indicator is set when the control unit terminates data transfer before the byte count has been decremented to zero or when the control unit requests data after the byte count has been decremented to zero. Those control units usually terminated by the channel, such as tapes on a write command, suppress incorrect length when chaining is to be executed.

  (3) The terminate bit (T) when set inhibits data transfer.

  T = 1 No data will be transferred. The terminate response will be given to data requests and the terminate bit will be set in byte location 127 (7F). After the transfer of a byte of data, the channel will set the T flip-flop to cause the byte count to be decremented.

- The byte count field indicates the number of bytes of data to be transferred. It is not used during the TIC operation. An all-zeros field results in the maximum byte count, 65,536 bytes, if the T bit is not set.

2.4.11. Device Status

The control unit status byte is stored in a program specified location as test I/O instructions in location 66 (42) when the channel is allowed to interrupt the program. Interrupts are handled in the order of their occurrence except that interrupts from the multiplexer or the selector channel are handled in order of their priority.

Control units and devices generate an interrupt and present a status byte upon detection of any of the following conditions:

Bit (Byte 4 of CSW)	Data Condition Detected
0	Attention
1	Status modifier
2	Control unit end
3	Busy
4	Channel end
5	Device end
6	Unit check
7	Unit exception

- Attention – The attention bit is set by a device to indicate an abnormal condition other than those associated with the initiation, execution, or termination of an I/O operation.

- Status Modifier – This status bit is set along with the Busy status bit to distinguish between Control Unit Busy and Device Busy in multidevice subsystems. It is also set along with the Device End status bit to indicate special ending conditions for command chaining operations (Skip designation).

- Control Unit End – This status bit can be generated by a control unit when it has been freed for another operation. It can accompany Channel End, or Device End, or may occur at a time in between, if the control unit is interrogated by initiation of another command.

- Busy – This status bit is set when the device or control unit cannot execute the command because of a previous command in progress or a pending status condition.

- Channel End – This status bit is set when the channel is ready for a new operation. The time of occurrence of the Channel End is dependent upon the type of control unit.

- Device End – This status bit is set when the device is ready for a new operation. If an unusual condition is detected during the initiation of a command, error status is present without Device End status.

- Unit Check – This status bit is set when errors are detected during an operation. Unit check is a summary notification of a unit – associated malfunction. Detailed status is obtained by initiating a Sense command.

- Unit Exception – This status bit is set for abnormal conditions which may or may not be an error. Its meaning is applicable to the type of command and device involved in an operation.

The selector channel does not store the status byte presented during initial selection sequence of an execute I/O instruction, but rather, the channel examines the status and sets the appropriate condition code for that status which the program may examine. Other than 0000 XX00, these codes representing status are stacked as long as the program remains in the processor mode. To examine the status, the program must be in I/O mode at which time status is examined in sequence. The status is subsequently cleared by a Test I/O or a Status In-Request In sequence.

Upon termination of an I/O operation, the status from a control unit is either accepted or stacked; accepted if the program remains in the processor, stacked if the program is in the I/O mode. On Control Immediate commands, some control units present channel end status during the initial selection sequence and device end status at a later time. This would be handled in regular order. The program must keep track of normal status presentation in processor mode, that is, a check on location 67 (43) is made for either the selector channel device address or multiplexer channel device address or absence of either for supervisor request call (CRC).

2.4.12. Channel Status Word (CSW)

A double word is used to store interrupt status information for the selector channel in the low order storage with the following format:

0	1	2	3	BYTES
COMMAND CODE	0 ———— 0	COMMAND ADDRESS		

4	5	6	7
DEVICE STATUS	0 ———— 0	BYTE COUNT (16 BITS)	

- Command Code – Contents of output register.
- Command Address – Address information points to location of next CCW.
- Device Status – The control unit or device generates an interrupt and presents a status byte upon the detection of any of the status formats given in 2.4.11.
- Byte Count – A value showing the original CCW byte count, decreased by the number of bytes transferred.

Channel Status is stored in location 65 (41) only when interrupt occurs. It is not possible to obtain the channel status with a test I/O instruction.

CSW format is as follows: EL00 0000

where: E = any-error bit
L = incorrect length bit

2.4.13. Channel Checking

The selector channel has a number of error flip-flops. Whenever the channel detects an error, the I/O interface conditions at the time of the error are stored along with the error flip-flops in the diagnostic locations 122 (7A) through 127 (7F). The error bit will be set in the CSW if an interrupt occurred. An error occurring on the selector channel does not stop the processor.

Addressing errors are generated and the address bit is set in location 126 (7E) if:

- the channel attempts to address a location beyond the capacity of the given systems.
- the channel attempts to transfer data to or from the privileged area and Initial Load is not set.
- the print area is not protected by the channel.

Interface error, Device Address Parity error, and Parity error flip-flops are similar to the ones for the multiplexer channel with limitations and error conditions the same as explained in 2.3.8.

2.4.14. Special Selector Channel Instructions

An XIOF or Test I/O instruction issued to device address 33 (21) causes an interface time check error and triggers a selective reset sequence. Channel status data stored in location 126 (7E) becomes 152 (98), and data in location 127 (7F) contains data from location 17 (11).

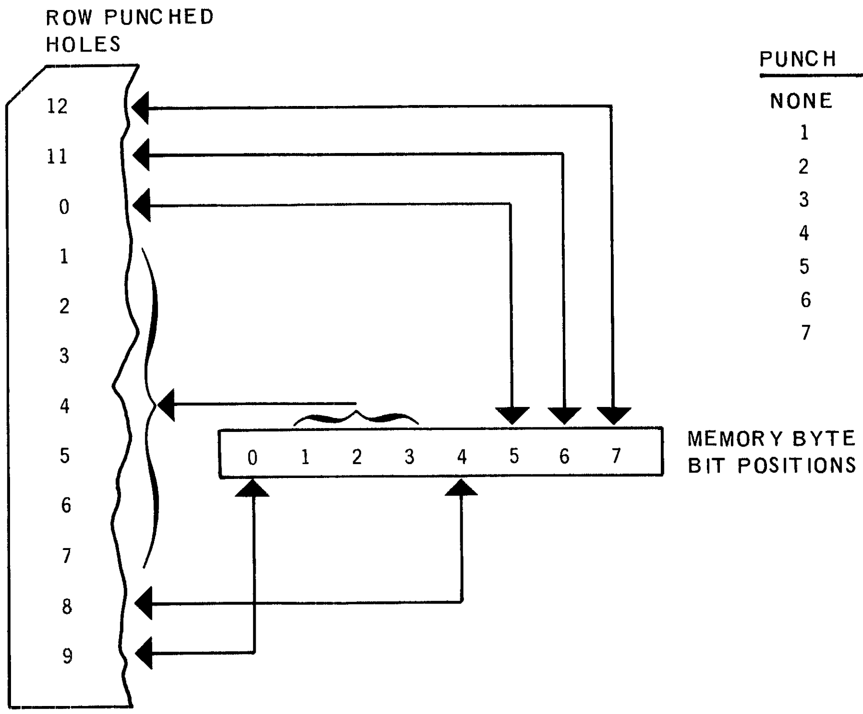
Device address in location 32 (20) is used to represent a dummy control unit. When XIOF or TIO is issued to this device address, normal termination will occur with Channel End and Device End included in the status byte. On read command, data in 211 (D3) is stored in the memory area specified by the CCW.

2.5. DATA TRANSLATION

There are 256 possible code combinations that can be represented by an 8-bit byte of memory. Card code translation is to or from compressed code. This code must be translated to the internal code for processing and then retranslated to the compressed code for punching.

2.5.1. Card Code Translation (80-Column Card)

Eighty-column cards can be read or punched in compressed code or image mode. In the case of compressed code, the hardware compression is to or from an eight-bit code in memory. Because this eight-bit code is not one of the internal processing codes, if it is to be manipulated it requires program translation after a card is read into storage, and it requires program retranslation to generate eight-bit code for punching. This hardware compression is shown in Figure 2–3.



PUNCH	BITS 1 2 3
NONE	000
1	011
2	101
3	001
4	010
5	100
6	111
7	110

Figure 2–3. Compressed Code for 80-Column Card

If more than one row punch from 1 through 7 is present on a card, the codes will be combined (logical OR); thus a 2 and a 5 punch will appear as a 2 punch.

2.5.1.1. Card Code Image Mode (80-Column Card)

When an 80-column card is read or punched in image mode, the 12 punched holes of a column are represented by 1's in 2 bytes of storage; bit positions 0 and 1 are not used in image mode. When punching image mode, bit positions 0 and 1 will be ignored. Bits 0 and 1 will be cleared to zeros on an image read. When reading or punching in the image mode, the 80 columns of the card occupy 160 bytes of storage. These 160 bytes are consecutively occupied by 2 bytes for each consecutive column in a card starting at column 1. Card code image mode for an 80-column card is shown in Figure 2–4.

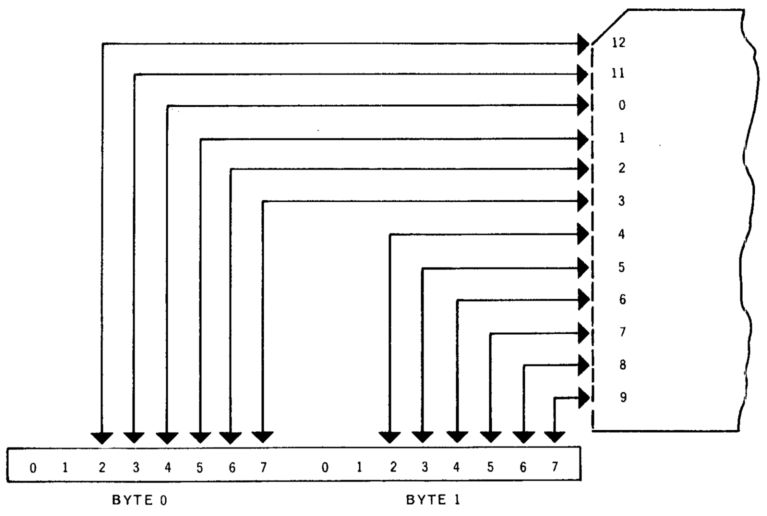
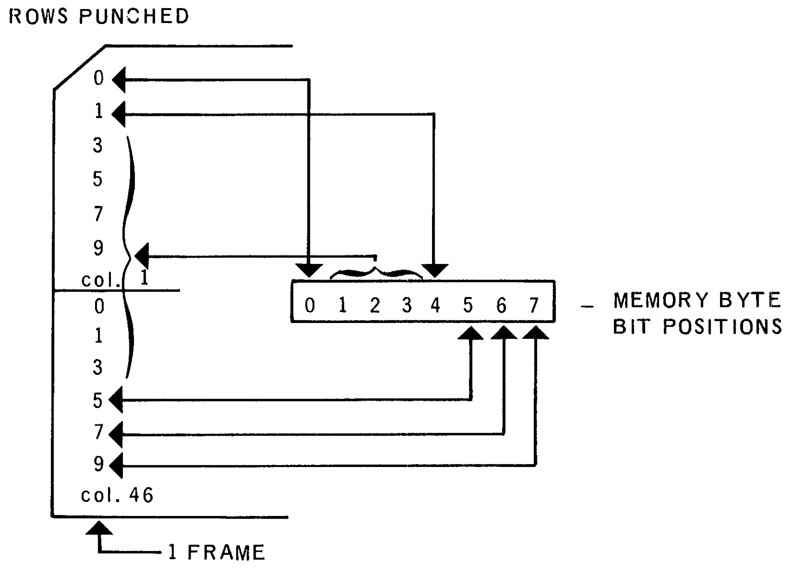


Figure 2–4. Image Mode for 80-Column Card

2.5.2. Card Code Translation (90-Column Card)

When a 90-column card is read or punched with hardware translation, the hardware translation is to or from an 8-bit code in storage. The hardware translation of row punches in the 90-column card to or from an 8-bit code are shown in Figure 2–5.



Punches in rows 3 (Upper) through 3 (Lower) generate the following bit combinations:

ROWS	123
No Punch	= 000
3 " (U)	= 110
5 " (U)	= 111
7 " (U)	= 100
9 " (U)	= 010
0 " (L)	= 001
1 " (L)	= 101
3 " (L)	= 011

If more than one row punch from 3 (Upper) through 3 (Lower) is present on a card, the combined codes will be OR'ed together; thus, a 7 (Upper) and a 1 (Lower) punch will appear to have been a 1 (Lower) punch.

Figure 2–5. Card Code Translation for 90-Column Card

2.5.2.1. Card Code Image Mode (90-Column Card)

When a 90-column card is read or punched in the image mode, the upper six possible punched holes and lower six possible punched holes of a frame are represented by 1's in two bytes of storage. As shown in Figure 2–6, bit positions 0 to 1 are unused in the image mode. When punching the image mode, 1's in bit positions 0 to 1 will be ignored. When reading or punching in the image mode the 90 columns of the card occupy 90 bytes of storage. These 90 bytes are consecutively occupied by 2 bytes for each consecutive frame in a card starting at frame 1 (column 1 and column 46). Bits 0 and 1 of the data bytes will be cleared to zeros on an image read.

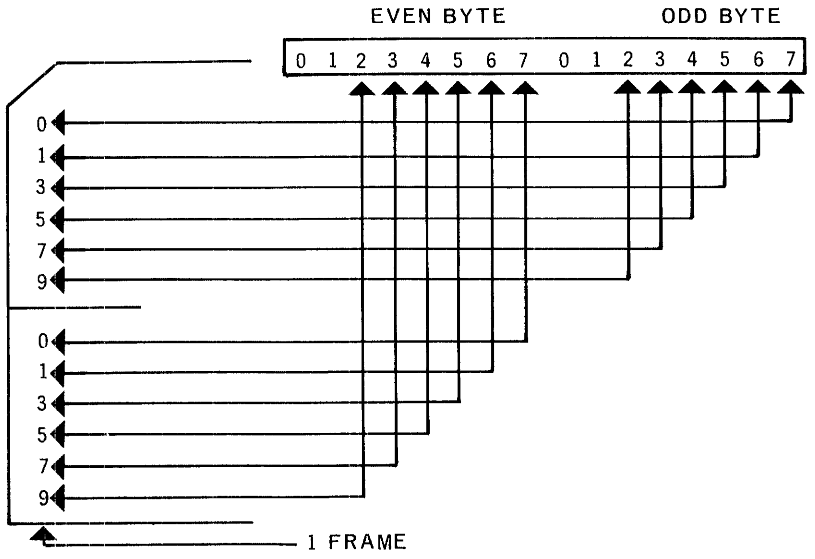


Figure 2–6. Image Mode for 90-Column Card

2.5.3. Internal Codes

There are 256 possible code combinations that can be represented by any eight-bit byte of storage. As described previously, card code translation is to or from a compressed code. This compressed code generated by the card reader must then be translated to the desired internal code for processing and retranslated from the internal code to the compressed code for punching. There are two sets of internal codes which are subsets of 256 possible code combinations which will be used to select the two bar printer graphics. Tables C–2 and C–3 of Appendix C contain the 63 character print set and the 48 character print set. Table C–4 contains the 16 character print set which is a subset of the 48 character print set.

3. INSTRUCTIONS

3.1. GENERAL

The UNIVAC 9200/9200 II/9300/9300 II Systems processor instruction repertoire consists of 35 machine instructions. These machine instructions are listed in Tables 3–1 through 3–4. Table 3–1 lists the instructions alphabetically by mnemonic symbol. Table 3–2 lists the instructions according to their six major functional types. Table 3–3 lists the instructions by type of format. Table 3–4 lists the instructions according to hexadecimal operation code.

The UNIVAC 9200 and 9200 II Systems processor may have, optionally, the Divide Packed Decimal, Multiply Packed Decimal, and Edit feature. This option is offered as either hardware instructions or as fixed, closed routines which duplicate the functions of the instructions. The use of the hardware option instructions does not differ from that of the standard instructions.

MNEMONIC	FUNCTION	HEXADECIMAL OPERATION CODE	FORMAT
AH	ADD HALFWORD	AA	RX
AI	ADD IMMEDIATE	A6	SI
AP	ADD PACKED DECIMAL	FA	SS2
BAL	BRANCH AND LINK	45	RX
BC	BRANCH ON CONDITION	47	RX
CH	COMPARE HALFWORD	49	RX
CLC	COMPARE LOGICAL CHARACTER	D5	SS1
CLI	COMPARE LOGICAL IMMEDIATE	95	SI
CP	COMPARE PACKED DECIMAL	F9	SS2
DP	DIVIDE PACKED DECIMAL	FD	SS2
ED	EDIT	DE	SS1
HPR	HALT AND PROCEED	A9	SI
LH	LOAD HALFWORD	48	RX
LPSC	LOAD PROGRAM STATE CONTROL	A8	SI
MP	MULTIPLY PACKED DECIMAL	FC	SS2
MVC	MOVE CHARACTERS	D2	SS1
MVI	MOVE IMMEDIATE	92	SI
MVN	MOVE NUMERICS	D1	SS1
MVO	MOVE WITH OFFSET	F1	SS2
NC	*AND* CHARACTERS	D4	SS1
NI	*AND* IMMEDIATE	94	SI
OC	*OR* CHARACTERS	D6	SS1
OI	*OR* IMMEDIATE	96	SI
PACK	PACK	F2	SS2
SH	SUBTRACT HALFWORD	AB	RX
SP	SUBTRACT PACKED DECIMAL	FB	SS2
SPSC	STORE PROGRAM STATE CONTROL	A0	SI
SRC	SUPERVISOR REQUEST	A1	SI
STH	STORE HALFWORD	40	RX
TIO	TEST I/O	A5	SI
TM	TEST UNDER MASK	91	SI
TR	TRANSLATE	DC	SS1
UNPK	UNPACK	F3	SS2
XIOF	EXECUTE INPUT/OUTPUT FUNCTION	A4	SI
ZAP	ZERO ADD PACKED DECIMAL	F8	SS2

Table 3–1. Instruction Mnemonics in Alphabetical Sequence

FUNCTIONAL TYPE	MNEMONIC	FUNCTION	HEXADECIMAL OPERATION CODE	FORMAT
Binary	STH	Store Halfword	40	RX
	LH	Load Halfword	48	RX
	CH	Compare Halfword	49	RX
	AI	Add Immediate	A6	SI
	AH	Add Halfword	AA	RX
	SH	Subtract Halfword	AB	RX
Logical	TM	Test Under Mask	91	SI
	MVI	Move Immediate	92	SI
	NI	AND Immediate	94	SI
	CLI	Compare Logical Immediate	95	SI
	OI	OR Immediate	96	SI
	HPR	Halt and Proceed	A9	SI
	MVN	Move Numeric	D1	SS1
	MVC	Move Character	D2	SS1
	NC	AND Character	D4	SS1
	CLC	Compare Logical Character	D5	SS1
	OC	OR Character	D6	SS1
	TR	Translate	DC	SS1
	ED	Edit	DE	SS1
Decimal	MVO	Move With Offset	F1	SS2
	PACK	Pack	F2	SS2
	UNPK	Unpack	F3	SS2
	ZAP	Zero Add Packed Decimal	F8	SS2
	CP	Compare Packed Decimal	F9	SS2
	AP	Add Packed Decimal	FA	SS2
	SP	Subtract Packed Decimal	FB	SS2
	MP	Multiply Packed Decimal	FC	SS2
	DP	Divide Packed Decimal	FD	SS2
Branch	BAL	Branch and Link	45	RX
	BC	Branch-On-Condition	47	RX
State Control	LPSC	Load Program State Control	A8	SI
	SPSC	Store Program State Control	A0	SI
	SRC	Supervisor Request Call	A1	SI
Input/Ouptut	XIOF	Execute I/O	A4	SI
	TIO	Test I/O	A5	SI

Table 3–2. Instructions Grouped by Functional Type

FORMAT	HEXADECIMAL OPERATION CODE	MNEMONIC	FUNCTION
RX	40	STH	STORE HALFWORD
RX	45	BAL	BRANCH AND LINK
RX	47	BC	BRANCH ON CONDITION
RX	48	LH	LOAD HALFWORD
RX	49	CH	COMPARE HALFWORD
RX	AA	AH	ADD HALFWORD
RX	AB	SH	SUBTRACT HALFWORD
SI	91	TM	TEST UNDER MASK
SI	92	MVI	MOVE IMMEDIATE
SI	94	NI	*AND* IMMEDIATE
SI	95	CLI	COMPARE LOGICAL IMMEDIATE
SI	96	OI	*OR* IMMEDIATE
SI	A0	SPSC	STORE PROGRAM STATE CONTROL
SI	A1	SRC	SUPERVISOR REQUEST
SI	A4	XIOF	EXECUTE INPUT/OUTPUT FUNCTION
SI	A5	TIO	TEST I/O
SI	A6	AI	ADD IMMEDIATE
SI	A8	LPSC	LOAD PROGRAM STATE CONTROL
SI	A9	HPR	HALT AND PROCEED
SS1	D1	MVN	MOVE NUMERICS
SS1	D2	MVC	MOVE CHARACTERS
SS1	D4	NC	*AND* CHARACTERS
SS1	D5	CLC	COMPARE LOGICAL CHARACTER
SS1	D6	OC	*OR* CHARACTERS
SS1	DC	TR	TRANSLATE
SS1	DE	ED	EDIT
SS2	F1	MVO	MOVE WITH OFFSET
SS2	F2	PACK	PACK
SS2	F3	UNPK	UNPACK
SS2	F8	ZAP	ZERO ADD PACKED DECIMAL
SS2	F9	CP	COMPARE PACKED DECIMAL
SS2	FA	AP	ADD PACKED DECIMAL
SS2	FB	SP	SUBTRACT PACKED DECIMAL
SS2	FC	MP	MULTIPLY PACKED DECIMAL
SS2	FD	DP	DIVIDE PACKED DECIMAL

Table 3–3. Instructions Grouped by Format

HEXADECIMAL OPERATION CODE	MNEMONIC	FUNCTION	FORMAT
40	STH	STORE HALFWORD	RX
45	BAL	BRANCH AND LINK	RX
47	BC	BRANCH ON CONDITION	RX
48	LH	LOAD HALFWORD	RX
49	CH	COMPARE HALFWORD	RX
91	TM	TEST UNDER MASK	SI
92	MVI	MOVE IMMEDIATE	SI
94	NI	*AND* IMMEDIATE	SI
95	CLI	COMPARE LOGICAL IMMEDIATE	SI
96	OI	*OR* IMMEDIATE	SI
A0	SPSC	STORE PROGRAM STATE CONTROL	SI
A1	SRC	SUPERVISOR REQUEST	SI
A4	XIOF	EXECUTE INPUT/OUTPUT FUNCTION	SI
A5	TIO	TEST I/O	SI
A6	AI	ADD IMMEDIATE	SI
A8	LPSC	LOAD PROGRAM STATE CONTROL	SI
A9	HPR	HALT AND PROCEED	SI
AA	AH	ADD HALFWORD	RX
AB	SH	SUBTRACT HALFWORD	RX
D1	MVN	MOVE NUMERICS	SS1
D2	MVC	MOVE CHARACTERS	SS1
D4	NC	*AND* CHARACTERS	SS1
D5	CLC	COMPARE LOGICAL CHARACTER	SS1
D6	OC	*OR* CHARACTERS	SS1
DC	TR	TRANSLATE	SS1
DE	ED	EDIT	SS1
F1	MVO	MOVE WITH OFFSET	SS2
F2	PACK	PACK	SS2
F3	UNPK	UNPACK	SS2
F8	ZAP	ZERO ADD PACKED DECIMAL	SS2
F9	CP	COMPARE PACKED DECIMAL	SS2
FA	AP	ADD PACKED DECIMAL	SS2
FB	SP	SUBTRACT PACKED DECIMAL	SS2
FC	MP	MULTIPLY PACKED DECIMAL	SS2
FD	DP	DIVIDE PACKED DECIMAL	SS2

Table 3–4. Instructions in Sequence of Hexadecimal Codes

3.2. INSTRUCTION FORMAT

There are three basic instruction formats used in the UNIVAC 9200, 9200 II, 9300 and 9300 II Systems processors: the RX format, the SI format, and the SS format. The content of each type of format is shown in Figure 3–1. The functions of the fields within each format are as follows:

R1	Specifies address of program register.
B1D1	Specifies address of operand 1. If the most significant bit of B1 is zero, B1D1 specifies a direct address. If the most significant bit of B1 is one, B1 specifies one of eight binary arithmetic registers. D1 specifies that an indexing value is to be added to the contents of the register. The resulting value specifies indexed address (displacement from the base address of operand 1–1).
B2D2	Specifies address of operand 2 (direct or indexed as for operand 1). D2 is the displacement from the base address of operand 2.
I2	Immediate data: this is operand 2 of instruction.
L1	Specifies byte length of operand 1; L1 value is 1 less than actual number of bytes processed.
L2	Specifies byte length of operand 2; L2 value is 1 less than actual number of bytes processed.
OP1	Operand 1
OP2	Operand 2
Code	The numeric operation code of the instruction.
Symbol	The expression or symbolic label used as operand 1.
Tag	The expression or symbolic label used as operand 2.

3.2.1. Register and Indexed Storage Operation (RX)

Instructions in the RX format are four bytes long. They are used to process fixed length data files having a length of two bytes. One operand always specifies a general register. This format is used for functions such as branching, comparing, adding, storing, and loading.

3.2.2. Storage and Immediate Operand Operation (SI)

Instructions in the SI format are four bytes long. They are used to process data one byte in length using control or additional data contained in the immediate operand. This format is used for logical, arithmetic, manipulative, and testing functions.

3.2.3. Storage-to-Storage (SS1)

Instructions in SS1 format are six bytes long. They are used to process data of variable length, up to 256 bytes, if the operands specify fields of equal length. This format is used for functions such as comparing, transferring, and translating.

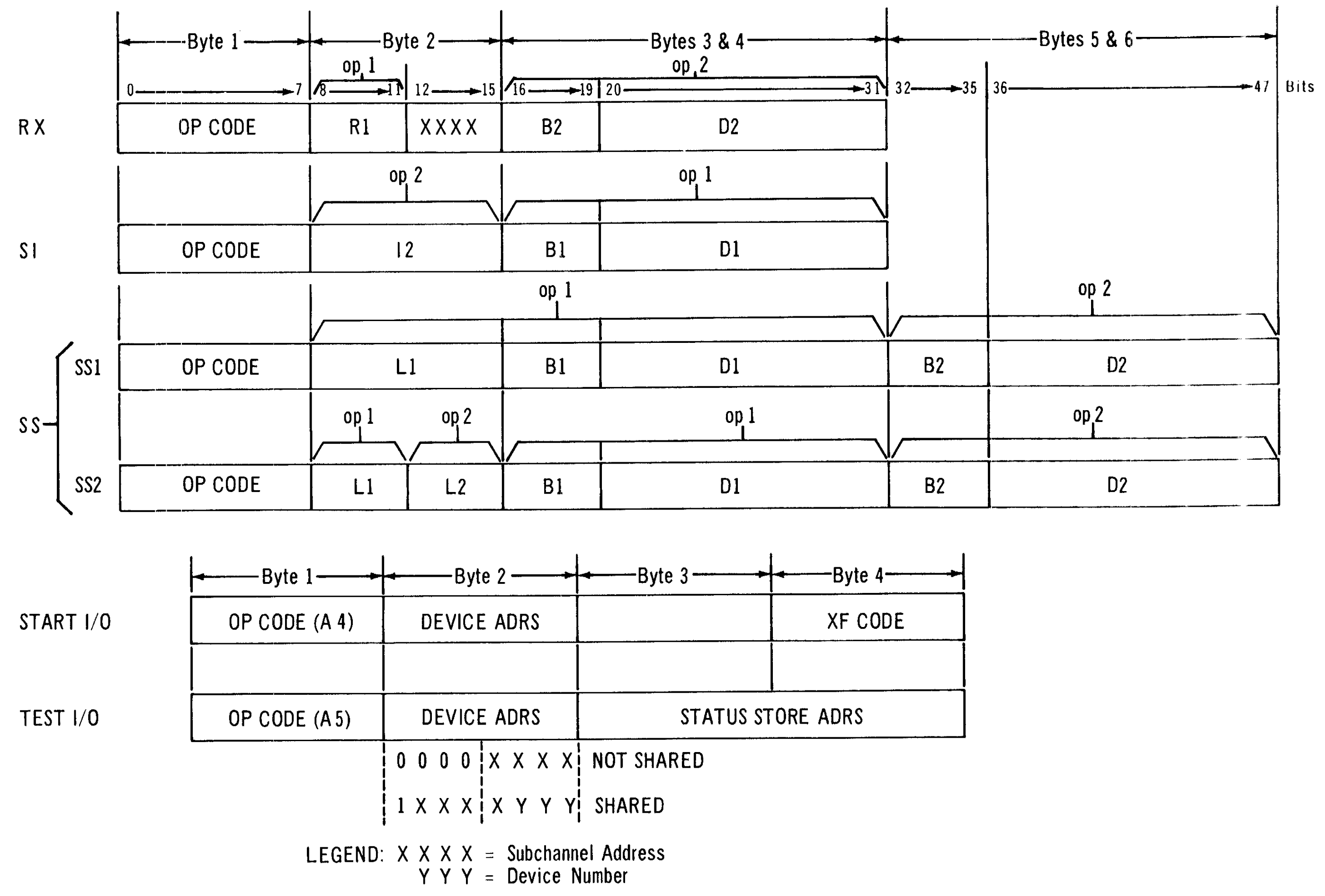


Figure 3–1. Instruction Formats

3.2.4. Storage-to-Storage (SS2)

Instructions in SS2 format are six bytes long. They are used to process variable length operands no larger than 16 bytes, when the operands are not of equal length. This format is used for functions such as shift operations, pack, and unpack.

3.3. INSTRUCTION REPERTOIRE

Each execute (instruction) cycle performed by the processor falls into one of the following types:

Binary instructions

Logical instructions

Decimal instructions

Branch instructions

State control instructions

Input/output instructions

Further detailed explanation pertaining to programming these instructions are given in *UNIVAC 9200/9200 II/9300/9300 II Systems Card Assembler Programmers Reference, UP-4092,* and *UNIVAC 9200/9200 II/9300/9300 II Systems Tape/Disc Assembler Programmers Reference, UP-7508,* (current versions). The descriptions follow the order given in Table 3–2; refer to Figure 3–1 for the functions of the fields within each instruction format.

3.3.1. Binary Instructions and Overflow

Binary instructions have the following common characteristics:

(1) All data involved is assumed to be in binary form, and is processed by binary arithmetic.

(2) All data processing is within halfword boundaries.

(3) With the exception of the Add Immediate, all binary instructions control the use of the arithmetic registers and use the RX instruction (fixed length) format.

A binary number is positive if the most significant bit of the number is a zero, and negative if the most significant bit is a one. Negative numbers are expressed as the two's complement of the magnitude of the number.

The binary instructions that control the use of arithmetic registers are:

Store halfword
Load halfword
Add halfword
Subtract halfword
Compare halfword

Some of the binary values that can be expressed in the halfword registers on their corresponding halfwords in memory are as follows:

BINARY NUMBER	DECIMAL EQUIVALENT	BIT REPRESENTATION
$2^{15}-1$	32767	0111 1111 1111 1111
2^8	256	0000 0001 0000 0000
2^0	1	0000 0000 0000 0001
0	0	0000 0000 0000 0000
-2^0	–1	1111 1111 1111 1111
-2^1	–2	1111 1111 1111 1110
-2^8	–256	1111 1111 0000 0000
$-2^{15}+1$	–32767	1000 0000 0000 0001
-2^{15}	–32768	1000 0000 0000 0000

The condition codes resulting from addition, subtraction, and comparison giving the results of comparing the address in the program or index register (OP1) with the D portion of the RX format (OP2) is as follows:

	CONDITION CODE SETTING			
	0	1	2	3
Compare halfword	Operands are equal	OP1 is less than OP2	OP1 is greater than OP2	---
Subtract halfword	zero result	negative result	positive result	overflow
Add halfword	zero result	negative result	positive result	overflow

Binary overflow occurs when the sign bit is changed erroneously by a carry from the adjacent bit; this happens when an attempt is made to develop a larger negative or positive number than can be expressed in 15 bits.

3.3.1.1. Store Halfword Instruction (STH)

The store halfword instruction stores the contents of the arithmetic register specified by the RX field of the instruction in the halfword (operand 2) at the storage address specified by the B2D2 field of instruction.

3.3.1.2. Load Halfword Instruction (LH)

The load halfword instruction loads the halfword specified by the B2D2 (operand 2) field of the instruction into the arithmetic register specified by the RX field of instruction.

3.3.1.3. Compare Halfword Instruction (CH)

The compare halfword instruction algebraically compares the halfword (operand 2) specified by the B2D2 field of the instruction with the contents of the arithmetic register specified by the RX field of the instruction. The comparison is accomplished by adding the twos complement of the contents of storage to the contents of the arithmetic register. The resulting number is not stored, but an appropriate condition code (contents of register is equal to, less than, or greater than the contents of storage byte) is stored in the FAS field of privileged storage.

3.3.1.4. Add Immediate Instruction (AI)

The add immediate instruction uses the SI format; it adds the contents of the I2 (immediate data) field of the instruction to the halfword (operand 1) specified by the B1D1 field. The I2 field of the instruction is expanded to a halfword boundary by extending the sign (one byte of all zeros or all ones) of the I2 field. The resulting number is stored in the halfword of storage specified by B1D1, and an appropriate condition code is stored in the FAS field of privileged storage.

3.3.1.5. Add Halfword Instruction (AH)

The add halfword instruction adds the halfword (operand 2) specified by the B2D2 field of the instruction to the contents of the arithmetic register specified by the RX field of the instruction. The sum is stored in the arithmetic register specified by the RX field and an appropriate condition code (zero, positive, negative, or overflow) is stored in the FAS field of privileged storage.

3.3.1.6. Subtract Halfword Instruction (SH)

The subtract halfword instruction adds the two's complement of halfword (operand-2) specified by the B2D2 field of instruction to the contents of the arithmetic register specified by the RX field of the instruction. The resulting difference is stored in the arithmetic register specified by the RX field, and an appropriate condition code (zero, positive, negative, or overflow) is stored in the FAS field of the privileged storage.

3.3.2. Logical Instructions

Logical instructions combine, modify, or otherwise manipulate data by processes that, generally speaking, are not arithmetic.

3.3.2.1. Test Under Mask Instruction (TM)

The test under mask instruction uses the SI format and matches the 1 bits in the I2 field of the instruction with the corresponding 1 bits in operand 1 (one byte) specified by the B1D1 field of the instruction. If all the bits in the mask are matched against 0 bits in the byte tested, the condition code will be 0. If some of the 1 bits in the mask have corresponding 1 bits in the byte tested, the condition code is 1. If all the 1 bits in the mask have corresponding 1 bits in the byte tested, the condition code is 3. If the mask is 0, the resultant condition code setting will be 0. The result is not stored, but the appropriate condition code is stored in the FAS field of the privileged storage.

3.3.2.2. Move Immediate Instruction (MVI)

The move immediate instruction uses the SI format and stores the I2 (immediate data) field of the instruction in operand 1 (one byte) specified by the B1D1 field of the instruction.

3.3.2.3. AND Immediate Instruction (NI)

The AND immediate instruction uses the SI format and forms a logical AND product with 1 bits (bits present) of the I2 field of the instruction and the corresponding 1 bits in operand 1 (one byte) specified by the B1D1 field of the instruction. If I2 and the storage byte both contain 1 bits in corresponding positions, the result is a 1 bit; otherwise, the result is a 0 bit. The resulting logical product is stored in the byte of memory specified by B1D1, and an appropriate condition code (result zero or result not zero) is stored in the FAS field of privileged storage.

3.3.2.4. Compare Logical Immediate Instruction (CLI)

The compare logical immediate instruction uses the SI format and compares operand 1 (one byte) specified by the B1D1 field of the instructions with the I2 field of the instruction. The comparison is accomplished by adding, without regard to sign, the twos complement of operand 1 to the contents of the I2 field. The result is not stored, but an appropriate condition code (contents of storage byte equal to, less than, or greater than contents of register) is stored in the FAS field of privileged storage.

3.3.2.5. OR Immediate Instruction (OI)

The OR immediate instruction uses the SI format and forms a logical OR sum with the 1 bits (bits present) of the I2 field of the instruction and the corresponding 1 bits of operand 1 (one byte) specified by the B1D1 field of the instruction. All 1 bits in I2 are superimposed onto the bits of operand 1; 0 bits in I2 leave the corresponding bits in the storage byte unchanged. The result is stored in the storage byte specified by B1D1, and an appropriate condition code (result zero or result not zero) is stored in the FAS field of privileged storage.

3.3.2.6. Halt and Proceed Instruction (HPR)

The halt and proceed instruction uses the SI format and causes the operation of the processor to stop, with bits 17 through 31 of the B1D1 field of the instruction displayed by the NEXT INSTRUCTION/HALT DISPLAY indicators on the control panel. Since normal indexing is possible, B1D1 when displayed may be an indexed value. The operation is resumed when the START switch on the control panel is pressed.

3.3.2.7. Move Numeric Instruction (MVN)

The move numeric instruction uses the SS1 format and transfers the numeric portion of the operand 2 bytes specified by the B2D2 field of the instruction to the numeric portion of the operand 1 bytes specified by the B1D1 field. The zone portions of operand 1 bytes are unaltered. The number of bytes transferred is one more than the number specified by the L field of the instruction.

3.3.2.8. Move Character Instruction (MVC)

The move character instruction uses the SS1 format and transfers operand 2 bytes specified by the B2D2 field of the instruction to operand 1 locations specified by the B1D1 field without altering data. The number of bytes transferred is one more than the number specified by the L field of the instruction.

3.3.2.9. AND Character Instruction (NC)

The AND character instruction uses the SS1 format and forms a logical AND product with operand 2 bytes specified by the B2D2 field of the instruction and operand 1 bytes specified by the B1D1 field of the instruction. The number of bytes processed is one more than the number specified by the L field. If the corresponding 1 bit of operand 1 and operand 2 bytes are both present, the result is a 1 bit; otherwise, the result is a 0 bit. The result is stored in operand 1 and an appropriate condition code (result zero or result not zero) is stored in the FAS field of privileged storage.

3.3.2.10. Compare Logical Character Instruction (CLC)

The compare logical character instruction uses the SS1 format and compares operand 1 bytes specified by the B1D1 field of the instruction to operand 2 bytes specified by the B2D2 field. The comparison is accomplished by adding the twos complement of operand 2 to operand 1. The number of bytes compared is one more than the number specified in the L field of the instruction or the comparison is terminated on the first inequality. The result is not stored, but an appropriate condition code (operand 1 is equal to, less than, or greater than operand 2) is stored in the FAS field of privileged storage.

3.3.2.11. OR Character Instruction (OC)

The OR character instruction uses the SS1 format and forms a logical OR sum with operand 2 bytes specified by the B2D2 field of the instruction and operand 1 bytes specified by the B1D1 field of the instruction. All 1 bits in operand 2 are superimposed on operand 1 bits; 0 bits in operand 2 leave the corresponding bits in operand 1 unchanged. The number of bytes processed is one more than the number specified by the L portion of the instruction. The result is stored in operand 1, and an appropriate condition code (result zero or result not zero) is stored in the FAS field of privileged storage.

3.3.2.12. Translate Instruction (TR)

The translate instruction uses the SS1 format and translates any eight-bit code to any other eight-bit code. Initially the data to be translated is located in operand 1, specified by the B1D1 field of the instruction, and the characters to which they will be translated are located in operand 2 specified by the eight-bit code to be translated plus the B2D2 field of the instruction. The operand 1 character is added to the contents of the B2D2 field and the result is used as an address to locate the corresponding (translated) character in operand 2. The translated character is then transferred from operand 2 to operand 1, replacing the character to be translated with the appropriate translated character.

3.3.2.13. Edit Instruction (ED)

The edit instruction uses the SS1 format and unpacks, expands, and modifies data contained in operand 2 bytes specified by the B2D2 field of the instruction and stores it in operand 1 specified by the B1D1 field. The value inserted in the zone position of the edited data is a hexadecimal F in EBCDIC mode or a hexadecimal 5 in ASCII mode.

The editing process basically consists of selected zero suppression and chatacter insertion. This process is controlled by a pattern of edit characters and special control bytes originally contained in operand 1, and by the significance indicator (AS FF) associated with the edit instruction. The edit characters and special control bytes contained in the edit pattern, and their effects on the edit process are as follows:

■ Fill Character	– Originally contained in the most significant byte of operand 1 and retained there after editing. During editing, it replaces nonsignificant data from operand 2 in the appropriate location of operand 1.
■ Other Edit Symbols	– Characters such as commas and decimal points are retained if the significance flip-flop (AS FF) is set, or are replaced by the fill character if the AS FF is not set.
■ Digit Select Byte (DSB)	– Before the significance flip-flop (AS FF) is set, the DSB causes zero digits in operand 2 to be replaced by the fill character. The DSB causes the AS FF to be set when the first nonzero is encountered in operand 2. Once the AS FF is set, the DSB causes the digit in operand 2 (including zero) to be transferred to the operand 1 location occupied by the DSB.

- Significant Start Byte (SSB) – Sets the AS FF, causing leading zeroes in operand 2 to be transferred to operand 1. It is placed in the edit pattern in the location preceding the one that is to contain the first leading zero. The SSB is replaced in operand 1 by the fill character.

- Field Separator Byte (FSB) – Clears the AS FF, so that all zeroes in operand 2 and all edit symbols in operand 1 following the FSB are replaced by the fill characters in operand 1 until the AS FF is again set by any of the conditions already mentioned. If the AS FF is set, any character in operand 1 other than a DSB, SSB, or FSB is retained, and operand 2 is not accessed. If the AS FF is not set, characters other than these are replaced by fill characters.

The number of bytes in operand 1 is one more than the number specified in the L portion of the instruction. The number of digits (including zeroes) in operand 2 must equal the sum of the DSB's, SSB's, and FSB's in operand 1.

A condition code determined by the sign of operand 2 (positive, negative, or unsigned), is stored in the FAS field of privileged storage. In the case of multiple field editing, the condition code reflects only the portion of operand 2 following the last FSB in the operand 1 pattern.

3.3.3. Decimal Instructions

The decimal instructions involve processing of data in decimal digit form. The data processing includes decimal arithmetic operations, which alter the data to or from the form required by the decimal arithmetic operations.

The decimal instructions all use the SS2 format; they process the data from right to left. The L field of these instructions is divided into two parts, L1 and L2, which specify one less than the number of bytes in operands 1 and 2 respectively. The B1D1 and B2D2 fields specify the locations of operands 1 and 2 respectively.

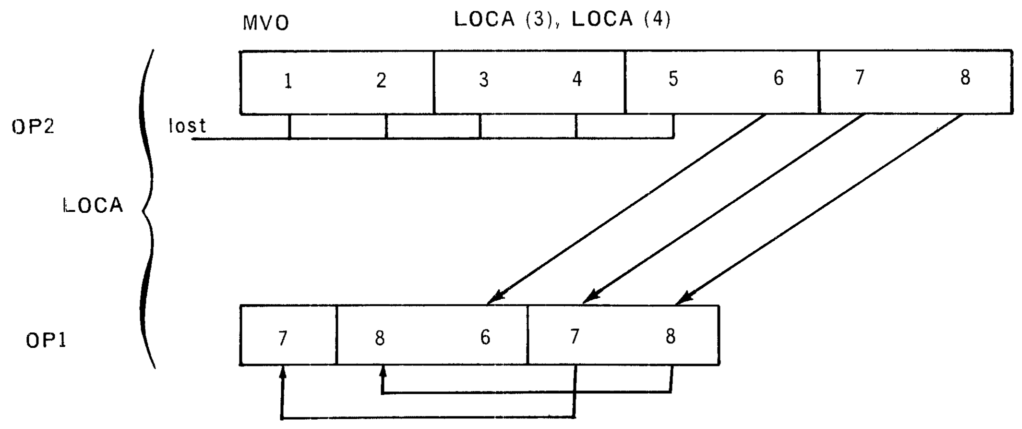
The decimal instructions involving arithmetic operations require the data to be in packed decimal format.

3.3.3.1. Move With Offset Instruction (MVO)

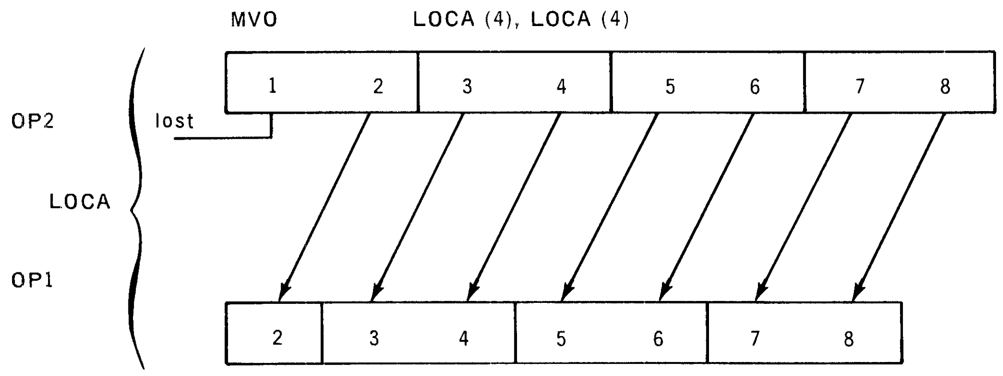
The move with offset instruction transfers data from operand 2 to operand 1, shifting the data four bits (one decimal digit) left. The least significant four bits of the least significant byte of operand 1 (decimal sign) is retained without alteration.

If the number of bytes in operand 2, specified by L2, is less than the number of bytes in operand 1, zero digits are assumed for operand 2, and the remaining digits of operand 1 are zerofilled. If L2 is greater than L1, the operation is terminated when L1 + 1 digits of operand 1 are processed, and the remaining digits of operand 2 are ignored.

Since the MVO operation proceeds from right to left and moves 4 bits at a time, the use of like sending and receiving location with unlike length can produce overlayed results. For example:



A shift of only 4 bits left, however, is correctly accomplished by the MVO operation as shown:



3.3.3.2. Pack Instruction (PACK)

The pack instruction transfer unpacked (zone-digit) decimal data from operand 2 to operand 1 in packed (digit-digit) format.

In the unpacked data of operand 2, each byte contains one decimal digit (including zone), and the sign digit is contained in the upper half of the least significant digit. When this data is packed into operand 1, the sign is inserted into the lower half of the least significant byte. The zone bits from each decimal digit in operand 2 are deleted, and the decimal digits are transferred into operand 1, into contiguous positions. Unpacked to packed format conversion is:

Z	1	Z	2	Z	3	Z	4	Sign	5	UNPACKED FORMAT
					1	2	3	4	5	Sign	PACKED FORMAT

If all bytes of operand 2 are processed before those of operand 1, zero-digits are inserted into the remaining positions of operand 1. If all bytes of operand 1 are processed before those of operand 2, the operation is terminated, and the remaining bytes of operand 2 are ignored.

3.3.3.3. Unpack Instruction (UNPK)

The unpack instruction transfers packed data from operand 2 into operand 1 in unpacked format (reverse of pack instruction). The zone code inserted into operand 1 bytes during unpacking is 0101 if operations are in ASCII mode. If operations are in EBCDIC mode, the zone code inserted into operand 1 bytes during unpacking is 1111.

If operand 1 is too short to contain all the unpacked digits from operand 2, the operation is terminated and the remaining digits of operand 2 are ignored. If all digits of operand 2 are processed before all operand 1 bytes, the remaining bytes of operand 1 are zerofilled.

3.3.3.4. Zero Add (Packed) Decimal Instruction (ZAP)

The zero and add instruction transfers data from operand 2 to operand 1. This is accomplished by accessing only the bytes (digits) of operand 2, and storing them in operand 1 locations.

Operand 1 must contain at least as many bytes as operand 2. If operand 1 contains more bytes than operand 2, the most significant bytes of operand 2 are zerofilled.

The sign of operand 2 is transferred directly to operand 1 without change, unless the result is a negative zero. In this case, the resulting sign of operand 1 is forced to plus. A condition code reflecting the sign of operand 2 (with the exception mentioned) is stored in the FAS field of privileged storage.

3.3.3.5. Compare (Packed) Decimal Instruction (CP)

The compare decimal instruction compares operand 2 with operand 1. The comparison is accomplished by adding the twos complement of operand 2 to operand 1 and setting an appropriate condition code. The result of the addition is not stored, but the condition code is stored in the FAS field of privileged storage.

The comparison is processed from left to right and will terminate at the first inequality. Operand 1 must include at least as many bytes as operand 2. If operand 1 is larger than operand 2, zero digits are assumed to extend to operand 2.

3.3.3.6. Add (Packed) Decimal Instruction (AP)

The add decimal instruction algebraically adds the decimal digits of operand 2 to those of operand 1, and stores the resulting sum in operand 1.

Operand 1 must include at least as many bytes as operand 2. If operand 1 is larger than operand 2, zero digits are used to extend operand 2.

If the result of the addition is a negative zero, the sign of the result in operand 1 is forced to plus. The sign inserted will be ASCII or EBCDIC minus or plus, according to the processing mode. An appropriate condition code (result zero, negative, positive, or overflow) is stored in the FAS field of privileged storage.

3.3.3.7. Subtract (Packed) Decimal Instruction (SP)

The subtract decimal instruction algebraically subtracts operand 1 from operand 2 and stores the resulting difference in operand 1. The subtraction is accomplished by adding the decimal tens complement of operand 2 to operand 1.

Operand 1 must include at least as many bytes as operand 2. If operand 1 is larger than operand 2, zero digits are assumed to extend operand 2.

If the result of the subtraction is a negative zero, the sign of the result of operand 1 is forced to a plus. The sign inserted will be a minus or plus ASCII or EBCDIC, according to the processing mode. An appropriate condition code (result zero, negative, positive, or overflow) is stored in the FAS field of privileged storage.

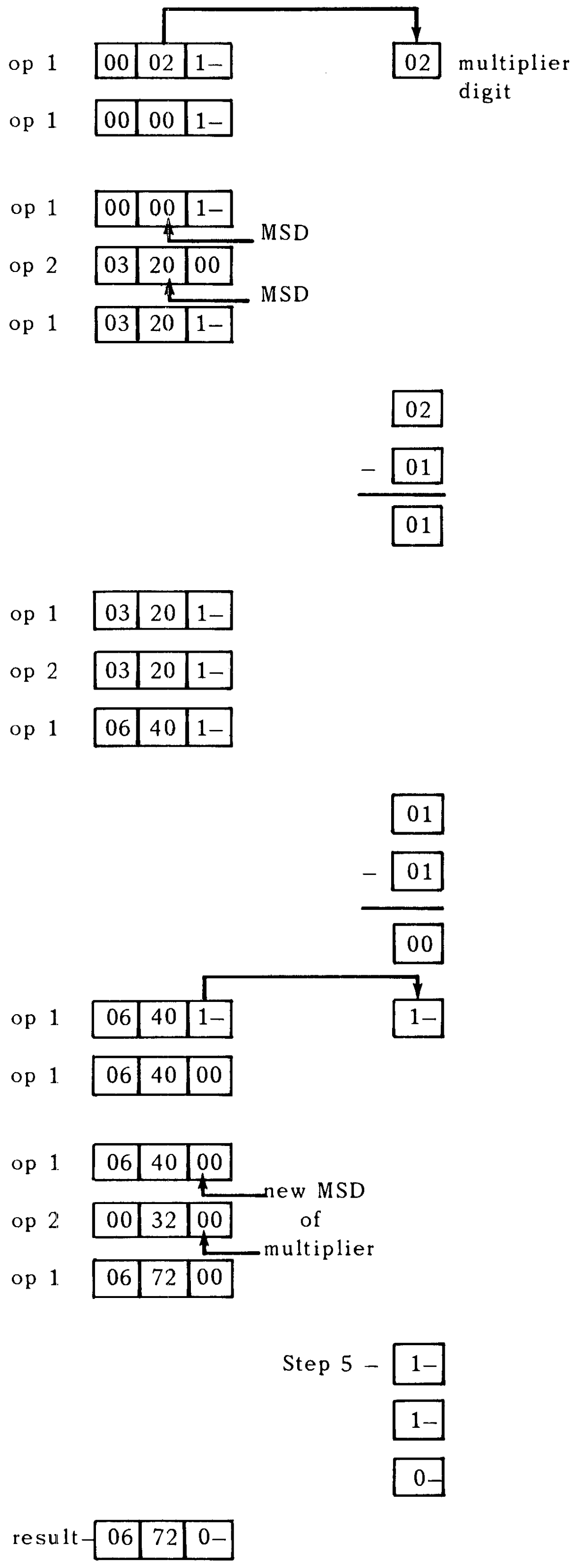
3.3.3.8. Multiply (Packed) Decimal Instruction (MP)

The multiply decimal instruction multiplies operand 2 (multiplicand) by operand 1 (multiplier) and stores the resulting product in operand 1. The number of bytes in the multiplier (originally in operand 1) is equal to the number of bytes in operand 1 minus the number of bytes in operand 2. If the multiplier includes more bytes than this difference, the most significant bytes of the multiplier are ignored.

The multiplication process consists of adding the multiplicand to operand 1 (initially zeroed) a number of times determined by the multiplier digits. Each addition is processed from right to left, starting at the digit position in operand 1 corresponding to the position of the most significant multiplier digit. After each addition, the most significant multiplier digit is reduced by one. This process is repeated until the most significant multiplier digit is reduced to zero and then the next most significant multiplier digit is treated as the most significant digit. When all multiplier digits are reduced to zero, operand 1 contains the completed product, except for the sign. If the multiplier and multiplicand have like signs, a plus is inserted into the least significant digit of the product; if the signs are unlike, a minus is inserted. An example of the basic multiply operation is as follows:

Example: Multiply 320 (+) by 21 (–)

Operand 1 = $\boxed{00}\boxed{02}\boxed{1-}$ (multiplier); Operand 2 = $\boxed{32}\boxed{0+}$ (multiplicand)



Operation

(1) Store multiplier digit. Clear same in operand 1.

(2) Add operand 2 to operand 1 starting from right to left at most significant digit (MSD) of multiplier.

(3) Decrement MSD of multiplier, op 1 saved.

(4) Repeat steps 2 and 3.

(5) Multiplier digit now equal to zero.

Store new digit and clear same in op 1.

(6) Add op 2 to op 1, from right to left, starting at digit position corresponding to new MSD of multiplier.

(7) Decrement new MSD of multiplier digit (step 5).

(8) Insert sign.

3.3.3.9. Divide (Packed) Decimal Instruction (DP)

The divide decimal instruction divides operand 1 (dividend) by operand 2 (divisor). The quotient and the remainder are stored at the address originally containing operand 1; the quotient is right justified in the (L1 – L2) most significant bytes of the storage area, and the remainder right justified in the (L2 + 1) least significant bytes in the same area. The sign of the quotient is positive if the signs of the dividend and divisor are alike, and negative if the signs are not alike.

The division process consists of subtracting the divisor from and adding the divisor to the dividend. A count of each subtraction and addition is stored in a working storage location called the quotient counter, whose content is inserted into the appropriate operand 1 (quotient) location after each subtraction cycle and associated addition cycle is completed.

Before each subtraction cycle the quotient counter is initially set to 09. The divisor is then subtracted from the most significant portion of the dividend (trial dividend). If the magnitude of the divisor is less than that of the dividend (determined after subtraction), the upper digit of the quotient counter is incremented by one. The subtraction is repeated until the magnitude of the dividend becomes less than that of the divisor before subtraction, and less than zero after subtraction. When this occurs, the quotient counter is not incremented, and the addition cycle begins.

The divisor is then right shifted one digit and is added to a new dividend which includes the next most significant digit, and the lower digit of the quotient counter is incremented by one. The addition continues until the trial dividend again becomes greater than zero.

When this occurs, the quotient counter is not incremented, and the entire contents of the quotient counter is entered into the most significant byte of the previous trial dividend.

The subtraction and addition cycles continue until the least significant byte, which contains the sign of the operand 1, is accessed. When this occurs, the subtraction cycle is completed as before. In the addition cycle following, however, the divisor is added to the dividend once to restore the dividend to the value it had prior to the last subtraction. Also during this cycle, the appropriate signs of the quotient and remainder are inserted into the proper locations. An example of the basic division operation is as follows:

Example: Divide 1149 (+) by 21 (–)

Operand –2 = $\boxed{02}\boxed{1-}$; Operand –1 = $\boxed{00}\boxed{01}\boxed{14}\boxed{9+}$

Quotient Counter	Operand –1	Operation
	00 01 14 9+ – 2 1 99 99 04 9+	(1) Subtract divisor from trial dividend.
09		(2) Store initial count of 09 in quotient counter; do not increment upper digit, since result is less than zero.
	99 99 04 9+ + 21 99 99 25 9+	(3) Dividend less than zero; add divisor, right shifted, to dividend.
08		(4) Dividend still less than zero; decrement lower digit of quotient counter.
07	99 99 46 9+	(5) Repeat steps 3 and 4.
06	99 99 67 9+	(6) Repeat steps 3 and 4.
05	99 99 88 9+	(7) Repeat steps 3 and 4.

Quotient Counter	Operand –1	Operation
05	05 00 09 9+	(8) Repeat step 3. Dividend greater than zero; do not decrement quotient counter. Store quotient count in operand 1.
	05 00 09 9+ – 2 1 05 00 07 8+	(9) Subtract divisor from new trial dividend.
1–		(10) a. Store appropriate sign of quotient in lower digit of quotient counter. b. Dividend still greater than zero; increment upper digit of quotient counter.
2–	05 00 05 7+	(11) Repeat steps 9, 10 b.
3–	05 00 03 6+	(12) Repeat steps 9, 10 b.
4–	05 00 01 5+	(13) Repeat steps 9, 10 b.
4–	05 99 99 4+	(14) Repeat step 9. Do not increment quotient counter, since dividend is less than zero.
4–	05 99 99 4+ + 2 1 05 4– 01 5+ $\underbrace{05\ 4-}_{\text{Quotient}}\ \underbrace{01\ 5+}_{\text{Remainder}}$	(15) Restore dividend. Store quotient count and sign in appropriate operand 1 location.

3.3.4. Branch Instructions

The branch instructions, which have the RX instruction format, are used to change the sequence in which program instructions are executed.

3.3.4.1. Branch-On-Condition Instruction (BC)

The branch-on-condition instruction tests the condition code stored in the FAS field of privileged storage, and initiates a program sequence change (branch) if a specified condition is met. The condition to be tested is specified by the M1 (R1) field of the instruction. If the condition is met, the branch is initiated by transferring the B1D1 field of the instruction to the FAP field of privileged storage.

The M1 bits are used to test the condition codes as follows:

M1 Bit	Test Code Condition	
8	0_{10}	00_2
9	1_{10}	01_2
10	2_{10}	10_2
11	3_{10}	11_2

3.3.4.2. Branch and Link Instruction (BAL)

The branch and link instruction unconditionally initiates a program sequence branch. The instruction causes the address of the next sequential instruction (FAP) to be stored in the register specified by the R1 field of the instruction; it then transfers the contents of the B2D2 field to FAP.

3.3.5. State Control (Privileged and Special) Instructions

The state control (privileged and special) instructions use the SI format and modify or store the processor or input/output fields of privileged storage. These instructions are not directly available to a programmer, but are generated by an assembler routine to provide overall (input/output) control for one program or several concurrent programs.

3.3.5.1. Load Program State Control Instruction (LPSC)

The load program state control (privileged) instruction replaces or modifies the processor or input/output word and directs the instruction sequence to proceed either under control of the processor or as specified by input/output control. This instruction is also used to restrict the alter and display functions of storage location 4 (04), or to remove the restriction. It should be noted that when the Restrict Alter display is set, only location 4 (04) can be altered but any location of main storage can be displayed. The various functions of the load program state control instruction are specified by the I2 field of the instruction as follows:

Bit 8	Bit 9	Modification
0	0	No program state control modification.
0	1	Load full program state control word.
1	0	Clear ASCII bit to zero, enter EBCDIC mode.
1	1	Set ASCII bit to one, enter ASCII mode.

Bit 10 = 1	Load or modify input/output program state control word.
Bit 10 = 0	Load or modify processor program state control word.
Bit 11 = 1	Next instruction is under control of processor program state control.
Bit 11 = 0	Next instruction is under control of input/output program state control.
Bit 12 = 1	Restrict display and alter function to location 4 (04).
Bit 13 = 1	Remove restriction on display and alter function.

3.3.5.2. Store Program State Control Instruction (SPSC)

The store program state control (privileged) instruction transfers the program state control word specified by the I2 field (bit 10) of the instruction to the storage locations specified by B1D1 field of the instruction. If bit 10 = 0, the processor program state control word is stored; if bit 10 = 1, the input/output program state control word is stored.

3.3.5.3. Supervisor Request Call Instruction (SRC)

The supervisor request call (special) instruction stores the I2 field of the instruction in the SRC field of the input/output program state control word in privileged storage and sets an interrupt request.

The interrupt request is granted immediately if the processor mode is in effect, and no interrupt requests are generated by I/O devices during the execution of the SCR instruction. If operation is in the input/output mode or if an I/O interrupt request becomes active, the SCR interrupt request is stored in the store interrupt pending flip-flop (ASIRF FF).

3.3.6. Input/Output Instruction

The input/output instruction provides a means of initiating the operation of all peripheral devices associated with the processor, and of determining the status of each device. These instructions use the SI format, modified to accomplish the required functions.

3.3.6.1. Execute I/O Instruction (XIOF)

The execute I/O instruction causes the device whose address is specified by the I2 (DA) field of the instruction to perform the function specified by the least significant byte of the B1D1 field of the instruction.

Bit 27 of the instruction is reserved to inhibit interrupts when the specified function is ended or if an error occurs during the execution of that function.

A condition code is stored in the FAS field of privileged storage to indicate whether the function was accepted or rejected by the selected device.

3.3.6.2. Test I/O Instruction (TIO)

The test I/O instruction tests the status indicators of the synchronizer associated with the device specified by the I2 (DA) field of the instruction. The contents of the status indicators is stored in the memory location specified by the B1D1 field of the instruction, and an appropriate condition code is stored in the FAS field of the privileged storage. If the device specified is not busy, the status indicators in the synchronizer, including interrupt request, are cleared as the status byte is transferred to the storage area specified in the instruction.

APPENDIX A. INSTRUCTIONS, FORMATS, CODES

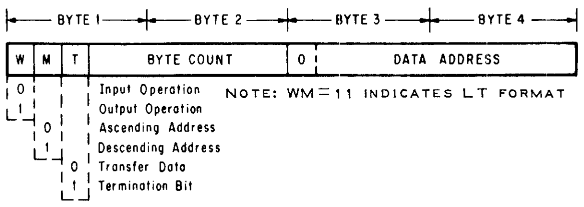
INSTRUCTION REPERTOIRE

FORMAT	OP CODE	MNEMONIC	INSTRUCTION	OPERATION
			BINARY	
	40	STH	Store Halfword	$(R_1) \rightarrow$ B2D2
	48	LH	Load Halfword	$(B2D2) \rightarrow R_1$
RX	49	CH	Compare Halfword	(R_1) : (B2D2); set CC
SI	A6	AI	Add Immediate	I_2 (Sign Ext)+(B1D1) $\rightarrow$ B1D1(2 Bytes); set CC
	AA	AH	Add Halfword	$(R_1)+(B2D2) \rightarrow R_1$; set CC
RX	AB	SH	Subtract Halfword	$(R_1)-(B2D2) \rightarrow R_1$; set CC
			LOGICAL	
	91	TM	Test Under Mask	I_2 (Mask) : B1D1; set CC
	92	MVI	Move Immediate	$I_2 \rightarrow$ B1D1
	94	NI	AND Immediate	$I_2 \odot$ (B1D1) $\rightarrow$ B1D1 ; set CC
SI	95	CLI	Compare Immediate	(B1D1) : I_2 ; set CC
	96	OI	OR Immediate	$I_2 \oplus$ (B1D1) $\rightarrow$ B1D1 ; set CC
	A9	HPR	Halt & Proceed	Display B1D1
	D1	MVN	Move Numeric	(B2D2) Lower $\rightarrow$ B1D1 Lower (Zones Unchanged)
	D2	MVC	Move Character	(B2D2) $\rightarrow$ B1D1
	D4	NC	AND Character	(B2D2) $\odot$ (B1D1) $\rightarrow$ B1D1; set CC
SS	D5	CLC	Compare Logical	(B1D1): (B2D2); set CC
	D6	OC	OR Character	(B2D2) $\oplus$ (B1D1) $\rightarrow$ B1D1; set CC
	DC	TR	Translate	[(B1D1) + B2D2] $\rightarrow$ B1D1
	DE	ED	Edit	Unpack & Expand OP_2 $(L_1+1) \rightarrow OP_1$; Control Pat in OP_1 MSB of OP_1 = Fill Char, DSB=20_{16}, SSB=21_{16}, FSB=22_{16}
			DECIMAL	
	F1	MVO	Move with Offset	(B2D2) $\rightarrow$ B1D1 (Shift left four bits)*
	F2	PACK	Pack	(B2D2) Packed $\rightarrow$ B1D1*
	F3	UNPK	Unpack	(B2D2) Unpk $\rightarrow$ B1D1 (Generate zones)*
	F8	ZAP	Zero and Add	(B2D2) $\rightarrow$ B1D1 Dec. $(L1 \geq L2)$*; set CC
	F9	CP	Compare Decimal	(B1D1):(B2D2) Dec. $(L1 \geq L2)$*; set CC
SS	FA	AP	Add Decimal	(B1D1)+(B2D2) $\rightarrow$ B1D1 Dec. $(L1 \geq L2)$* ; set CC
	FB	SP	Subtract Decimal	(B1D1)−(B2D2) $\rightarrow$ B1D1 Dec. $(L1 \geq L2)$* ; set CC
	FC	MP	Multiply Decimal	$OP_2 \times OP_1 \rightarrow OP_1$; L_1+1 = Size of product; L_2+1 = Size of Mult.; OP_1 must have L_2+1 leading dec. zero*
	FD	DP	Divide Decimal	$OP_1 \div OP_2 \rightarrow OP_1$; $L_1 > L_2$; Quotient in L_1-L_2 MSB of OP_1*
			BRANCH	
RX	45	BAL	Branch and Link	(FAP) $\rightarrow R_1$, B2D2 $\rightarrow$ FAP
	47	BC	Branch on Condition	If match, (B2D2) $\rightarrow$ FAP; see chart
			PRIVILEGED (State Control)	
SI	A0	SPSC	Store State	(PSC) $\rightarrow$ B1D1
	A8	LPSC	Load State	(B1D1) $\rightarrow$ PSC
			SPECIAL (State Control)	
SI	A1	SRC	Supervisor Call	$I_2 \rightarrow 11_{(16)}$; set interrupt
			INPUT/OUTPUT	
SI	A4	XIOF	Execute I/O	Function $\rightarrow$ Device; set CC
	A5	TIO	Test I/O	Test Device Status $\rightarrow$ B1D1; set CC

*Data processed right to left $\oplus$ = OR − = Minus $\odot$ = AND
: = Compare R = Prog. Reg. ( ) = Contents of CC = Condition Code + = Plus

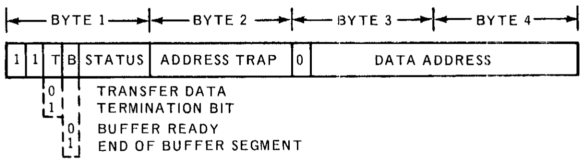
MULTIPLEXER CHANNEL

BASIC BUFFER CONTROL WORD



NOTE: WM=11 INDICATES LT FORMAT

LT BUFFER CONTROL WORD



STATUS BYTE

Bit	0	1	2	3	4	5	6	7
Detail	Att.	Stat. Mod.	Cont. Unit End	Busy	Chan. End	Dev. End	Unit Chk.	Unit Except.

CHANNEL ERROR STATUS

Mem. Loc.	Bit Pos.	Signal Function	Mem. Loc.	Bit Pos.	Signal Function
001D	0	Interface Error	001E	0	Status In
	1	Device Address Parity Error		1	Service Out
	2	Bus in Parity Error		2	Service In
	3	Address Out		3	TIME OUT REQUEST
	4	Select Out		4	Suppress Out
	5	Operational In		5	Select In
	6	Address In		6	Terminate/KØ FF
	7	Command Out		7	Input Direction/K1 FF
			001F	0−7	Device Address Register

MULTIPLEXER CHANNEL COMMANDS

Function	XF Code Bits								
	P	0	1	2	3	4	5	6	7
Test	P	X	X	X	X	0	0	0	0
Sense	P					0	1	0	0
Write	P							0	1
Read	P		COMMAND DETAILS					1	0
Control	P							1	1
Read Backward	P					1	1	0	0
Reserved for Chan. Cont.	P	X	X	X	X	1	0	0	0

X = Variable to Control Units
P = Parity Bit (Odd)

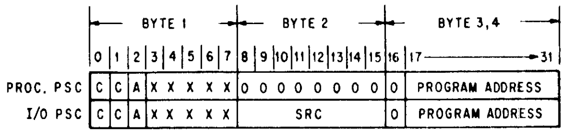
FIRST SENSE BYTE

Bit	Indication
P	Parity (Odd)
0	Command Reject
1	Intervention Required
2	Bus Out Parity
3	Equipment Check
4	Data Check
5	Data Late
6	Undefined
7	Undefined

CONDITION CODE (CC) SETTINGS

Instr.	Condition Codes/Conditions			
	0(00)	1(01)	2(10)	3(11)
SH (AB) AH (AA) AI (A6) AP (FA) SP (FB)	Result=Zero ↓	Result=Neg. ↓	Result Positive ↓	Overflow ↓
CH (49) CLI (95) CLC (D5) CP (F9)	$(R_1)=(OP_2)$ $(OP_1)=I_2$ $(OP_1)=(OP_2)$ $(OP_1)=(OP_2)$	$(R_1)<(OP_2)$ $(OP_1)<I_2$ $(OP_1)<(OP_2)$ $(OP_1)<(OP_2)$	$(R_1)>(OP_2)$ $(OP_1>I_2$ $(OP_1)>(OP_2)$ $(OP_1)>(OP_2)$	
ZAP (F8)	$(OP_2)=\emptyset$	(OP_2) Neg.	(OP_2) Pos.	
NI (94) NC (D4) OI (96) OC (D6)	Result=Zero ↓	Result≠Zero ↓		
TM (91)	No match or mask=Ø	Partial match		Full match
XIOF (A4)	Accepted	Status Pending	Busy	Rejected
TIO (A5)	Available	Valid Status	Busy	Rejected

PROGRAM STATE CONTROL STORAGE


BYTE 1
BYTE 2
BYTE 3,4
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 31
PROC. PSC
C C A X X X X X 0 0 0 0 0 0 0 0 0 PROGRAM ADDRESS
I/O PSC
C C A X X X X X SRC 0 PROGRAM ADDRESS

CC = Condition Code A = 1 = ASCII
A = ASCII Control Code A = 0 = EBCDIC

PROGRAM STATE CONTROL

Load Action*			PSC Selection		Next Instr. Control *		Alter/Display Action *		
Instr. Bit		Action	Instr. Bit	PSC	Instr. Bit	Control PSC	Instr. Bit		Action
8	9		10		11		12	13	
0	0	None	0	Proc.	0	Proc.	1	0	Restrict
0	1	Fullword	1	I/O	1	I/O	0	1	Remove Restriction
1	0	ASCII Off							
1	1	ASCII On							

*Load State Instruction only

INTERNAL I/O COMMANDS

I/O Device	DA	Instruction Bit/Function							
		24	25	26	27	28	29	30	31
Card Reader	1				Inh. Int. ↓		Image ↓	Read ↓	
Read/Punch	2				↓	Stkr Sel.			Punch
Printer	3	48 Char. bar (24=1)	Numeric					*	*

* 30 and 31 ⟹ Space; 31 only ⟹ Print and Space

BRANCH/CONDITION

Condition Code	0	1	2	3
BC Instr. Bit	8	9	10	11

	SIGN		ZONE
	+	−	
ASCII	1010	1011	0101
EBCDIC	1100	1101	1111

PRINT/MEMORY LOCATIONS

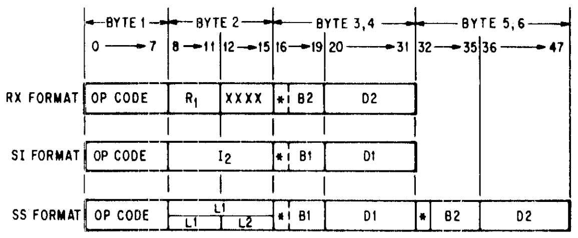
Col.	Decimal	0 1 2 3 4 5 6 7 8 9	1 0 1 2 3 4 5 6 7 8 9	2 0 1 2 3 4 5 6 7 8 9	3 0 1 2 3
Mem. Loc.	Hexadecimal	8 0 1 2 3 4 5 6 7 8 9 A B C D E F	9 0 1 2 3 4 5 6 7 8 9 A B C D E F	A 0	

Col.	Decimal	3 4 5 6 7 8 9	4 0 1 2 3 4 5 6 7 8 9	5 0 1 2 3 4 5 6 7 8 9	6 0 1 2 3 4 5 6
Mem. Loc.	Hexadecimal	A 1 2 3 4 5 6 7 8 9 A B C D E F	B 0 1 2 3 4 5 6 7 8 9 A B C D E F	C 0 1	

Col.	Decimal	6 7 8 9	7 0 1 2 3 4 5 6 7 8 9	8 0 1 2 3 4 5 6 7 8 9	9 0 1 2 3 4 5 6 7 8 9
Mem. Loc.	Hexadecimal	C 2 3 4 5 6 7 8 9 A B C D E F	D 0 1 2 3 4 5 6 7 8 9 A B C D E F	E 0 1 2	

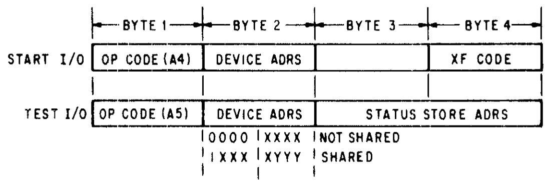
Col.	Decimal	10 0 1 2 3 4 5 6 7 8 9	11 0 1 2 3 4 5 6 7 8 9	12 0 1 2 3 4 5 6 7 8 9	13 0 1 2
Mem. Loc.	Hexadecimal	E 3 4 5 6 7 8 9 A B C D E F	F 0 1 2 3 4 5 6 7 8 9 A B C D E F	10 0 1 2 3	

INSTRUCTION FORMAT



R1 = Adrs of program register
B1D1 = Adrs of operand one
B2D2 = Adrs of operand two
I = Immediate operand
L1 = One less than length of operand one
L2 = One less than length of operand two
* = Most sig. bit of B1 or B2 Field = 1 indicates indexing of OP_1 or OP_2

I/O INSTRUCTION FORMAT



XXXX = Subchannel Address
YYY = Device Number

PRINTABLE CHARACTERS

Zone Bits/Char. 63 Char. Bar				Numeric Bits 63 and 48 Char. Bar				Zone Bits/Char. 48 Char. Bar			
Bits 2 and 3								Bits 2 and 3			
00	01	10	11	Bits				00	01	10	11
Character				4	5	6	7	Character			
	&	-	0	0	0	0	0	A		Q	0
A	J	/	1	0	0	0	1	B		R	1
B	K	S	2	0	0	1	0	C		S	2
C	L	T	3	0	0	1	1	D		T	3
D	M	U	4	0	1	0	0	E		U	4
E	N	V	5	0	1	0	1	F		V	5
F	O	W	6	0	1	1	0	G		W	6
G	P	X	7	0	1	1	1	H		X	7
H	Q	Y	8	1	0	0	0	I		Y	8
I	R	Z	9	1	0	0	1	J		Z	9
¢	!		:	1	0	1	0	K		+	.
.	$	,	#	1	0	1	1	L		&	$
<	*	%	@	1	1	0	0	M		%	*
(	)	_	'	1	1	0	1	N		#	-
+	;	>	=	1	1	1	0	O		@	/
|	¬	?	"	1	1	1	1	P		'	,

COMPRESSED CARD CODE

Hexadecimal	Bit Positions	Punch Positions	
	4 5 6 7		↓
	0 1 2 3	↓	
0	0 0 0 0		
1	0 0 0 1	3	12
2	0 0 1 0	4	11
3	0 0 1 1	1	12,11
4	0 1 0 0	5	0
5	0 1 0 1	2	12,0
6	0 1 1 0	7	11,0
7	0 1 1 1	6	12,11,0
8	1 0 0 0	9	8
9	1 0 0 1	9,3	8,12
A	1 0 1 0	9,4	8,11
B	1 0 1 1	9,1	8,12,11
C	1 1 0 0	9,5	8,0
D	1 1 0 1	9,2	8,12,0
E	1 1 1 0	9,7	8,11,0
F	1 1 1 1	9,6	8,12,11,0

I/O STATUS BYTE

I/O Device/Status Indication*			
Printer (DA=3)	Card Punch or Read Punch (DA=2)	Card Reader (DA=1)	Status Byte Bit*
Abnormal or Not Ready	Stkr Jam, Intlk, Punch Entry, or Exit Chk Error	Stkr Jam, Cont. Par. Err., or Photocell Chk Err.	0
Paper Runaway		Misfeed, Not Ready, Hopper Empty, or Stacker Full	1
Memory Overload	Punch Chk Err.		2
Data Par. or Cont. Par. Err.	Data Par. or Cont. Par. Err.		3
Bar Switch in Err.	Photocell Chk Err.	TRIPLE STROBE ERR.	4
Interrupt Pending	Interrupt Pending	Interrupt Pending	5
Form Overflow	Hopper Empty or Stkr Full		6
Paper Low			7

*All Ø's ⟶ function performed as specified

9300 EXTENDED OPERATION CODES

MNEMONIC	FUNCTION	HEXADECIMAL OPERATION CODE, R1	FORMAT
B	BRANCH	47 F	RX
NOP	NO OPERATION	47 0	RX
	USED AFTER COMPARISON INSTRUCTIONS		
BH	BRANCH IF HIGH	47 2	RX
BL	BRANCH IF LOW	47 4	RX
BE	BRANCH IF EQUAL	47 8	RX
BNH	BRANCH IF NOT HIGH	47 D	RX
BNL	BRANCH IF NOT LOW	47 B	RX
BNE	BRANCH IF NOT EQUAL	47 7	RX
	USED AFTER TEST UNDER MASK INSTRUCTIONS		
BO	BRANCH IF ALL ONES	47 1	RX
BZ	BRANCH IF ALL ZEROS	47 8	RX
BM	BRANCH IF MIXED	47 4	RX
BNO	BRANCH IF NOT ALL ONES	47 E	RX
BNZ	BRANCH IF NOT ALL ZEROS	47 7	RX
BNM	BRANCH IF NOT MIXED	47 B	RX
	USED AFTER ARITHMETIC INSTRUCTIONS		
BO	BRANCH IF OVERFLOW	47 1	RX
BZ	BRANCH IF ZERO	47 8	RX
BM	BRANCH IF MINUS	47 4	RX
BP	BRANCH IF POSITIVE	47 2	RX
BNO	BRANCH IF NO OVERFLOW	47 E	RX
BNZ	BRANCH IF NOT ZERO	47 7	RX
BNM	BRANCH IF NOT MINUS	47 B	RX
BNP	BRANCH IF NOT POSITIVE	47 D	RX

CONSTANT CHARACTERISTICS

CONSTANT TYPE	EXPLICIT LENGTH	IMPLICIT LENGTH	TRUNCATION OR PADDING	VALUE PADDED
C	variable 1 – 256	maximum 256	on right side	blanks
X	variable 1 – 256	maximum 256	on left side	hexadecimal 0
P*	variable 1 – 16	maximum 16	on left side	hexadecimal 0
Z*	variable 1 – 16	maximum 16	on left side	EBCDIC 0
H*	variable 1 – 2	2	on left side	hexadecimal 0
Y	variable 1 – 2	2	on left side	hexadecimal 0
S*	2	2	none	none

*9300 only.

APPENDIX B. HEXADECIMAL TABLES

B.1. HEXADECIMAL AND DECIMAL CONVERSION

To convert to a decimal number, find the corresponding decimal number for each hexadecimal representation. Then find the sum of the decimal values. To convert to a hexadecimal, subtract the value in the table from the original decimal number that produces the smallest difference and note the hexadecimal equivalent. Repeat this procedure using the difference obtained in a previous calculation until a zero difference is obtained. The resulting hexadecimal representation (including zeros from columns not used) are the equivalent of the decimal number.

BYTE				BYTE				BYTE			
0123		4567		0123		4567		0123		4567	
HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
0	0	0	0	0	0	0	0	0	0	0	0
1	1,048,576	1	65,536	1	4,096	1	256	1	16	1	1
2	2,097,152	2	131,072	2	8,192	2	512	2	32	2	2
3	3,145,728	3	196,608	3	12,288	3	768	3	48	3	3
4	4,194,304	4	262,144	4	16,384	4	1,024	4	64	4	4
5	5,242,880	5	327,680	5	20,480	5	1,280	5	80	5	5
6	6,291,456	6	393,216	6	24,576	6	1,536	6	96	6	6
7	7,340,032	7	458,752	7	28,672	7	1,792	7	112	7	7
8	8,388,608	8	524,288	8	32,768	8	2,048	8	128	8	8
9	9,437,184	9	589,824	9	36,864	9	2,304	9	144	9	9
A	10,485,760	A	655,360	A	40,960	A	2,560	A	160	A	10
B	11,534,336	B	720,896	B	45,056	B	2,816	B	176	B	11
C	12,582,912	C	786,432	C	49,152	C	3,072	C	192	C	12
D	13,631,488	D	851,968	D	53,248	D	3,328	D	208	D	13
E	14,680,064	E	917,504	E	57,344	E	3,584	E	224	E	14
F	15,728,640	F	983,040	F	61,440	F	3,840	F	240	F	15
6		5		4		3		2		1	

Examples:

Hexadecimal 6D03A1 to Decimal

```
600 000 = 6291456
 D0 000 =  851968
  0 000 =    0000
    300 =     768
     A0 =     160
      1 =       1
6D0 3A1   7144353  Result
```

Decimal 27043 to Hexadecimal

```
27043
24576* = 6000
 2467
 2304* =  900
  163
  160*     A0
    3  =    3
         69A3  Result
```

*Subtract largest number; see table

B.2. HEXADECIMAL ADDITION

X	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
1	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11
2	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12
3	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13
4	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14
5	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15
6	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16
7	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17
8	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18
9	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19
A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A
B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B
C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E
F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20

B.3. HEXADECIMAL MULTIPLICATION

X	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E	20
3	06	09	0C	0F	12	15	18	1B	1E	21	24	27	2A	2D	30
4	08	0C	10	14	18	1C	20	24	28	2C	30	34	38	3C	40
5	0A	0F	14	19	1E	23	28	2D	32	37	3C	41	46	4B	50
6	0C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A	60
7	0E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69	70
8	10	18	20	28	30	38	40	48	50	58	60	68	70	78	80
9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87	90
A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96	A0
B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5	B0
C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4	C0
D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3	D0
E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2	E0
F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1	F0
10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	100

APPENDIX C. CHARACTER GRAPHIC SET CODES AND PRINTER SYNCHRONIZER CODE

C.1. EBCDIC AND 80-COLUMN PUNCHED CARD CODES

80- COLUMN CARD CODE	CHARACTER	EBCDIC
12–1	A	1100 0001
12–2	B	1100 0010
12–3	C	1100 0011
12–4	D	1100 0100
12–5	E	1100 0101
12–6	F	1100 0110
12–7	G	1100 0111
12–8	H	1100 1000
12–9	I	1100 1001
11–1	J	1101 0001
11–2	K	1101 0010
11–3	L	1101 0011
11–4	M	1101 0100
11–5	N	1101 0101
11–6	O	1101 0110
11–7	P	1101 0111
11–8	Q	1101 1000
11–9	R	1101 1001
0–2	S	1110 0010
0–3	T	1110 0011
0–4	U	1110 0100
0–5	V	1110 0101
0–6	W	1110 0110
0–7	X	1110 0110
0–8	Y	1110 1000
0–9	Z	1110 1001
0	0	1111 0000
1	1	1111 0001
2	2	1111 0010
3	3	1111 0011
4	4	1111 0100
5	5	1111 0101
6	6	1111 0110
7	7	1111 0111
8	8	1111 1000
9	9	1111 1001
	ƀ (space)	0100 0000
12–8–2	¢	0100 1010
12–8–4	<	0100 1100
12–8–5	(	0100 1101
12–8–6	+	0100 1110
12–8–7	|(vertical line)	0100 1111
12	&	0101 0000
11–8–2	!	0101 1010
11–8–3	$	0101 1011
11–8–4	*	0101 1100
11–8–5	)	0101 1101
11–8–6	;	0101 1110
11–8–7	¬ (not)	0101 1111
11	–	0110 0000
0–1	/	0110 0001
0–8–4	%	0110 1100
0–8–5	_ (underscore)	0110 1101
0–8–6	>	0110 1110
0–8–7	?	0110 1111
8–2	:	0111 1010
8–3	#	0111 1011
8–4	@	0111 1100
8–5	'(apostrophe)	0111 1101
8–6	=	0111 1110
8–7	''	0111 1111
0–8–3	, (comma)	0110 1011
12–8–3	.	0100 1011

C.2. 63-CHARACTER GRAPHIC SET CARD CODE AND 6-BIT PRINTER SYNCHRONIZER CARD

80-COLUMN CARD CODE	CHARACTER	63-CH GRAPHICS 6 BIT PRINTER CODE	80-COLUMN CARD CODE	CHARACTER	63-CH GRAPHICS 6 BIT PRINTER CODE
12–1	A	000001	12	&	010000
12–2	B	000010	11	−	100000
12–3	C	000011	0–1	/	100001
12–4	D	000100	12–8–3	.	001011
12–5	E	000101	11–8–3	$	011011
12–6	F	000110	0–8–3	,	101011
12–7	G	000111	8–3	#	111011
12–8	H	001000	12–8–4	<	001100
12–9	I	001001	11–8–4	*	011100
11–1	J	010001	0–8–4	%	101100
11–2	K	010010	8–4	@	111100
11–3	L	010011	12–8–5	(	001101
11–4	M	010100	11–8–5	)	011101
11–5	N	010101	0–8–5	(Underscore)	101101
11–6	O	010110	8–5	'	111101
11–7	P	010111	12–8–6	+	001110
11–8	Q	011000	11–8–6	;	011110
11–9	R	011001	0–8–6	>	101110
0–2	S	100010	8–6	=	111110
0–3	T	100011	12–8–7	|	001111
0–4	U	100100	11–8–7	¬	011111
0–5	V	100101	0–8–7	?	101111
0–6	W	100110	8–7	"	111111
0–7	X	100111	12–8–2	¢	001010
0–8	Y	101000	11–8–2	!	011010
0–9	Z	101001	0–8–2	Not Assigned	101010
			8–2	:	111010
0	0	110000			
1	1	110001			
2	2	110010			
3	3	110011			
4	4	110100			
5	5	110101			
6	6	110110			
7	7	110111			
8	8	111000			
9	9	111001			

A second 63-character type bar, containing 6 different special symbols is available for the UNIVAC 9200/9200 II/9300/9300 II Systems processor printer and are given as follows:

STANDARD PRINTER BAR	80-COLUMN CARD CODE	6-BIT PRINTER CODE	SPECIAL ORDER PRINTER BAR
— Underline	0-5-8	101101	≠ Not Equal
| Absolute	12-8-7	001111	] Right Bracket
¬ Logical Not	11-8-7	011111	Δ Delta
" Quotes	8-7	111111	⌑ Lozenge
¢ Cent Sign	12-8-2	001010	[Left Bracket
Not Assigned	0-8-2	101010	\ Back Slash

Note that the symbols on the special bar take on the 80-column card codes and 6-bit printer codes already assigned. It is therefore the responsibility of programming to perform, by means of suitable software, any required translation.

C.3. 48-CHARACTER GRAPHIC SET CARD CODE AND 6-BIT PRINTER SYNCHRONIZER CODE

80-COLUMN CARD CODE	CHARACTER	48-CH GRAPHICS 6-BIT PRINTER CODE	80-COLUMN CARD CODE	CHARACTER	48-CH GRAPHICS 6-BIT PRINTER CODE
12-1	A	000000	0	0	110000
12-2	B	000001	1	1	110001
12-3	C	000010	2	2	110010
12-4	D	000011	3	3	110011
12-5	E	000100	4	4	110100
12-6	F	000101	5	5	110101
12-7	G	000110	6	6	110110
12-8	H	000111	7	7	110111
12-9	I	001000	8	8	111000
11-1	J	001001	9	9	111001
11-2	K	001010			
11-3	L	001011			
11-4	M	001100	12	&	101011
11-5	N	001101	11	–	111101
11-6	O	001110	0-1	/	111110
11-7	P	001111	12-8-3	.	111010
11-8	Q	100000	11-8-3	$	111011
11-9	R	100001	0-8-3	,	111111
0-2	S	100010	8-3	#	101101
0-3	T	100011	11-8-4	*	111100
0-4	U	100100	0-8-4	%	101100
0-5	V	100101	8-4	@	101110
0-6	W	100110	8-5	'	101111
0-7	X	100111	12-8-6	+	101010
0-8	Y	101000			
0-9	Z	101001			

The UNIVAC 9300/9300 II Systems processor printer does not use a 48-character print set.

C.4. 16-CHARACTER GRAPHIC SET CARD CODE AND 6-BIT PRINTER SYNCHRONIZER CODE

80-COLUMN CARD CODE	CHARACTER	16-DIGIT GRAPHIC 6-BIT PRINT CODE
0	0	110000
1	1	110001
2	2	110010
3	3	110011
4	4	110100
5	5	110101
6	6	110110
7	7	110111
8	8	111000
9	9	111001
12–8–3	.	111010
11–8–3	$	111011
11–8–4	*	111100
11	– (dash or minus)	111101
0–1	/	111110
0–8–3	,	111111

The UNIVAC 9200/9200 II Systems Processor printer does not use a 16-character print set.

APPENDIX D. POWERS OF 2; POWERS OF 16

POWERS OF 2

2^n	n	2^n	n
1	0	262 144	18
2	1	525 288	19
4	2	1 048 576	20
8	3	2 097 152	21
16	4	4 194 304	22
32	5	8 388 608	23
64	6	16 777 216	24
128	7	33 554 432	25
256	8	67 108 864	26
512	9	134 217 728	27
1 024	10	268 435 456	28
2 048	11	536 870 912	29
4 096	12	1 073 741 824	30
8 192	13	2 147 483 648	31
16 384	14	4 294 967 296	32
32 768	15	8 589 934 592	33
65 536	16	17 179 869 184	34
131 072	17	34 359 738 368	35

POWERS OF 16

16^n	n
1	0
16	1
256	2
4 096	3
65 536	4
1 048 576	5
16 777 216	6
268 435 456	7
4 294 967 296	8
68 719 476 736	9
1 099 511 627 776	10
17 592 186 044 416	11
281 474 976 710 656	12
4 503 599 627 370 496	13
72 057 594 037 927 936	14
1 152 921 504 606 846 976	15

INDEX

CUT

Comments concerning this manual may be made in the space provided below. Please fill in the requested information.

System: ____________________

Manual Title: ____________________

UP No: __________ Revision No: __________ Update: __________

Name of User: ____________________

Address of User: ____________________

Comments:

FOLD

FIRST CLASS
PERMIT NO. 21
BLUE BELL, PA.

BUSINESS REPLY MAIL NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY

UNIVAC

P.O. BOX 500

BLUE BELL, PA. 19422

ATTN: SYSTEMS PUBLICATIONS DEPT.

FOLD

CUT