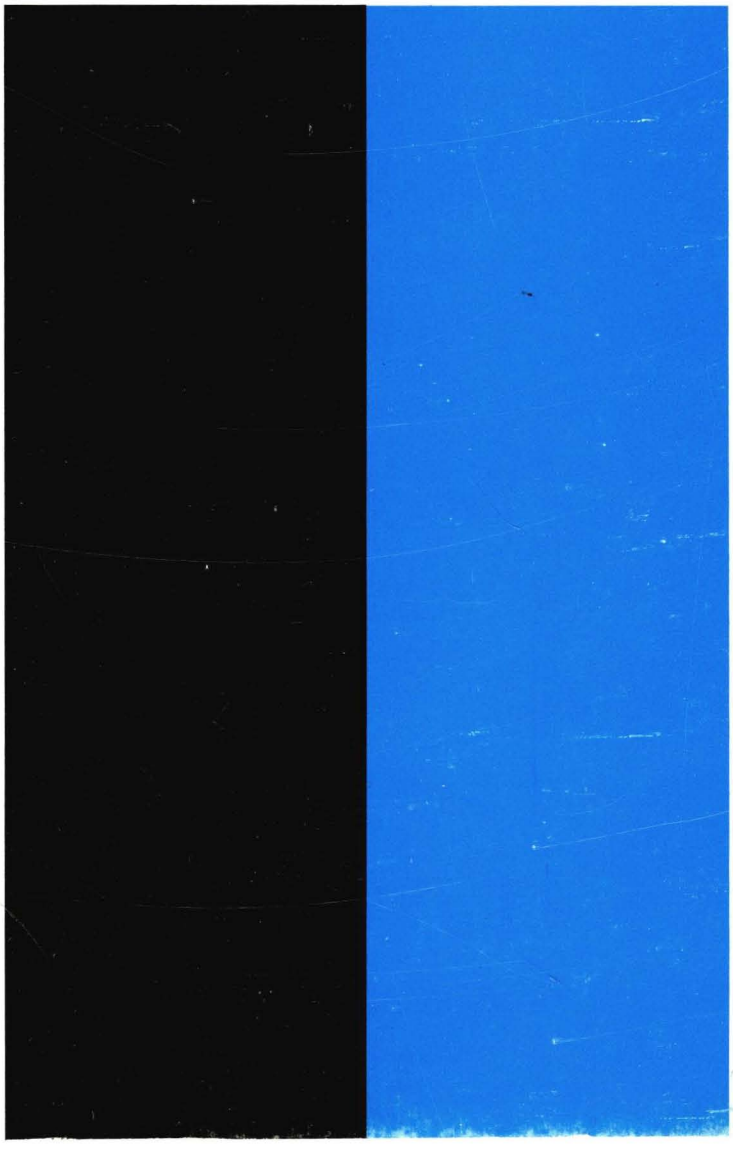
SPERRY RAND

UNIVAC 1110 SYSTEM



SYSTEM
DESCRIPTION

This manual describes a UNIVAC®system. It is as complete and accurate at the time of publication as is feasible by current documentation techniques. The Univac Division will issue complete revisions of this manual when necessary. The Univac Division reserves the right to make such additions, corrections, and/or deletions as, in the judgment of the Univac Division, are required by the development of its systems. To assure that you have the current version of this manual and for the current status of the system, contact your local Univac Representative.

PRINTED IN U.S.A.

CONTENTS

FIGURES

TABLES

1. INTRODUCTION

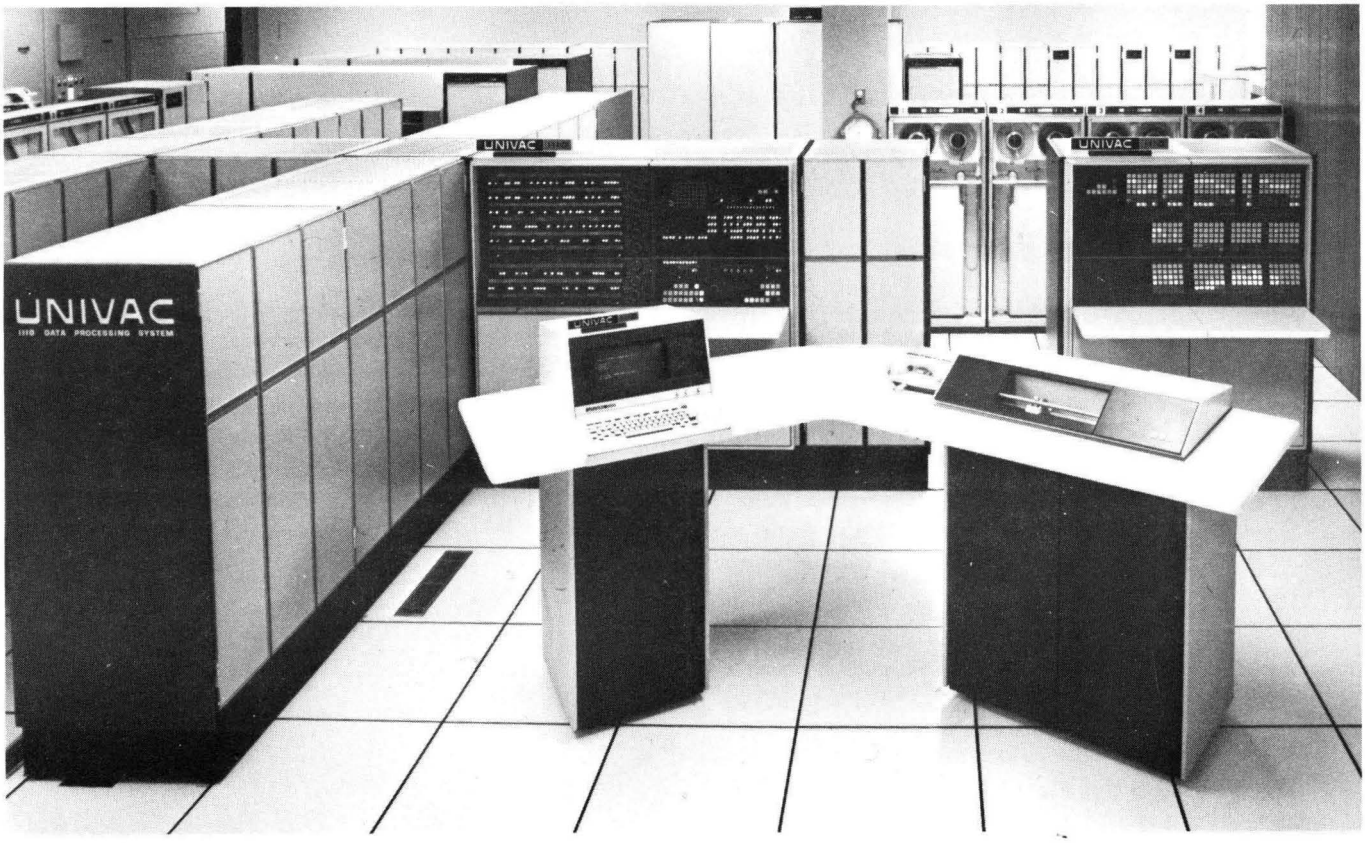


Figure 1–1. UNIVAC 1110 System

The UNIVAC 1110 System (see Figure 1–1) is the logical successor to the UNIVAC 1106 System and the UNIVAC 1108 Multi-Processor System. Designed to enhance the efficiency of the UNIVAC 1100 Series, the UNIVAC 1110 System offers dependable and highly effective processing in real time, demand, and batch modes, and excells in multiprocessing time-sharing applications.

Although the 1106 and 1108 Systems differ from the 1110 in hardware design, software compatibility is maintained. All components of the 1110 System (Command/Arithmetic Units, input/output units, storage units, and peripherals) are controlled by an 1100 Series-compatible executive system. Industry standard language processors and application software are provided. The uniqueness and flexible design of the UNIVAC 1110 System offer the user a variety of custom-tailored systems.

The UNIVAC 1110 System places great emphasis on achieving high throughput performance by obtaining maximum efficiency in a refined multiprocessor system.

2. SYSTEM DESCRIPTION

2.1. GENERAL

The UNIVAC 1110 System is a general purpose, high performance multiprocessor system incorporating the latest advances in computer design, system organization, and programming technology. The various components of the 1110 System are designed as separate logical units providing maximum functional modularity. The multiprocessing capabilities are an integral part of the system; each of the Command/Arithmetic Units can perform numerous tasks simultaneously under control of a common executive. The flexible modular structure enables a user to tailor a system to his individual requirements. Principal features of the 1110 System are:

- Common resource systems organization
- Multiple Command/Arithmetic Units (CAU) and Input/Output Access Control Units (IOAU)
- Character manipulating instructions
- Partial-word, double-word, and full-word addressability
- System partitioning capability
- Redundancy among system components
- Two levels of directly addressable storage
- Large modular plated-wire main storage
- Large modular extended main storage
- Storage protection
- Program address relocation
- Independent Input/Output Access Units (IOAU)
- Extensive software library and language processors
- Dynamic adjustment to a mix of batch, demand, and real time modes
- Wide choice of high performance peripheral subsystems
- Independent, simultaneous communications processing

2.2. SYSTEM COMPONENTS

There are three basic groups of components used in the UNIVAC 1110 System: processors, storage, and peripheral subsystems. Each component is functionally independent and has the following properties:

- has two or more access paths;
- resolves access conflicts by priority logic;
- does not prevent continued system operation if any individual component fails;
- can be logically removed for servicing without disabling the entire system.

The 1110 System consists of seven types of components:

- Command/Arithmetic Units
- Input/Output Access Units
- System Console
- System partitioning unit
- Main storage
- Extended storage
- Peripheral subsystems

2.2.1. Command/Arithmetic Unit (CAU)

The basic 1110 System configuration consists of two Command/Arithmetic Units (CAU) and one Input/Output Access Unit. All control and arithmetic functions are executed by the two CAU's. Each CAU is a multitask instruction-stacking device capable of controlling up to four instructions at various stages of execution. Each CAU can interface with up to four main storage units by means of either an instruction path or an operand path. Dual data paths connect each unit with extended storage through a maximum of eight Multiple Access Interface (MAI) units. The data paths to main and extended storage have overlapping and interleaving capabilities. The user can specify and can change which units are to be used, thereby permitting maintenance of individual units without affecting the total system. Interrupt signals may be sent or received on one of the eight interprocessor lines.

Additional features of each CAU are:

- Capability of executing 1.8 million instructions per second
- 300-nanosecond effective basic instruction time
- Four-deep instruction stack
- 112-word general register stack (GRS)
- Character manipulation by means of byte-oriented instructions

2.2.2. Input/Output Access Unit (IOAU)

The Input/Output Access Unit (IOAU) controls all transfers of data between the peripheral devices and main and extended storage. Transfers are initiated by a CAU under program control. The IOAU includes two nonconcurrent data transfer paths, one for main storage and one for extended storage.

The IOAU consists of two sections: a control section and a section containing from 8 to 24 input/output channels. I/O data transfers may occur simultaneously with the execution of programs in the CAU.

The control section includes all logic associated with the transfer of function, data, and status words between main or extended storage and the subsystems. It also services I/O requests from either one or both of the CAU's and routes interrupts to one of the two command/arithmetic units. Interrupt routing may be specified by program.

Some outstanding features of the IOAU are:

- Aggregate transfer rate of 4 million 36-bit words per second (24 million characters)
- Externally specified index (ESI) and internally specified index (ISI) transfer modes on any channel
- Data chaining
- Interrupt tabling

2.2.3. System Console

The system console provides the means for communication with the executive system. The basic console consists of the following major components:

- The CRT/keyboard consists of a UNISCOPE 100 CRT and keyboard. The display format is 16 lines with 64 characters per line. The seven-bit ASCII character set, consisting of 95 characters plus the space, is used. The keyboard provides all of the operator controls required for generating data and initiating transfers.
- The incremental printer provides a hard copy of console messages. The cabinet containing the printer also contains the power supplies and control logic required to select the CRT, incremental printers, and facilities for the real time maintenance communication system (RTMCS). This unit also contains the interface between the console and any ISI channel on the input/output access unit.
- The real time maintenance communication system (RTMCS) interface section consists of a serializer/staticiser and an internal modem which provides the capability to perform diagnostic maintenance from a remote site by means of a telephone line.
- The fault indicator, located on the incremental printer, provides the operator with a visual indication of a fault condition in a major system component. The actual component and nature of the fault may then be determined from indicators on the maintenance panel.

  Up to five additional incremental printers may be connected to the console.

2.2.4. System Partitioning Unit (SPU)

The system partitioning unit (SPU), when included in the UNIVAC 1110 System, permits offline maintenance of units, enables the operator to logically divide the system into two or three independent systems, and initiates a recovery sequence in the event of failure. The SPU performs six functions, five under operator control and one under software control. With the SPU, the operator can:

- partition the total system into two or three smaller systems;
- isolate units and take them offline for maintenance without disrupting the rest of the system;
- function as a system monitor by observing the status of the various major components;
- perform initial load into the primary system;
- allow automatic recovery procedures if an interrupt is not received.

Under software control, the SPU presents status information to the input/output access units.

When all optional features are included, the system partitioning unit is able to interface with:

- Four Command/Arithmetic Units
- Four Input/Output Access Units
- 262K words of main storage
- Eight Multiple Access Interface (MAI) units (1048K words of extended storage)
- 48 multi-access subsystems

2.2.5. Main Storage

Main storage consists of high-speed, non-destructive readout (NDRO) plated wire storage units with nominal random read and write cycles of 320 and 520 nanoseconds, respectively. The basic 32K-word storage unit consists of four 8K modules, and may be expanded in one 32K increment to a 65K unit. A total of four 65K units provide a maximum main storage capacity of 262K words in an 1110 system. The basic 32K storage unit accommodates eight channels, servicing four of them simultaneously; a 65K unit accommodates sixteen channels, servicing eight of them simultaneously. Partial (sixth, quarter, third, and half) as well as full word operation is provided.

2.2.6. Extended Storage

The second level of directly addressable storage in the UNIVAC 1110 System is the extended storage system. The minimum extended-storage configuration consists of two units, each with a capacity of 131K 36-bit words. Extended storage capacity may be expanded up to a maximum of 1048K 36-bit words in 131K increments. Each unit has a 1.5 microsecond read/write cycle. Extended storage is connected to the system by Multiply Access Interface (MAI) units which provide up to ten channels to each storage unit.

2.2.7. Peripheral Subsystems

The UNIVAC 1110 System offers a full range of peripheral subsystems; this wide range provides the capability to satisfy many requirements. The standard Univac peripheral subsystems include:

- Multiple High Speed Printer Subsystem
- Card Subsystem
- 9000 Series Subsystem
- UNISERVO VIII-C Magnetic Tape Subsystem
- UNISERVO 12/16 Magnetic Tape Subsystem
- UNISERVO 20 Magnetic Tape Unit
- FH-432/1782 Drum Subsystem
- FASTRAND Mass Storage Subsystem
- 8414 Disc File Subsystem
- 8440 Disc File Subsystem
- Communications/Symbiont Processor (C/SP)
- Data Communications Terminal (DCT) 500
- Data Communications Terminal (DCT) 1000
- Data Communications Terminal (DCT) 2000
- UNISCOPE 100

2.2.8. 1106/1108 Peripheral Subsystems

The following peripheral subsystems, standard on UNIVAC 1106/1108 Systems, may be included in the UNIVAC 1110 System and are software supported in the same manner as on the 1106/1108 Systems:

- Communication Terminal Synchronous (CTS)
- Communications Terminal Module Controller (CTMC)
- 1004 Remote Subsystem
- UNISCOPE 300

2.3. CONFIGURATIONS

The basic 1110 processor consists of three functionally and physically independent units: two CAU's and one IOAU. The processor organization is intrinsically that of a multi-task processor and is designed for operation in a multiprogramming and multiprocessing environment. This basic processor may be expanded by adding CAU's and/or IOAU's up to a total of 4 CAU's and 4 IOAU's. The basic configuration is shown in Figure 2–1; Table 2–1 lists all fully supported configurations.

2.3.1. Minimum Peripheral Complement

The following list of peripheral equipment is the minimum available with the UNIVAC 1110 System. This minimum has been established to ensure an adequate complement for Customer Engineering and software support.

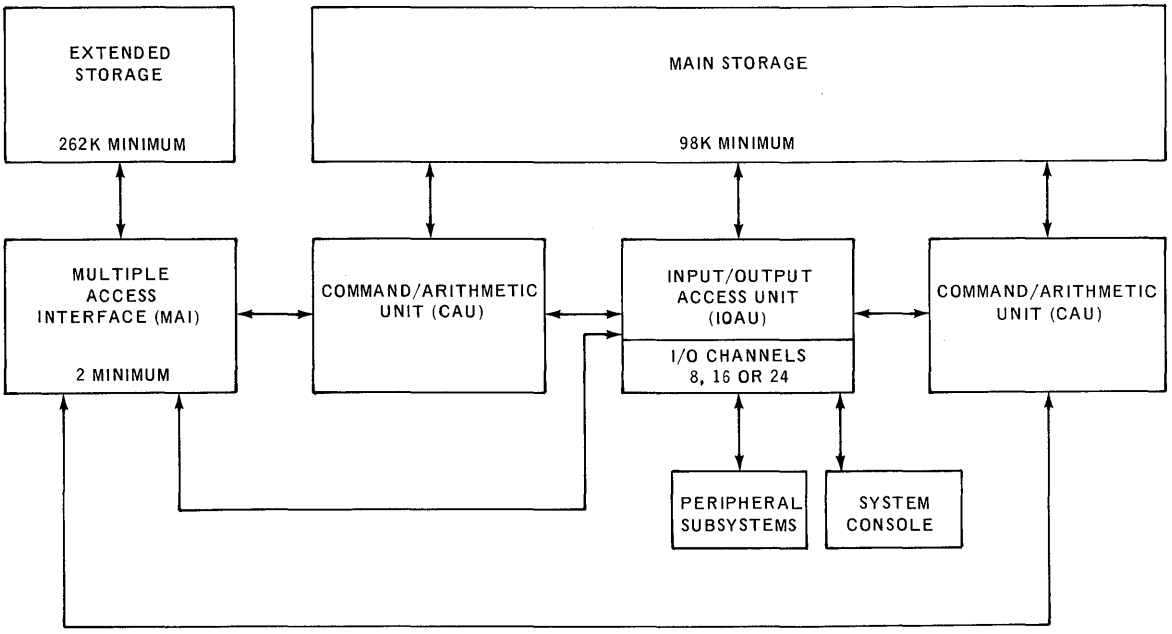


Figure 2–1. UNIVAC 1110 System, Basic Configuration (2 x 1)

Minimum Complement	Alternate
(1) Communications/Symbiont Processor (C/SP)	UNIVAC 9300 Subsystem with card reader and integral printer, or Multiple high-speed printer subsubsystem (one printer), and card subsystem with one high-speed card reader
C/SP with card reader and high speed printer	
(2) Drum Subsystem	
FH-432/1782 Drum Subsystem with two FH-432	FH-432/1782 Drum Subsystem with one FH-1782 Drum
(3) Mass Storage Subsystem	
8414 Disc File Subsystem with two 8414 Disc Drives	FASTRAND Mass Storage Subsystem with one FASTRAND II or FASTRAND III Drum
(4) Magnetic Tape Subsystem	
UNISERVO 12/16 Magnetic Tape Subsystem with four magnetic tape units	UNISERVO VIII-C Magnetic Tape Subsystem with four magnetic tape units

UNITS	CONFIGURATION			
	2 x 1	2 x 2	4 x 2	4 x 4
CAU	2	2	4	4
IOAU	1	2	2	4
STORAGE UNIT (words)	98–262K	98–262K	131–262K	131–262K
STORAGE UNIT (words)	262K–1048K	262K–1048K	262K–1048K	262K–1048K
MAI	2–8	2–8	2–8	2–8
SYSTEM CONSOLE	1–2	1–2	2	2–4
SYSTEM PARTITIONING UNIT	0–1	0–1	1	1

Table 2–1. Fully Supported Configurations

3. COMMAND/ARITHMETIC UNIT (CAU)

3.1. GENERAL

The Command/Arithmetic Unit (CAU) is a multitask instruction stacking processor maintaining control over the arithmetic and instruction sequencing operations in the UNIVAC 1110 System. Each of the two CAU's in a 2 × 1 configuration is capable of interfacing with a maximum of four main storage units (262K 36-bit words). Independent instruction and operand paths are provided for each main storage unit; each CAU also is capable of interfacing with Multiple Access Interface (MAI) units for up to 1048K words of extended storage by means of dual data paths. The paths to extended storage have overlapping capability which permits faster effective cycle time.

The CAU generates interrupt signals on any one of eight interprocessor interrupt lines and receives interprocessor interrupt signals from a maximum of eight sources. The CAU is capable of controlling one or two IOAU's.and is divided into five interacting components which are described in 3.1.1 through 3.1.5. Figure 3–1 illustrates the configuration of the Command/Arithmetic Unit.

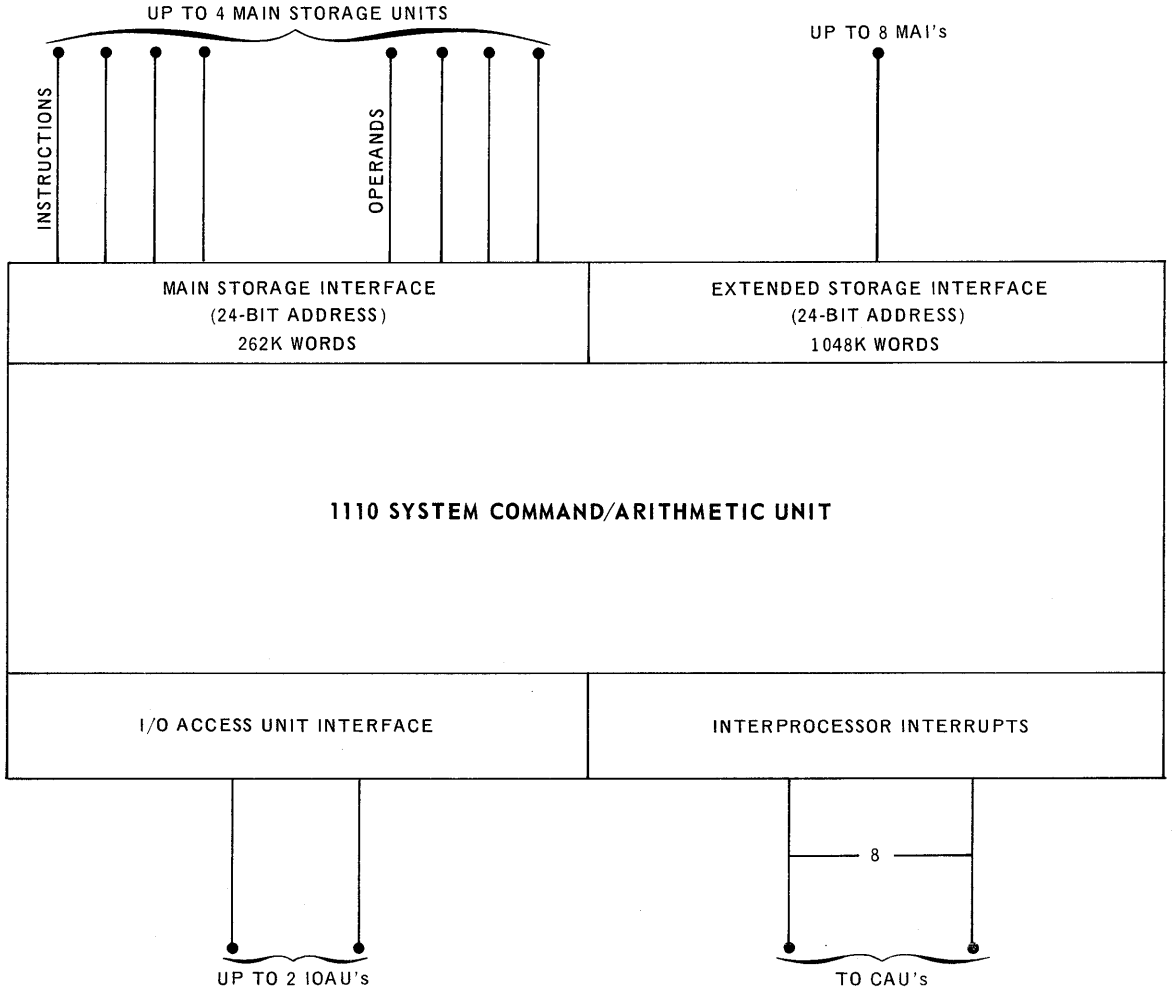


Figure 3–1. Command/Arithmetic Unit Configuration

3.1.1. Address Formation Section

The relative and absolute operand and absolute instruction addresses are formed in this section. During the operand address generation cycle, storage conflicts are checked, the base address is formed, program limits are checked, and index incrementation is performed.

The 18-bit relative operand address is formed by adding the contents of the modifier portion of the specified index register to the u field of the instruction. The 24-bit absolute address is formed by adding the contents of the index register specified in the instruction, the u field of the instruction, and the contents of a 15-bit base (instruction base or data base) from the processor state register.

The instruction address section accepts a 24-bit absolute address from the address formation section. The proper request is then formed to accompany the address to the proper storage module. The address is increased by one, and upon receipt of the proper acknowledge, the next request is sent to storage. Instruction referencing continues for consecutive addresses until a jump instruction is executed or an interrupt distrupts the sequence. In the case of a jump, a new absolute address is received from the operand addressing section, and the instruction referencing sequence is reinitiated.

3.1.2. General Register Stack (GRS)

The general register stack (GRS) consists of 112 integrated circuit control registers, program-addressable, each with a capacity of 36 bits. The GRS includes indexing registers, accumulators, repeat counters, a mask register, a real time clock register, and temporary storage locations for the processor state register. The stack is divided into an odd/even structure to allow simultaneous referencing when double-precision 72-bit word instructions are used.

The executive, through modes established in the processor state register, has exclusive use of a duplicate set of control registers.

Figure 3–2 illustrates the assignments for the general register stack.

3.1.3. Conditional Jump Section

The conditional jump section minimizes execution time through the CAU when a conditional jump instruction is encountered. This section, divorced from the main arithmetic section, tests whether the jump conditions are satisfied before control passes to the arithmetic section.

3.1.4. Store Operation Section

The store operation section, a subdivision of the control section of the CAU, handles all writing into storage. This section performs operations necessary for full-word and partial-word transfers into storage.

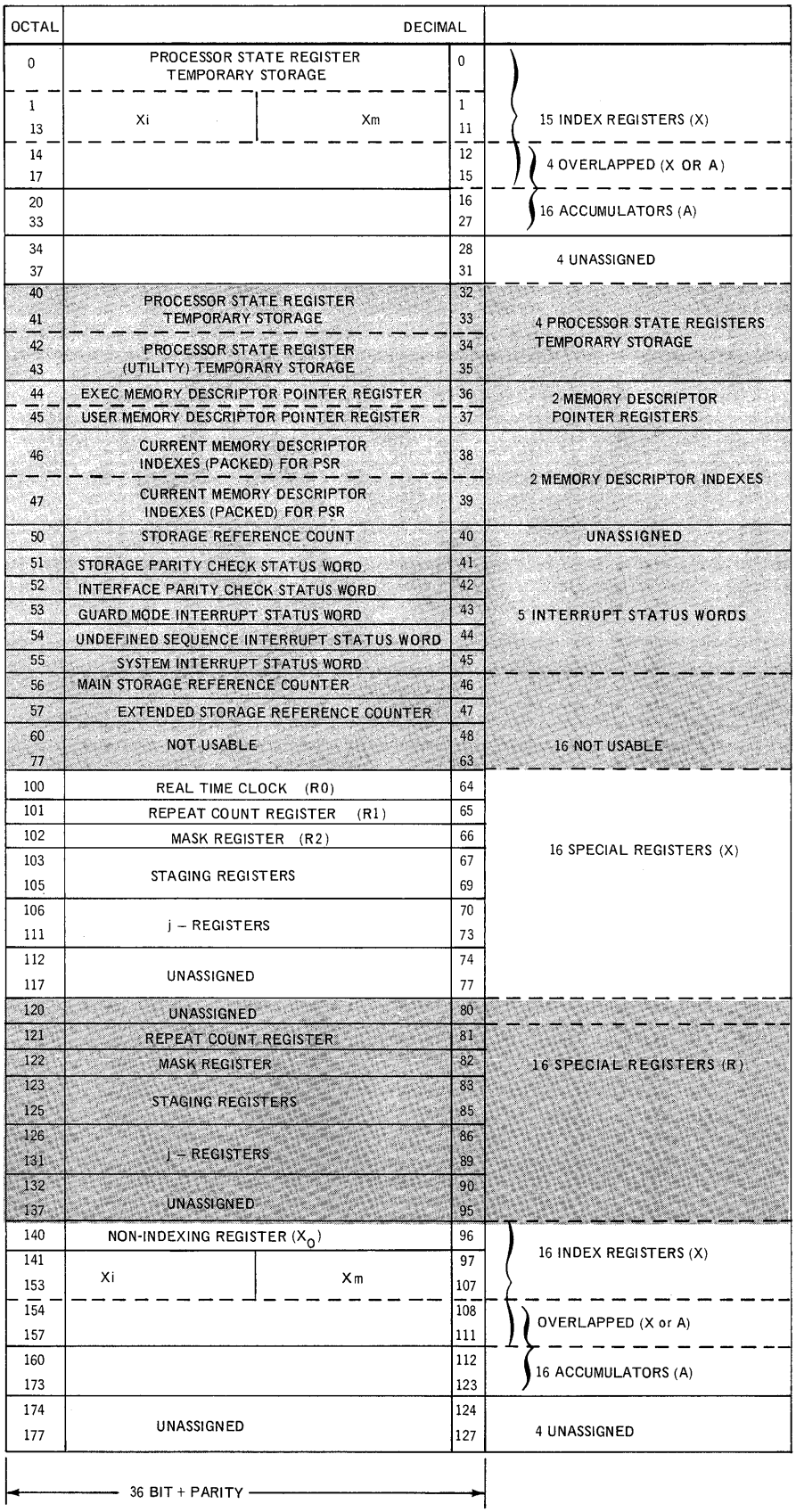


Figure 3–2. General Register Stack (GRS)

3.1.5. Arithmetic Section

In the UNIVAC 1110 System, the manipulation of data (addition, subtraction, multiplication, division, shifting) takes place in the arithmetic section of the CAU. During the execution of an arithmetic instruction, nonaddressable transient storage registers within the arithmetic section are used for the actual computation. The arithmetic section has the following capabilities:

- For fixed-point single-precision instructions, the j designator selects all of a word or a portion (half, third, quarter, or sixth) of a word for use in the arithmetic operation.
- By using special split-word arithmetic instructions, simultaneous addition or subtraction of corresponding half- or third-words is done.
- Through the use of a shift matrix, multiposition shifts require the same time as one bit position shifts. Right and left shifts of single- or double-length operands can be specified. Left shifting may be circular or logical. Right shifts may be circular, logical, or algebraic. When the results of an arithmetic operation are in double-word 72-bit form, they are automatically stored in consecutive control registers and may be retrieved in 72-bit length results or operands.
- Alphanumeric comparisons utilizing the mask register (R2) allow any selection of bits in one 36-bit word to be directly compared with the corresponding bits of another word or series of words.
- 1108 compatibility.

3.1.5.1. The Adder

The adder in the CAU is a ones complement subtractive adder for 36-bit and 72-bit operations. For purposes of analysis and debugging, the programmer may manually simulate the computer operation by simple binary or octal addition.

Two special internal designators associated with the arithmetic adder are the overflow designator and the carry designator. The fixed-point addition and subtraction instructions, single and double precision, and the Load Processor State Register instruction are the only instructions which affect these two designators.

3.1.5.2. Arithmetic Accumulators

The 16 arithmetic accumulators can be addressed directly by the programmer and are available for storage operands and results of arithmetic computations. These arithmetic accumulators should not be confused with the nonaddressable transient registers that are used in actual computation and are contained within the arithmetic section itself.

With the Add To X and Add Negative To X instructions, the index registers also act as accumulators in the same manner as the arithmetic registers.

3.1.5.3. Partial-Word Transfers

To minimize shifting and masking and to allow computation based on selected portions of words, the UNIVAC 1110 System permits the transfer of partial words into and out of the arithmetic section in a varying pattern (see Figure 3–3).

By selecting the coding of the j designator in the instruction word and bit 17 of the processor state register, a programmer may transfer a chosen portion of an operand to or from a control register or the arithmetic section. The transfer to an arithmetic register may also be accompanied by sign extension for subsequent arithmetic operations, depending on the j designator.

J	PSR BIT 17	BIT POSITIONS OF (U)→ A, X, or R	BIT POSITIONS OF (A), (X), or (R)→ U
00	–	35–00 → 35–00	35–00 → 35–00
01	–	17–00 → 17–00	17–00 → 17–00
02	–	35–18 → 17–00	17–00 → 35–18
03	–	17–00 → S 17–00	17–00 → 17–00
04	0	35–18 → S 17–00	17–00 → 35–18
	1	26–18 → 08–00	08–00 → 16–18
05	0	11–00 → S 11–00	11–00 → 11–00
	1	08–00 → 08–00	08–00 08–00
06	0	23–12 → S 11–00	11–00 → 23–12
	1	17–09 → 08–00	08–00 → 17–09
07	0	35–24 → S 11–00	11–00 → 35–24
	1	35–27 → 08–00	08–00 → 35–27
10	–	05–00 → 05–00	05–00 → 05–00
11	–	11–06 → 05–00	05–00 → 11–06
12	–	17–12 → 05–00	05–00 → 17–12
13	–	23–18 → 05–00	05–00 → 23–18
14	–	29–24 → 05–00	05–00 → 29–24
15	–	35–30 → 05–00	05–00 → 35–30
16	–	18 bits* → 17–00	NO TRANSFER
17	–	18 bits* → S 17–00	NO TRANSFER

* If x = 0, h, i, and u are transferred
If x ≠ 0, u + $(X_x)_m$ is transferred

S = Sign extension, where the sign is the leftmost bit of the partial word selected by j.

Figure 3–3. j-Determined Partial-Word Operation

3.1.5.4. Split-Word Arithmetic

The system can perform addition and subtraction of half-words or third-words simultaneously. The right halves of two operands, for example, are added and the sum is stored in the right half of the selected accumulator. At the same time, the left halves of the save two operands are added and the result is stored in the left half of the save accumulator. There is no carry interaction between the halves. The same holds true for thirds of words. Each partial word operates as independent entity with its own end-around carry.

3.1.5.5. Shifting

The UNIVAC 1110 System can perform both single-length shifting (36-bits) or double-length shifting (72-bits), treating the latter as if operating with a single 72-bit register. A high-speed shift matrix makes execution time independent of the number of places involved in the shift, which means that an operand can be shifted from 0 to 72 positions in one storage cycle time.

Six types of shift operations are provided:

- Right Circular – bits shifted out at the right reappear at the left.
- Left Circular – bits shifted out at the left reappear at the right.
- Right Logical – zeros replace bits shifted out of the most significant positions. Bits shifted out of the least significant position are lost.
- Left Logical – zeros replace bits shifted out of the least significant positions. Bits shifted out of the most significant position are lost.
- Right Algebraic – sign bits replace bits shifted out of the most significant positions. Bits shifted out of the least significant position are lost.
- Scale-Factor Shift – a single or double accumulator left shift which positions the word and simultaneously counts the number of shifts required until the most significant bit of the accumulator is different than the next most significant bit.

3.1.5.6. Double-Precision Fixed-Point Arithmetic

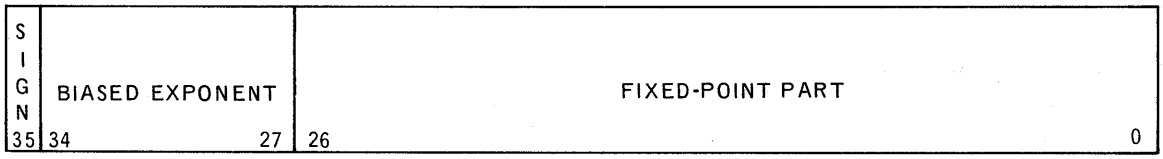
The UNIVAC 1110 System provides 72-bit, double-precision fixed-point addition and subtraction. Operands are processed as if they occupied a single 72-bit register. Bit 71, the high order bit, is the sign bit.

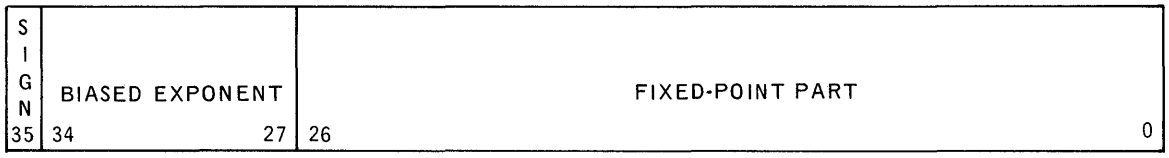
In addition, several arithmetic instructions produce two-word results. With fixed-point multiplication, a double-length product is stored in two arithmetic registers for integer and fractional operations. Ingeter and fractional division is performed upon a double-length dividend with the quotient retained in A and the remainder retained in A + 1.

3.1.5.7. Floating-Point Arithmetic

The UNIVAC 1110 System provides an extensive hardware repertoire of floating-point instructions. If the arithmetic is single precision, the range is from 10^{38} to 10^{-38} with eight-digit precision. The word formats are given below.

Operand Format:





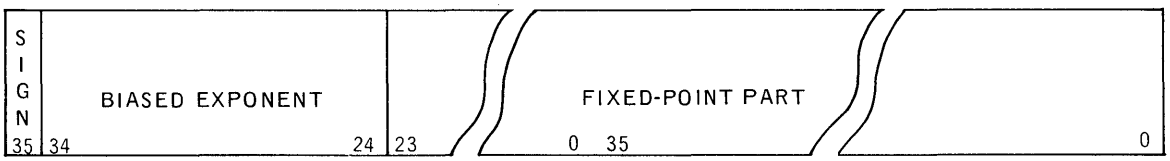
WORD 1

WORD 2

In a single-precision floating-point operation, word 1 is the more significant portion of the result. Word 2 contains the less significant portion. Mathematical error tracing can determine how much accuracy is being lost in calculations using this format. The two-word result of these single-precision operations is developed in two contiguous arithmetic registers.

Double-precision floating-point operands and results fall into the approximate range of 10^{307} to 10^{-308} with 18-digit precision. The values are expressed in two adjacent words as shown in the following format.

Source and Result Format:



Full double-precision operations do not require a repeated sign and exponent in the 36 least significant bits.

To express negative exponents, the hardware biases (floats) the exponent on a midvalue. The sign bit of the floating-point word applies to the fixed-point part. The true and biased ranges of the exponent are as follows:

	True	Biased
Single precision (8 bits)	-128_{10} to $+127_{10}$	$0 - 255_{10}$
Double precision (11 bits)	-1024_{10} to $+1023_{10}$	$0 - 2047_{10}$

The fixed-point part of a floating-point number is normally in the range from 1/2 to 1. Such a value places a 1 bit in the most significant bit position of the fixed-point part. When this condition exists, the floating-point number is said to be normalized. A negative floating-point number is represented by the one's complement of the corresponding positive fixed-point part.

Floating-point instructions are also provided for the following operations:

- Determining differences in exponents;
- Packing and unpacking exponents and fixed-point parts (single and double precision);
- Conversion – Single- to double-precision
  Double- to single-precision

3.2. INSTRUCTION STACKING

The CAU uses the instruction stacking technique to control up to four instructions at various stages of execution. Instruction execution generally involves a series of of five basic operations:

- Instruction acquisition sequence
- Address generation sequence
- Operand acquisition sequence
- Computational sequence
- Results storage sequence

As a result of the four-deep instruction stack and dual datapath arrangements, these sequences may be initiated such that the effective instruction execution time per non-extended sequence instruction is 300 nanoseconds.

Figure 3–4 shows a cross section in time of an instruction stream flowing through the CAU.

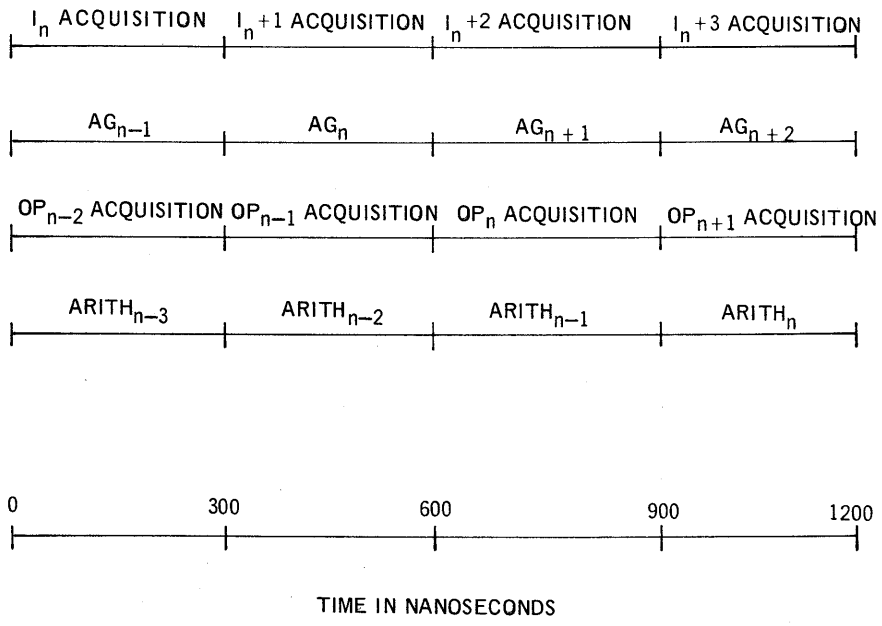
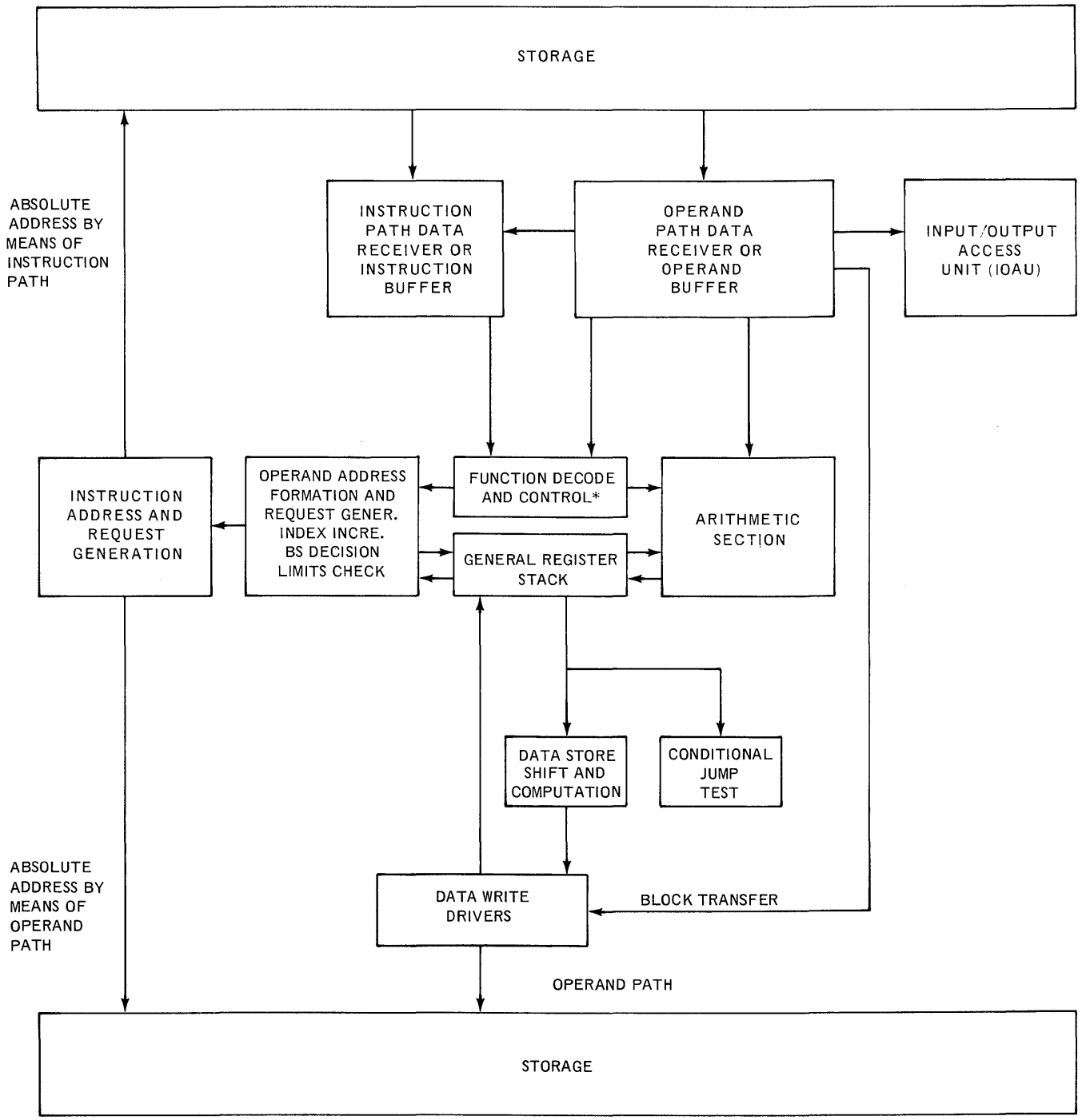


Figure 3–4. Instruction Execution (Approximate Timing) in a Command/Arithmetic Unit

3.3. OPERATIONS IN BRIEF

A block diagram of CAU operations is given in Figure 3–5. Instructions to be interpreted are routed to the function decode and control section from either the data path from storage, the instruction buffer, the operand data path, or the operand buffer. The route taken depends upon the initial reference and state of the processing section.



*Control may pass directly to the input/output access unit

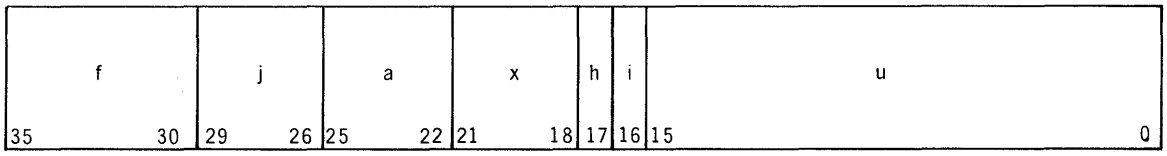
Figure 3–5. Instruction Path Through the Command/Arithmetic Unit

If reading data from storage is necessary for the execution of the instruction, an absolute operand address is formed and the proper data is requested. Upon receipt of the data, the arithmetic section performs the required operation; the results of the operation are temporarily stored.

The arithmetic section is bypassed if the instruction requires only writing data into storage. In this case, from the function and decode section, the instruction is passed to the general register stack (GRS). From the GRS, data is passed through the store section and written into storage.

3.4. INSTRUCTION WORD FORMAT

The format of the UNIVAC 1110 System instruction word is illustrated below followed by an explanation of each field. Some fields have more than one meaning, depending on the class of instruction.



3.4.1. Function Code – f Field

These six bits specify the operation to be performed. For function codes above 70_8, the f and j fields are combined to produce a 10-bit function code. An invalid function code generates an interrupt.

3.4.2. Partial-Word or Immediate-Operand Designator – j Field

For function codes less than 70_8, the j designator specifies partial-word or immediate-operand selection. (See Figure 3–3 for specific partial-word selections.)

3.4.3. Control Register Designator – a Field

The four-bit a field designates which control register, within a group selected by the function code, is involved in the operation. For most operations, the a field refers to an arithmetic register; for a few, it refers to either an index register or some other control register. In input/output instructions, it is used in specifying the the channel and its associated input or output access control register. For function code 70_8, the a and j designators together address one of the 112 control registers.

3.4.4. Index Register Designator – x Field

The x field specifies one of the 15 index registers to be used in address modification. When x = 0 is designated, indexing is suppressed.

3.4.5. Index Modification Designator – h Field

The h field controls modification of the index value (Xm) by the increment field (Xi) after indexing (see 3.5.1). If h = 1, the right half of the index register is normally modified by adding the contents of its left half; if h = 0, modification is suppressed.

3.4.6. Indirect Address Designator – i Field

The i designator controls the use of indirect addressing during instruction execution. If i = 0, the instruction functions without indirect addressing. If i = 1, the 22 least significant bit positions of the instruction (x, h, i and u fields) are normally replaced in the instruction register with the contents of the 22 least significant bit positions of the word at the effective address. Indirect addressing continues as long as i = 1 with full indexing capability at each level.

3.4.7. Address Field – u Field

The u field normally specifies the operand address. However, for certain instructions it may hold a constant. For example, the shift instructions use the seven least significant bit positions to produce the shift count. For all instructions, the value in the u field may be modified by the contents of an index register.

3.5. CONTROL REGISTERS

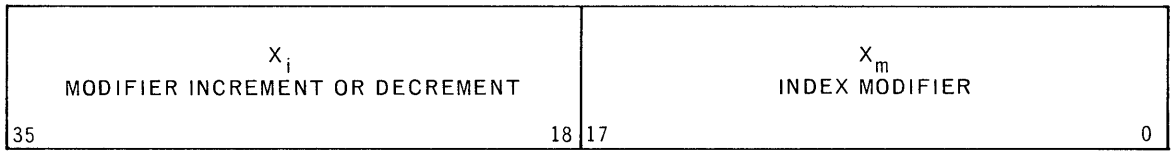
The 112 program-addressable control registers consist of 36-bit integrated-circuit registers, with a basic cycle time of 75 nanoseconds. Two parity bits are included with each control register.

Effective use of multiple accumulators and index registers for the development and use of constants, index values, and operands substantially improves performance.

In the following descriptions only programmable registers are discussed. The executive, through modes established by the contents of the processor state register, has exclusive use of the duplicate set of control registers indicated by the shaded areas in Figure 3–2, as well as the associated registers in the IOAU.

3.5.1. Index Registers

Control register locations 0_8 – 17_8 are index registers and normally have the following format:



The Xm portion of the index register is an 18-bit modifier to be added to the base operand address of the instruction. The Xi portion of the index word updates the Xm portion, *after* base operand address modification.

Index register modification is specified by a 1 bit in the h field of the instruction, while indexing itself is specified by a nonzero value in the x field. Both functions take place within the basic instruction execution cycle.

When cascaded indirect addressing is used in a programmed operation, full indexing capabilities are provided at each level. Indirect addressing replaces the x, h, i, and u portions of the instruction register, beginning with a new indexing cycle for each cascaded sequence. This process continues until the i field is zero.

3.5.2. Arithmetic Accumulators

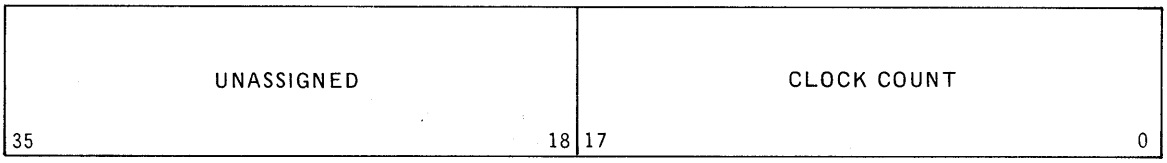
Control register locations 14_8 – 33_8 are arithmetic accumulators for programmed storage of arithmetic operands and results. The actual computation is performed in nonaddressable transient registers within the arithmetic section.

Depending upon the instruction, use of the accumulators results in a variety of word formats. Double-precision instructions and a number of logical instructions reference two contiguous accumulators, that is, A and A + 1. In arithmetic operations requiring two-word operands or producing two-word results, A + 1 always holds the least significant part of an operand or result. Some instructions, such as single-precision floating-point operations, call on a one-word operand from main storage but produce a two-word result in the specified A and A + 1.

3.5.3. R Registers

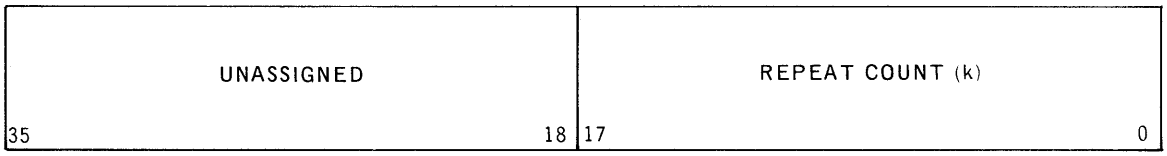
The sixteen control register locations 100_8 – 117_8 are R registers. The first three of these (R0, R1, R2) have specified functions and formats as described below. The remaining R registers are not specifically assigned; typically they are used as loop counters, transient registers, or storage for intermediate values or constants. R3, R4, and R5 are used as S registers in character instructions.

3.5.3.1. R0 – Real Time Clock



This register is initially loaded by the program. The contents are then decremented once each 200 microseconds. A real time clock interrupt occurs when the clock count is decremented through zero. Thus, if the clock is initially loaded with the value 5000, an interrupt occurs in exactly one second.

3.5.3.2. R1 – Repeat Counter



The repeat counter controls repeated operations such as block transfer and search instructions. To execute a repeated instruction k times, the repeat counter is loaded with k prior to the execution of the instruction.

3.5.3.3. R2 – Mask Register

The mask register functions as a filter in determining which portions of words are to be compared in repeated masked search operations. The mask register must be loaded prior to executing the search command. The contents of the effective address are compared to the contents of the arithmetic register only with respect to those positions which contain one's in the mask register.

3.6. INSTRUCTION REPERTOIRE

The UNIVAC 1110 System is provided with an unusually powerful and flexible instruction repertoire. Many of the instructions are effectively accessed and completed in the time of one storage cycle. Included in the complete set of instructions is a group which permits fast and simplified executive system control.

To a great extent, the instruction repertoire of the UNIVAC 1110 System is identical to that of the UNIVAC 1106/1108 System in order to maintain total compatibility. To utilize the greater capacity of the UNIVAC 1110 System, character manipulation instructions and additional privileged instructions have been included.

The following sections group the instructions by functional class to illustrate the scope of the UNIVAC 1110 instruction repertoire.

3.6.1. Load Instructions

Single-precision load instructions cause transfers from storage to the a field designated registers through the arithmetic unit according to the j designator value.

Double-precision load instructions use the j field as a minor function code designator.

The load instructions are:

Load A
Load Negative A
Load Magnitude A
Load Negative Magnitude A
Load R
Load X
Load X Modifier
Load X Increment
Double Load A
Double Load Negative A
Double Load Magnitude A
Load BI And Jump
Load BD And Jump
Load PSR Designators

3.6.2. Store Instructions

The contents of the A, R, and X registers specified by the a field are transferred to a storage location under j designator control. If U is less than 200_8, the complete data word is transferred to a control register in the general register stack. The Store Zero instruction stores zeros at location U. The Block Transfer instruction moves a block of information in storage. The number of words transferred is defined by the repeat count stored in the repeat count register.

The store instructions are:

Store A
Store Negative A
Store Magnitude A
Store R
Store Zero
Store X
Block Transfer
Double Store A

3.6.3. Fixed-Point Arithmetic Instructions

Add or Add Negative instructions add or subtract the contents of U from specified A or X registers. The transfer of data from location U to the arithmetic unit is under control of the j designator. For double-precision and parallel half-word and third-word arithmetic operations, the value in the j field is a minor function code.

The sign of the product or quotient of multiply and divide instructions is determined by algebraic rules; operands of like signs produce a positive result and operands of unlike signs produce a negative result.

The fixed-point arithmetic instructions are:

Add To A
Add Negative To A
Add Magnitude To A
Add Negative Magnitude To A
Add Upper
Add Negative Upper
Add To X
Add Negative To X
Multiply Integer
Multiply Single Integer
Multiply Fractional
Divide Integer
Divide Single Fractional
Divide Fractional
Double-Precision Fixed-Point Add
Double-Precision Fixed-Point Add Negative
Add Halves
Add Negative Halves
Add Thirds
Add Negative Thirds

3.6.4. Floating-Point Arithmetic Instructions

Floating-point arithmetic instructions cause the transfer of single- or double-precision operands from storage to the arithmetic unit. The j field is a minor function code designator. The single-precision floating-point operations of Add, Add Negative, Multiply, and Divide instructions produce results consisting of two single-precision words which are: sum and residue, difference and residue, most and least significant position of the double length product, and quotient and remainder, respectively.

When the mantissa produced for a double-precision floating-point Add, Add Negative, Multiply, Divide, or Load And Convert To Floating instruction, or an Expand And Load instruction is executed, and the mantissa of the double-precision result is +0 or −0, all 72 bits of the final result are cleared to zeros; a characteristic overflow or underflow interrupt never occurs.

If the floating-point Compress And Load instruction is executed and the double-precision input operand has a +0 or −0 mantissa, the final result is cleared to all zeros.

Input operands for floating-point Multiply and Divide, Floating Expand And Load, and Floating Compress And Load must be normalized.

The floating-point arithmetic instructions are:

Floating Add
Floating Add Negative
Floating Multiply
Floating Divide
Load And Unpack Floating
Load And Convert To Floating
Magnitude Of Characteristic Difference To Upper
Characteristic Difference To Upper
Double-Precision Floating Add
Double-Precision Floating Add Negative
Double-Precision Floating Multiply
Double-Precision Floating Divide
Double Load And Unpack Floating
Double Load And Convert To Floating
Floating Expand And Load
Floating Compress And Load

3.6.5. Repeated Search Instructions

The repeated search instructions require that a repeat count be loaded into a register designated for this purpose. If a search instruction is initiated when the repeat value is zero, the search instruction is effectively skipped. Otherwise, the contents of the a field specified A register are transferred to the arithmetic unit, and the operation specified by the instruction is initiated; this involves decrementing of the repeat count transferring an operand to the arithmetic section and performing the specified comparisons, and index register decrementation.

If the condition being sought is not fulfilled and the repeat count is not zero, the instruction is executed again. The instruction is repeated until the condition specified by the instruction is fulfilled or until the repeat count is decreased to zero. If the condition is fulfilled, the operation is terminated, the next instruction is skipped, and the following instruction is performed; otherwise, the next instruction is taken.

For masked search instructions, the repeat count is loaded and a mask value is preloaded into the register specified for that purpose. A logical AND is performed with individual mask bits and the register contents designated by the instruction.

The repeated search and masked search instructions are:

Search Equal
Search Not Equal
Search Less Than Or Equal
Search Greater
Search Within Range
Search Not Within Range
Mask Search Equal
Mask Search Not Equal
Mask Search Less Than Or Equal
Mask Search Greater
Masked Search Within Range
Masked Search Not Within Range
Masked Alphanumeric Search Less Than Or Equal
Masked Alphanumeric Search Greater

3.6.6. Test (or Skip) Instructions

These instructions cause data to be transferred to the arithmetic section and tested as prescribed by the instruction. If the condition specified is fulfilled, a skip is performed; otherwise the next sequential instruction is executed.

The test instructions are:

Test Even Parity
Test Odd Parity
Test Less Than Or Equal To Modifier
Test Zero
Test Nonzero
Test Equal
Test Not Equal
Test Less Or Equal
Test Greater
Test Within Range
Test Not Within Range
Test Positive
Test Negative
Double-Precision Test Equal

3.6.7. Shift Instructions

When performing shift instructions, the contents of the specified A register are moved left or right a specified number of bit positions. The shifted values are stored back into the same A registers.

There are three basic types of shift instructions: circular, logical, and algebraic. The shift count is specified by bit positions 6 through 0 of U. If the shift count exceeds 72, the results are not defined.

In the shift and count instructions, the shift count is not controlled by the instruction word. Instead, the contents of a single-word operand (U) or a double-word operand (U, U + 1) are transferred to the arithmetic unit where the register contents are scaled; the contents are shifted left circularly until the leftmost two bits are unequal. The scaled data is stored in the a field designated A registers specified by the instruction and the count (of the number of shifts necessary to scale the data) is stored in the adjoining A register.

The shift instructions are:

Single Shift Circular
Double Shift Circular
Single Shift Logical
Double Shift Logical
Single Shift Algebraic
Double Shift Algebraic
Load Shift And Count
Double Load Shift And Count
Left Single Shift Circular
Left Double Shift Circular
Left Single Shift Logical
Left Double Shift Logical

3.6.8. Executive System Control Instructions

This group of instructions allows executive system control of programs operating in a multiprocessing or multiprogramming environment. These instructions are used for establishing the contents of the processor state register, storage limits boundaries, interrupt locations, initiation and control of I/O activity, interprocessor communication, and task assignment. If these instructions are executed while in guard mode, a guard mode interrupt is generated.

The executive system control instructions are:

Prevent All I/O Interrupts And Jump
Store Channel Number
Load Processor State Register
Initiate Interprocessor Interrupt
Load/Store Storage Limits Register
Set MSR 1108 (Select Interrupt Location)
Load Channel Select Register
Load Last Address Register
Enable/Disable Day Clock
Load Processor Interrupt Pointer
Load Breakpoint Register
Store Jump Stack

3.6.9. Jump Instructions

Each of the conditional jump instructions performs a specific test. If the condition specified by the instruction exists, the instruction stored at address U is executed next; otherwise, the next sequential instruction is taken.

The Jump Greater And Decrement instruction forms a general register stack address from the combined ja fields; the contents of this address is tested to determine if the jump will be made.

The conditional and unconditional jump instructions are:

Jump Greater And Decrement
Double-Precision Zero Jump
Store Location And Jump
Jump Positive And Shift
Jump Negative And Shift
Jump Zero
Jump Nonzero
Jump Positive
Jump Negative
Jump Keys
Halt Keys And Jump
Allow Interrupts And Jump

Jump No Low Bit
Jump Low Bit
Jump Modifier Greater And Increment
Load Modifier And Jump
Jump Overflow
Jump No Overflow
Jump Carry
Jump No Carry
Load I Bank Base and Jump
Load D Bank Base and Jump

3.6.10. Logical Instructions

The logical instructions perform logical AND, OR, and XOR operations. One of the operands is obtained from storage location U and the other from A. The logical result is placed in A + 1.

The logical instructions are:

Logical OR

Logical XOR

Logical AND

Masked Load Upper

3.6.11. Miscellaneous Instructions

The Execute instruction allows the execution of an instruction at a remote address U without performing a jump to that address. The program address count is increased, and the instruction after the Execute instruction is performed next.

The Executive Return instruction allows a user to interrupt and switch control to the executive system.

The Test And Set instruction is used to test the entrance to areas of code that are not reentrant so as to be able to avoid attempting to share them simultaneously between CAU's. The UNIVAC 1110 System provides a skip in addition to an interrupt, using the a field as minor function code.

The miscellaneous instructions are:

Execute
Executive Return
No Operation (NO-OP)
Test And Set
Load/Store PSR Designators

3.6.12. I/O Instructions

This group of instructions allows the program (usually the executive system) to initiate, test, and control the IOAU and its operations. The UNIVAC 1110 System has a maximum channel capacity of 48 channels for two IOAU's. If these instructions are executed while in the guard mode, a guard mode interrupt is generated.

The I/O instructions are:

Load Input Channel
Load Input Channel And Monitor
Jump Input Channel Busy
Disconnect Input Channel
Load Output Channel
Load Output Channel And Monitor
Jump Output Channel Busy
Disconnect Output Channel
Load Function Channel
Load Function Channel And Monitor
Jump Function In Channel
Allow All Channel External Interrupts
Prevent All Channel External Interrupts
Load Input Access Word
Load Output Access Word
Load Input Chain Word
Load Output Chain Word
Store Input Access Word
Store Output Access Word
Store Input Chain Word
Store Output Chain Word
Load Pointer Base Register
Load Processor Interrupt Pointer Register
Allow Channel Interrupts
Prevent Channel Interrupts

3.6.13. Invalid and Not-Used Codes

Invalid operation codes generate an interrupt to location 241_8 when executed. The not-used codes give unpredictable results when used and are reserved for future instruction repertoire expansion as are the invalid operation codes.

3.6.14. Character Instructions

The UNIVAC 1110 System provides a comprehensive subset of character-oriented data processing instructions. The instructions operate on strings of characters under the control of a set of three staging registers. The generation of addresses for the individual character strings utilizes both J registers and X registers. The J registers are implicitly addressed by the instructions and are used when indexing through the character strings. The x field of the instruction specifies the first index register; additional index registers are X + 1 and X + 2 where X must be less than or equal to 15_8. The X registers are used to locate the individual character strings.

3.6.14.1. Byte Instructions

The byte instructions are:

Byte Move
Byte Move With Translate
Byte Translate And Test
Byte Translate And Compare
Byte Compare
Byte To Packed Decimal Convert
Packed Decimal To Byte Convert
Edit

3.6.14.2. Byte/Binary Conversion Instructions

The byte/binary conversion instructions are:

Byte To Binary Single Integer Convert
Byte To Binary Double Integer Convert
Binary Single Integer To Byte Convert
Binary Double Integer To Byte Convert
Byte To Single Floating Convert
Byte To Double Floating Convert
Single Floating To Byte Convert
Double Floating To Byte Convert
Quarter-Word Byte To Binary Compress
Binary To Quarter-Word Byte Extend
Quarter-Word Byte To Binary Halves Compress
Binary Halves To Quarter-Word Byte Extend
Quarter-Word Byte To Double Binary Compress
Double Binary To Quarter-Word Byte Extend

3.6.14.3. Decimal Arithmetic Instructions

The decimal arithmetic instructions are:

Byte Add

Byte Subtract

3.7. STORAGE REFERENCE COUNTERS

The CAU includes two storage reference counters, one each for main and extended storage. Each reference to storage made by a CAU will cause the appropriate storage reference counter to be incremented. These counters are for use by the executive system in storage and activity management. The storage reference counters are contained in the GRS (see Figure 3–2).

4. EXECUTIVE SYSTEM CONTROL FEATURES

4.1. GENERAL

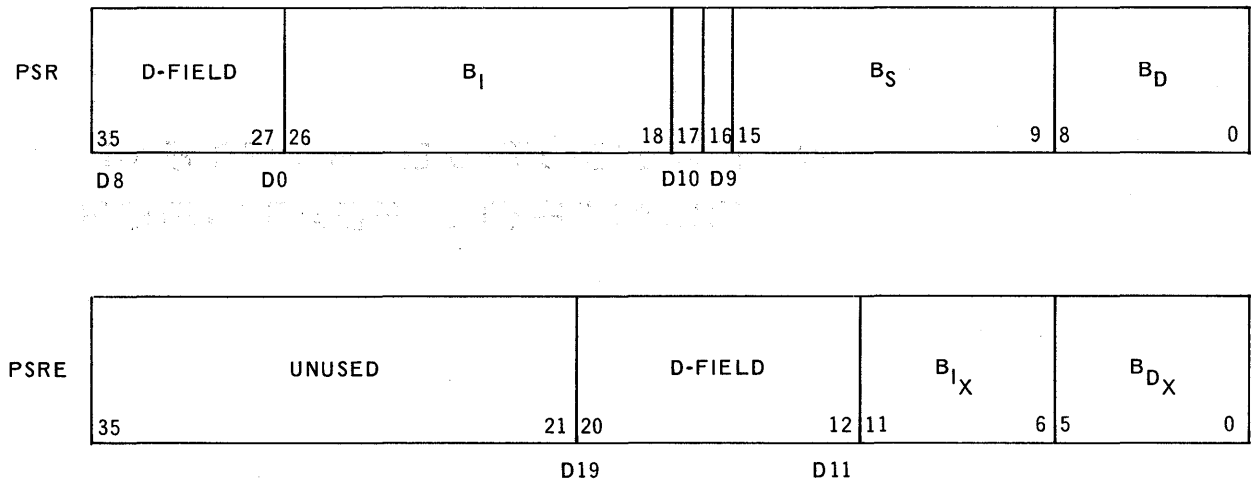
The executive system, by use of special hardware features, maintains complete control over the multiprogramming and multiprocessing environment of the UNIVAC 1110 System.

The multiprogramming and multiprocessing capabilities of the UNIVAC 1110 System are based on guard mode operation. When operating in this mode, certain instructions, registers, and storage locations are available for the exclusive use of the executive. Under the guard mode, unrelated programs are protected from interacting or interfering with one another.

4.2. PROCESSOR STATE REGISTER

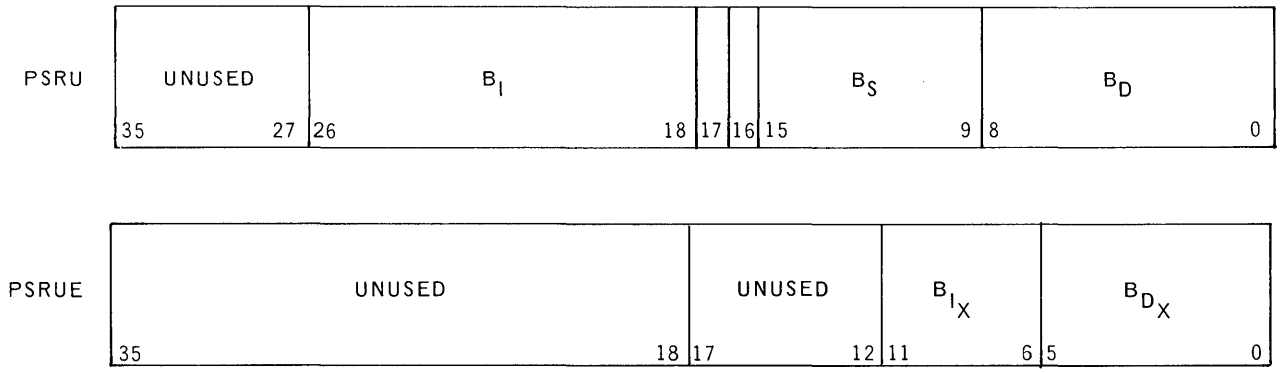
The processor state register (PSR) is used for locating programs in any available area in storage and for establishing various operational modes that are provided. The PSR is primarily controlled by the executive for itself and for user programs. The user is permitted to modify certain portions of the PSR by a memory descriptor table (MDT) prepared by the executive for that program. The PSR also records the status of arithmetic overflow and carry designators when an interrupt occurs. During an interrupt sequence the contents of the PSR are stored in the GRS and program control is transferred to the executive system.

The format of the processor state register is described below:



PSR – first word of main PSR
PSRE – second word (extension portion) of main PSR
D-Field – location of control bits D0 through D8
B_I – I bank base
B_S – program effective switch point
B_D – D bank
D0 – carry designator
D1 – overflow designator
D2 – guard mode and storage protection
D3 – write only storage protection
D4 – character addressing mode
D5 – double-precision underflow
D6 – control register (GRS) selection
D7 – base register suppression
D8 – floating point zero
D9 – index register mode selector (24 bit if set and D7 and the i-bit of instruction are also set)
D10 – quarter-word mode
D11 – operand base selector (utility base if set and the i-bit of instruction is set – the jump operand is excluded)
D12 – PSR SLR Selector (PSRU and SLRU if set)
D13 – PSR I bank write selector (write illegal if set)
D14 – PSR D bank write selector (write illegal if set)
D15 – PSRU I bank write selector (write illegal if set)
D16 – PSRU D bank write selector (write illegal if set)
D17 – set D9 on interrupt
D18 – PSR and Storage Limits Register Auto-Switch (when set allows switching from the currently active PSR and SLR, as specified by D12, to the inactive PSR and SLR if the storage limits test fails. For other than a jump, D12 remains in the same state).
D19 – EXEC MDP allow
B_{I_X} – 6-bit extension value for B_I
B_{D_X} – 6-bit extension value for B_D

The Processor State Register is composed of two words called PSR and PSRE. A second processor state register, called the utility processor state register has a similar format, except that the two words are called PSRU and PSRUE. The format of the utility PSR shown below is identical to the main PSR except that bits 35–27, 17, and 16 of PSRU and bits 20–12 of PSRUE are not used.

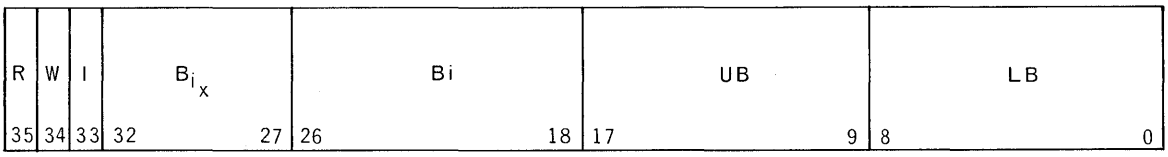


PSRU – first word of utility PSR
PSRUE – second word (extension portion) of the utility PSR
B_I – same as the PSR
B_S – same as the PSR
B_D – same as the PSR
B_{I_X} – same as the PSRE
B_{D_X} – same as the PSRE

Unused fields of the PSRU (bits 35–27, 17, 16) and the PSRUE (bits 35–18, 17–12) must be zeros.

4.2.1. Memory Descriptor Table

The user can reload part of his processor state register from the memory descriptor table (MDT). An MDT is prepared for the user by the executive and consists of a sequence of one or more 36-bit memory descriptor words. The format of a memory descriptor word (MDW) is described below:



LB = Lower boundary of program area (I bank or D bank)
UB = Upper boundary of program area (I bank or D bank)

When executing an LIJ instruction, the least significant 7 bits of UB are transferred to the BS field of the currently active PSR.

$B_i = B_I$ or B_D

$B_{i_x} = B_I$ Extension or B_D Extension (B_{I_X} or B_{D_X})

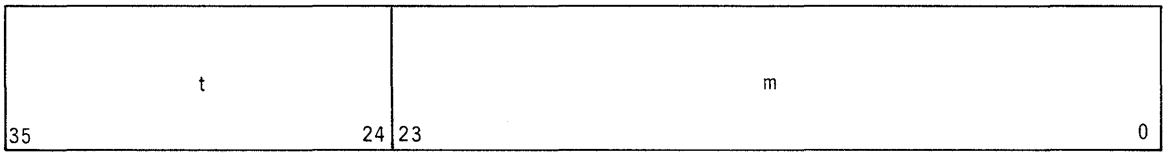
R = Residency bit (interrupt if set). In this case, the MDW may contain information needed to identify nonresident code so it can be transferred to storage before returning control to the user.

W = Write protect. Causes the appropriate I bank or D bank no-write bit to be set in the PSR. The W bit is ignored if either I or R bit is set.

I = Indirect index to EXEC MDT. Bits 29–18 of the MDW are used as the pointer into the EXEC MDT; the address of the next MDW is formed by adding the contents of the m field of the EXEC MDP and with bits 29–18 of the MDW. The I bit is ignored if the R bit is set.

4.2.2. Memory Descriptor Pointer (MDP)

The location and length of a memory descriptor table is defined by the corresponding memory descriptor pointer registers. There are two MDP registers, each having the same format: the executive MDP at general register stack address 44_8 and the user MDP at general register stack address 45_8. The format is presented below:



m – 24-bit pointer to the table (must address storage)

t – table length, 12 bits (4096 = maximum number of entries)

4.3. INTERRUPTS

The interrupt network of the UNIVAC 1110 System is extensive. It is the means of effecting real time, multiprogramming, and time-sharing operations on the system. The interrupt is a control signal generated by either a peripheral subsystem (external interrupt), an IOAU, or the control section of a CAU. Specific interrupt locations are assigned within the lower addresses of a main or extended storage module as specified by a 9-bit module select register (MSR). These interrupt locations are programmed to capture the interrupted address and enter interrupt response subroutines in the executive system. The synchronization of input/output activities and response to real time situations are accomplished through some of these interrupts.

Other interrupts are provided for certain error conditions detected within a CAU or IOAU. These may result from a programming fault such as an illegal instruction, a storage parity error, or a user program violation such as an attempt to write into a protected area of storage or a violation of guard mode. These fault interrupts are used by the executive to initiate remedial or terminating action when they are encountered.

4.4. GUARD MODE

Guard mode operation prevents user programs from executing any of the instructions listed below. These are reserved for the executive. It also permits protection of certain specified locations in main storage and control registers reserved for executive operations.

Guard mode is established by the Load Processor State Register instruction. Execution of this instruction with the appropriate PSR bit pattern is the only way that guard mode can be made operative and provides the only direct access to the PSR. Under guard mode, an attempt to perform any of the privileged instructions or functions listed below results in a Guard Mode fault interrupt:

- Storage limits violation when PSR bits D2 and/or D3 are set
- All I/O instructions
- Disabling of I/O interrupts for more than 100 microseconds
- Attempting to write into any of the executive control registers (40_8 – 100_8 or 120_8 – 177_8) when PSR bit D2 is set.

Guard mode is disabled by the occurrence of any interrupt. This stores the contents of PSR and PSRU in GRS locations 40_8 through 43_8, clears certain bit positions of the PSR, sets D7 and D6=1, and establishes executive mode operation.

5. INPUT/OUTPUT ACCESS UNIT

5.1. GENERAL

The Input/Output Access Unit (IOAU) has exclusive control over all input/output operations. An IOAU (see Figure 5–1) receives commands from the control section of either of the two Command/Arithmetic Units (CAU) to which it is connected. Information pertaining to the channel selected is routed back to the CAU that requested the information and interrupts are routed to either one of the two CAU's.

The method selected is controlled by the processor interrupt pointer register (see 5.1.1).

The basic IOAU section has eight channels and may be expanded to 24 channels. Any channel can be operated in either the internally specified index (ISI) or externally specified index (ESI) mode. When operating in either the ISI or ESI mode, data chaining is provided. Base relocation registers are provided for converting the relative address in an ACW to an absolute address. External interrupt and monitor interrupt tabling in the ESI mode are provided with a table pointer for each channel.

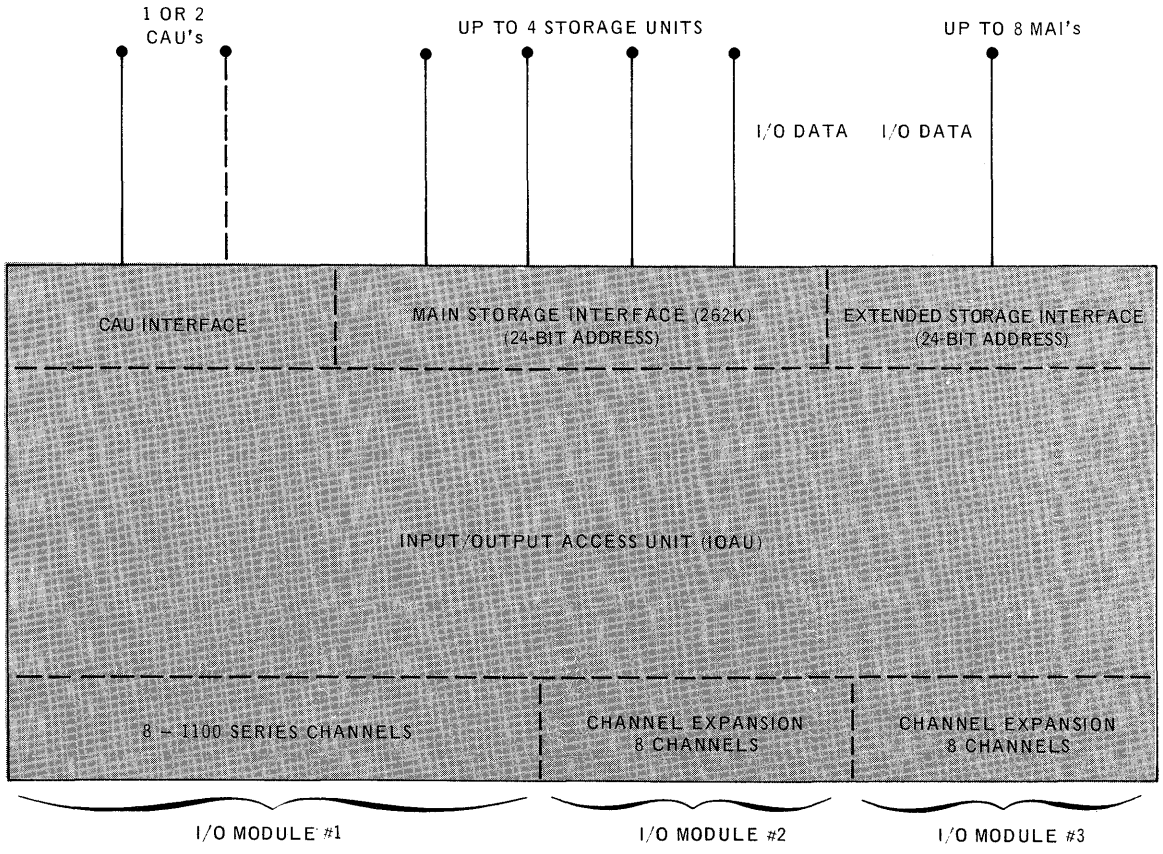


Figure 5–1. Input/Output Access Unit

5.1.1. Processor Interrupt Pointer Register

The processor interrupt pointer register is a two-bit register which controls the method by which interrupts are directed to a CAU. A channel interface is provided allowing console selection of the CAU to which operator initiated interrupts are routed. The contents of the processor interrupt select register are:

00_2 – Interrupts are directed to CAU 0

01_2 – Interrupts are directed to CAU 1

10_2 – Interrupts are connected between CAU 0 and CAU 1

11_2 – Interrupt to the CAU that initiated the operation on the I/O channel involved

5.1.2. Storage Interface

The IOAU has one access path to main storage and one access path to extended storage. All channels have ESI, ISI, and interrupt tabling capabilities. Through use of a maintenance panel switch, channels may be individually selected to operate in either the ESI or ISI mode. Each IOAU interfaces with all of main storage and extended storage. The paths to the multiple access interface (MAI) have overlapping capabilities allowing faster cycle time. The IOAU selection sequence of the multiple access interface units may be altered to allow extended storage interleaving. The sequence selection is designed to permit maintenance of individual storage units without any effect on the rest of the system.

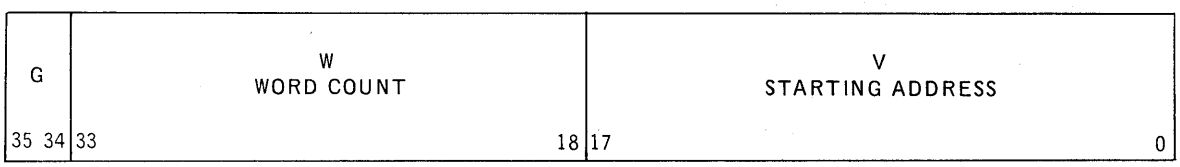
The IOAU has the ability to logically connect an input and an output channel back to back (physical connection also required) under executive control allowing storage transfers between the extended storage subsystem and main storage by means of IOAU storage interface paths.

5.2. INTERNALLY SPECIFIED INDEX MODE

Each I/O channel operates in one of three states: input, output, and function. Input and output are the data transmission states. The function state is actually an output state during which the processor sends one or more function words to the subsystem. Each function word specifies an operation to be performed by the subsystem.

The actual word-by-word transmission (regardless of transfer state) is governed by an access control word stored in an access control register. Two of these registers, one for input and one for output, are assigned to each I/O channel.

The format of the ISI access control word is as follows:



V 18 bits, the relative starting address for data transfer.

W 16 bits, the number of words still to be transferred. It decreases by 1 each time a word is transferred.

G 2 bits, the incrementation control for V.

= 00, increment V by 1 after each word is transferred.

= 10, decrement V by 1 after each word is transferred.

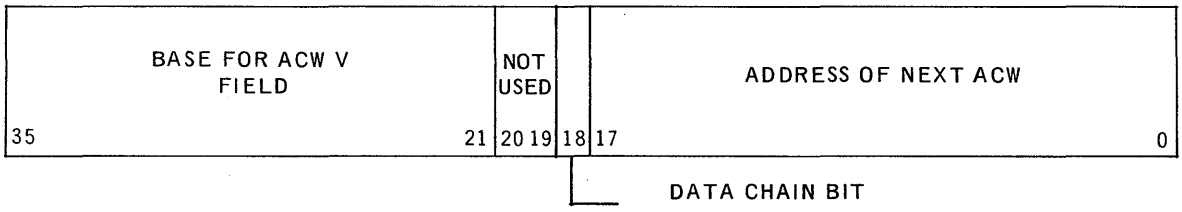
= 01 or 11, do not change V.

In initiating an input/output operation, the CAU sends an access control word to the IOAU, which is then loaded in the IOAU control register associated with a given channel. Depending on the contents of G, the I/O control section transfers subsequent words to or from successive locations in storage (increasing or decreasing addresses) or to or from a single location. After each transfer the word count, W, is decreased by one and tested for zero. A nonzero calls for transfer of the next word in the block; a zero terminates the transfer; and if the instruction calls for monitoring, the input/output monitor interrupt is set.

5.2.1. ISI Data Chaining

Data chaining is the linking of a series of access control words to provide the IOAU with the capabilities for scatter/read, gather/write operations. The access control words (ACW) are sequentially replaced whenever the ISI input/output ACW count field (W) is decreased from one to zero.

Whenever an ISI or output word count field (W) is decreased from one to zero, a chain sequence to replace it and the associated data chain pointer word are initiated if bit 18 of the current chain pointer (CP) word is equal to one. The new ACW and CP word is then read from two consecutive storage addresses specified by the address field of the current CP word plus chain base register (CBR). Data chaining continues until a CP word with bit 18 equal to zero is detected. Chaining will not occur if the word count field of the ACW equals zero initially.



Bits 35–21 Base value that is added to V field of ACW to form absolute address for data or function word transfer.

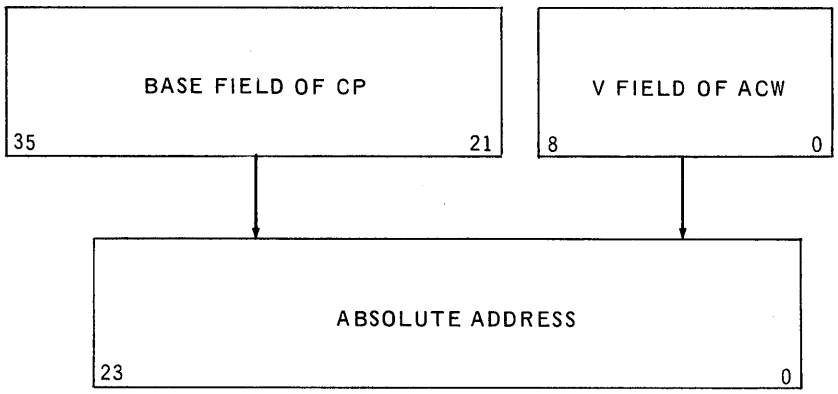
Bits 20–19 Zeros

Bit 18 Bit 18 equal to one: specifies data chaining for all ESI and ISI input or output. Bit 20 equal to zero: specifies no chaining. Bit 20 must be zero for a function transfer.

Bits 17–0 The relative address of the next ACW for chaining. A one is added to the address for next chain pointer word. The absolute address of the next ACW is formed by adding the contents of the chain base register (CBR) to the relative address (bits 17–0) to form a 24-bit absolute address.

5.2.2. ISI Absolute Address Generation

The absolute address for an ISI data or function transfer is formed by adding the address of the V field in the access control word and the base field of the chain pointer. The base field is not used for decrementing the address (V) field of the access control word.



5.2.3. ISI External Interrupts

When an IOAU receives an external interrupt signal from a subsystem, it responds as follows:

- Through the operation of the I/O priority network in the appropriate CAU, its instruction sequence is altered and the address of the next instruction is obtained from the fixed-address location.
- The status word on the input word/status word lines is transferred to a specifically reserved location in main storage as determined by the memory select register, the CAU number, and the channel group number.
- The input acknowledge signal is transmitted to the subsystem. The subsystem turns off the external interrupt and status word signals.

5.2.4. ISI Monitor Interrupts

A monitor interrupt is initiated if the transfer was initiated by the execution of a Load Input Channel and Monitor (LICM), Load Output Channel and Monitor (LOCM), or Load Function Channel and Monitor (LFCM) instruction and the word count field is zero and the chain bit of the chain pointer word is zero.

When a monitor interrupt occurs, the sequence of instructions being executed is altered. The next instruction to be executed is obtained from a fixed-address location in main storage.

5.2.5. Back-to-Back Mode

A special ISI channel back-to-back data transfer mode is provided. This special mode allows simultaneous execution of back-to-back storage transfers between areas of main and extended storage concurrently with normal I/O operation. Only one channel within the IOAU operates in this mode at any given time. The channel input and output must be connected with a back-to-back cable.

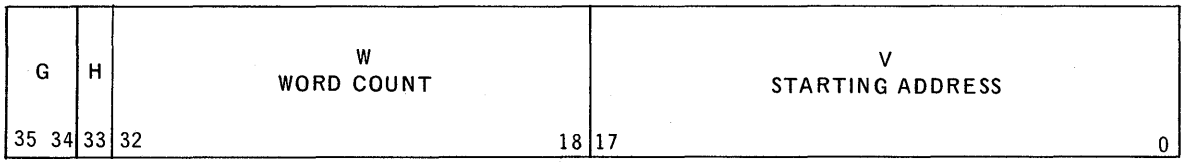
5.3. EXTERNALLY SPECIFIED INDEX MODE

The externally specified index (ESI) mode, in conjunction with data communications equipment, allows multiplexed remote communication devices to communicate with main storage over a single I/O channel on a self-controlled basis without disturbing the main program. Each such remote device communicates with its own area of main storage.

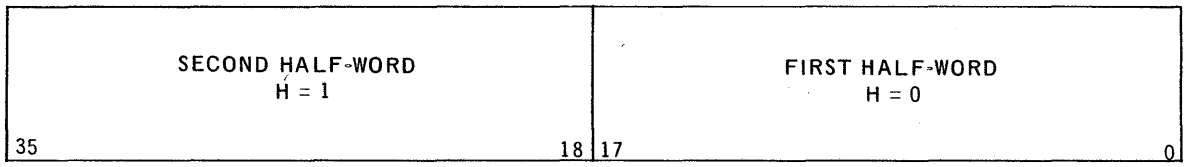
Any I/O channel can be set to ESI mode by means of a switch and a patch card. Furthermore, by means of a patch card, an ESI channel can be set to operate in either half-word (18-bit) or quarter-word (9-bit) mode.

Because an I/O channel can be used by many devices in ESI mode, data flow must be governed by an access control word unique to the device currently in operation rather than to the channel as in ISI. These access control words are stored in storage at relative addresses assigned to the devices. As a device transfers data, it presents the address of its own access control word; thus, no complicated program monitoring is necessary to control data flow. The contents of the 9-bit MSR are used to extend the 15-bit relative address to produce a 24-bit absolute address of the ACW.

The format for the ESI access control word differs somewhat from that for ISI to enable control of half- and quarter-word transfers. The half-word access control word is as follows:



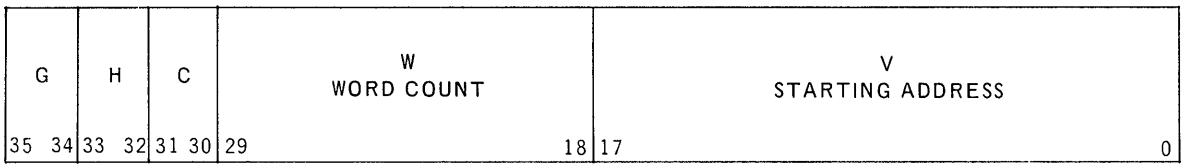
The G, W, and V fields have the same meaning as in the ISI access control word except that W is reduced to 15 bits and counts half-words. There is also a one-bit H field; this field is used to indicate which half of location V is to be used, as follows:



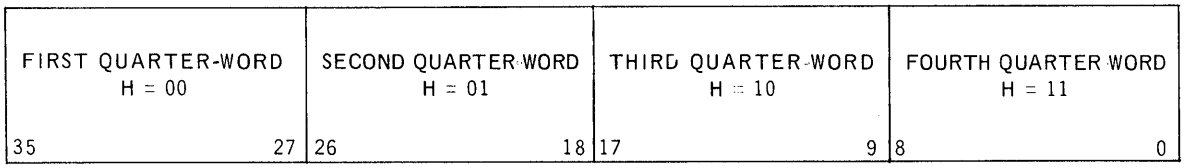
H = 0; use first half-word of location V and switch H to 1.

H = 1; use second half-word of location V, change V address as specified by G, decrement word count W, and switch H to 0.

Quarter-word operations are similar to half-word operations except that additional programmed control is provided in terminating transmission. For this purpose the access control word includes two extra control bits.



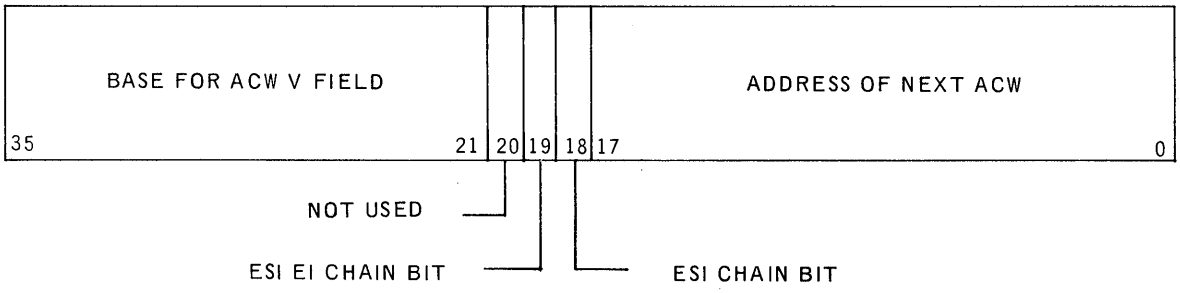
G, W, and V have the same meanings as for ISI though W is now only 12 bits long and now counts quarter-words. H indicates which quarter-word portion of V is being addressed as follows:



C is a two-bit control field for use in output operations that prevents loss of data by generating an extra programmed monitor interrupt, if required, and a programmed end-of-transmission signal. The normal monitor interrupt occurs when W goes from 1 to 0. However when bit 30 is set to 1, the IOAU sends a monitor interrupt to a CAU when W decreases from 2 to 1. Similarly, if bit 31 is set to 1, the subsystem generates the end-of-transmission signal as W goes from 2 to 1.

5.3.1. ESI Data Chain Pointer

Each ESI access control word (ACW) has an associated chain pointer word located in the next higher storage location. The format of the ESI chain pointer word is presented below:



Bits 35–21 Base value that is added to V field of ACW to form absolute address for data or function word transfer.

Bit 20 Zero

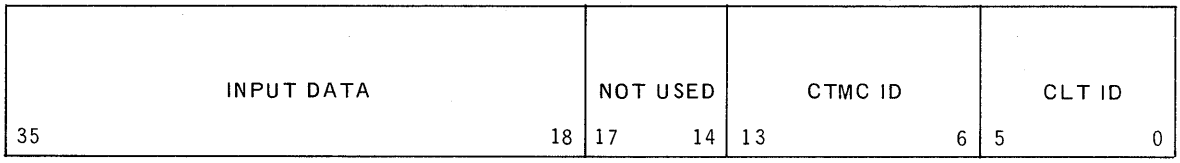
Bit 19 Bit 19 equal to one: specifies data chaining when an ESI EI occurs. Bit 19 equal to zero: specifies no chaining.

Bit 18 Bit 18 equal to one: specifies data chaining for all ESI and ISI input or output. Bit 18 equal to zero: specifies no chaining. Bit 18 must be zero for a function transfer.

Bits 17–0 The relative address of the next ACW for chaining. A one is added to the address for next chain pointer word. The absolute address of the next ACW is formed by adding the contents of the chain base register (CBR) to the relative address (bits 17–0) to form a 24-bit absolute address.

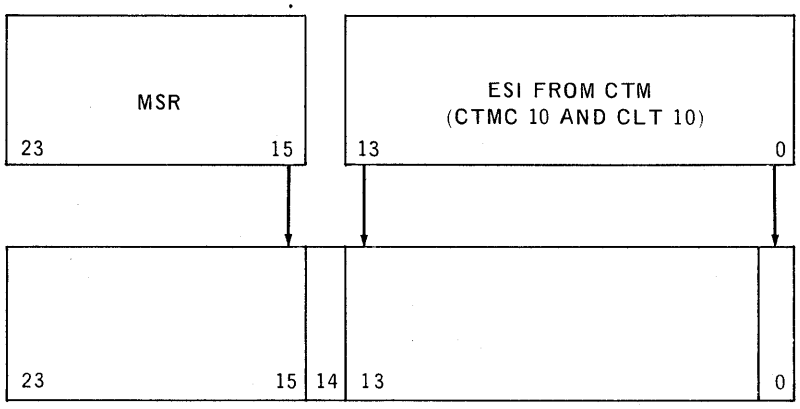
5.3.2. ESI Absolute Address Generation

After the ESI value is accepted by the IOAU, it is realigned and augmented with the contents of the memory select register to form the absolute ESI ACW address as shown in Figure 5–2.



Bit 0 of the absolute address is zero for an ACW address. Bit 0 is one for the ESI data chain pointer word address.

To form the absolute address for ESI data transfers, the contents of the base field at the associated ESI data chain pointer word are added to the V field of the ESI ACW as indicated below:



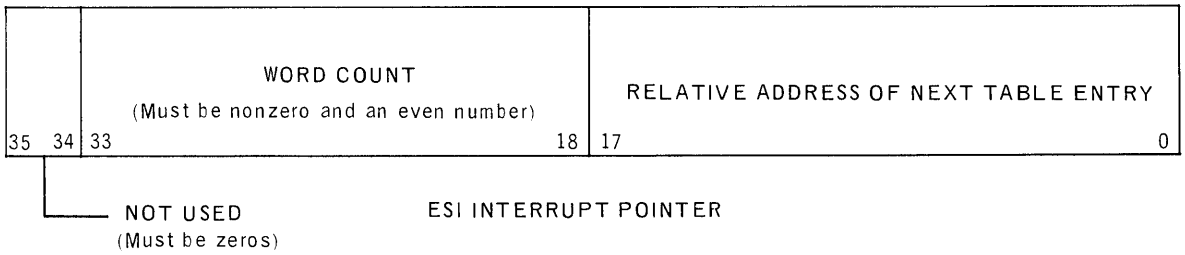
ABSOLUTE ESI ADDRESS

5.3.3. ESI External and Monitor Interrupts

Interrupts on ESI channels are hardware tabled. An interrupt tabling sequence is initiated in the IOAU when an ESI external interrupt occurs or when the word count field of the ESI ACW reaches the terminal condition and an input or output transfer with monitor was specified by the corresponding LOCM or LICM instruction.

Tabling is accomplished using the contents of the channel ISI input access control register as an active ESI interrupt pointer. This pointer word specifies the number of words required to fill the table and the relative address in which to store the next interrupt status word. The channel ISI input chain pointer register holds an auxiliary ESI interrupt table pointer with the same format as the active interrupt table pointer.

The active and auxiliary pointers must be initially loaded by software and do not have to be restored by the program. During a tabling sequence, the 24-bit absolute address of the first of the two consecutive words stored for each table entry is formed by adding bits 17-0 of the active interrupt pointer and the contents of the chain base register (CBR). The contents of the chain base register are added to bit position 9 and above of the relative address to form an absolute address. After each tabling sequence, the word count of the active interrupt pointer is decreased by two and the address is increased by two.

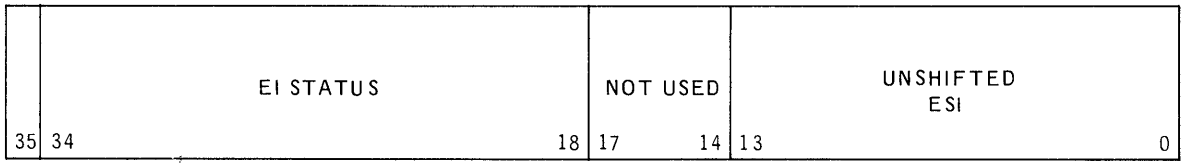


When the word count field of the active interrupt pointer is decreased to zero, the contents of the auxiliary interrupt pointer are transferred to the active interrupt pointer location (ISI input ACW register); and ESI table full interrupt to address 226_8 plus memory select register is initiated.

The program has the option of being notified of ESI channel tabling activity either immediately upon occurrence of the activity, or by having the channel retain the fact that tabling occurred by means of the table activity designator and then interrogating this designator.

The contents of the auxiliary interrupt pointer is not disturbed by the hardware. The software has the option of allowing interrupt tabling to continue in the table or reloading the auxiliary interrupt pointer, thereby chaining the interrupt tables.

Software controls process interrupts in the table ensuring that processing is ahead of tabling. The format of the tabled interrupt status word pair is:



First Word

Bit 35 – Is set in external interrupt status word and cleared for monitor.

Bits 34–18 – Status word for external interrupt and zeros for monitor.

Bits 17–14 – Not used.

Bits 13–0 – ESI.

Second Word

Is the input or output data transfer ACW for the ESI that generated the external interrupt (see 5.3 for format). For input or output monitor, the second word is not written into storage.

5.3.4. ESI Function Word Transfer

When the LFC or LFCM instruction is executed, an output access control word is transferred from the location specified by the instruction to the output access control register associated with the specified output channel. When the instruction is executed, one function word is transferred from the storage location specified by the output access control word. It is transmitted by means of the output word/function word lines of the associated channel to the subsystem.

The contents of the base field of the channel output chain pointer register are added to the V field of the function ACW to form the absolute address.

5.3.5. ESI Buffer Termination and ESI Data Chaining

Whenever the word count field of an ESI input or output data ACW reaches the terminal condition, a sequence to replace it and the associated data CP word is initiated, if bit 18 of the current chain pointer word is a one. Whenever an ESI EI occurs, a sequence to replace the CAW and the associated data CP word are initiated if bit 19 of the current CP word is equal to one. The new ACW and chain pointer word is read from the storage address and the address plus one specified by the address field of the current data chain pointer word plus the chain base register.

ESI data chaining continues until a chain pointer word with a zero for bit 18 is encountered.

6. STORAGE

6.1. GENERAL

The UNIVAC 1110 System incorporates a two-level hierarchy of directly addressable, executable storage. The first level, called main storage, consists of high-speed nondestructive readout plated wire storage. The second level, called extended storage, consists of moderate cost core storage.

Design of storage in this manner not only allows more storage on the system, but permits a choice of storage media according to particular needs. For example, a frequently used compute-bound function could reside in high speed main storage, while the less often used contingency routines associated with it would be in extended storage. Within the system, this two level storage hierarchy is treated as a set of system components in the same manner as peripheral devices, allowing efficiencies not before available in most computer systems.

Among the featured characteristics of the storage hierarchy are:

- Independently accessible modules
- Continuous addressing structure
- Access priority structure in case of conflicts
- Parity checking on data and address
- Interrupt generation in case of address or data parity error
- Ease of storage expansion
- Partial word capability for read and write operations

6.2. MAIN STORAGE

Main storage is composed of fast access nondestructive readout (NDRO) plated wire storage modules and built in Multi-Module Access (MMA) units. Among its features are:

- Nominal 320-nanosecond read and 520-nanosecond write cycles
- 8,192-word modularity for simultaneous access
- Parity checking on addresses and data
- Access through the MMA's by up to four CAU's and four IOAU's
- Expandability in 32,768 word increments up to a maximum of 262K words
- Interleaved access to boost performance and reduce conflicts (odd-even addressing to two adjacent 8K modules)
- Access conflicts resolved on 8K boundaries
- Partitionable in 32K-word increments

6.2.1. Main Storage Unit (MSU)

Each main storage unit (see Figure 6–1) consists of 32,768 words of storage, MMA, power supplies, and maintenance panel. Each word consists of 36 data bits, two parity bits (one bit per half-word), and two spare bits at nonaddressable levels. Four 8,192-word modules constitute a minimum storage unit. This unit may be expanded in one increment by the addition of four 8,192-word modules for a total capacity of 65,536 words. Four fully expanded MSU's of 65K words may be included in a UNIVAC 1110 System for a total main storage capacity of 262,144 words. A 32K main storage unit is capable of simultaneously servicing four requests, one per 8K module, while a fully expanded 65K main storage unit can service up to eight simultaneous requests.

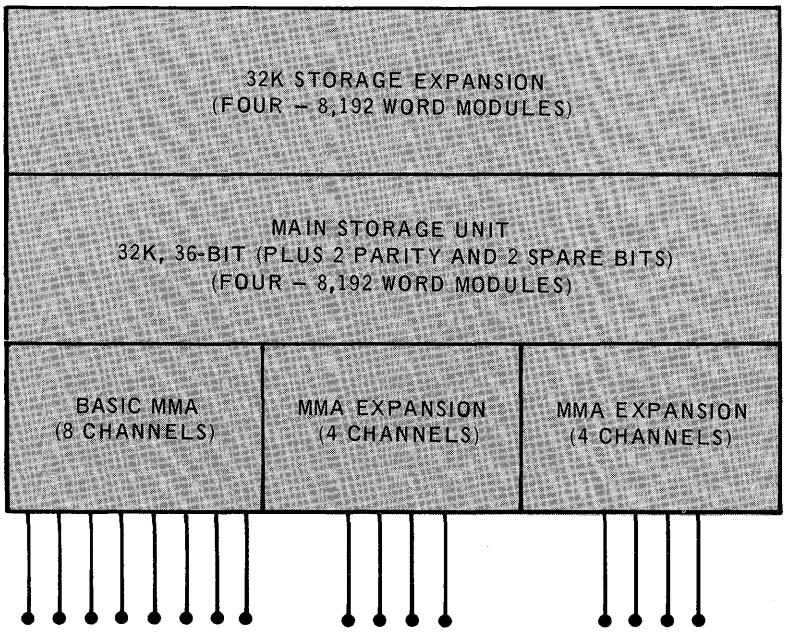


Figure 6–1. Main Storage Unit

6.2.2. Multi-Module Access Unit (MMA)

The multi-module access (MMA) unit is physically contained in the main storage unit cabinet and is functionally located between the main storage unit and the CAUs and IOAUs. The MMA furnishes eight priority-ordered connection paths to each of the main storage modules in the main storage unit. The number of paths to the MMA may be increased to 16 by the use of two MMA expansion features.

Should an access conflict occur among requestors, the multiple module access unit grants main storage access to the processor having the highest priority, then the next, and so on. Communications between a requestor and a single storage module can, therefore, be asynchronous. If the storage module is busy servicing one requestor, a passive wait cycle occurs in requestors of lower priority referencing the same module. Delays in honoring I/O transfers are eliminated as the IOAUs are attached to the higher priority paths of the MMA.

6.3. EXTENDED STORAGE SUBSYSTEM

The extended storage subsystem is composed of moderate cost core storage modules and associated multiple access interfaces (MAI). Among its features are:

- 1.5-microsecond read/write cycle
- 131,072-word modularity
- Parity checking on addresses and data
- Access through the MAI's by up to four CAU's and four IOAU's.
- Expandability in 131,072-word increments up to a maximum of 1,048K words
- Capability of interleaved operation to boost performance and reduce conflicts (Interleaves of 1, 2, or 4 may be selected and changed at the site)
- Partitionable in 131K increments

6.3.1. Extended Storage Unit (ESU)

Each extended storage unit consists of 131,072 words of 1.5-microsecond core storage, power supplies, and maintenance panel. Each word consists of 36 data bits, two parity bits (one bit per half-word), and two spare bits at nonaddressable levels.

UNIVAC 1108 main storage units may optionally be used as extended storage in the UNIVAC 1110 System. For each 65K UNIVAC 1108 storage unit in the system, the 1,048K maximum capacity of extended storage is reduced by 65K. For example, if eight 1108 storage units were used, the extended storage capacity would be reduced to a total of 524K words.

6.3.2. Multiple Access Interface

The MAI, operating in the same manner as the integral MMA on main storage, interfaces with 131K of extended storage and up to four CAU's and four IOAU's An MAI expansion may be added to the MAI, providing identical interfaces for a second extended storage unit. The MAI expansion, functionally a second MAI, shares the same cabinet and power supplies with the MAI to which it is added.

Each MAI (or MAI expansion) interfaces optionally with one 65K UNIVAC 1108 main storage unit.

6.4. PARITY

Parity is checked on addresses presented to MMA's, MAI's, MSU's, and ESU's prior to performing the operation requested. In this way, malfunctions may be isolated, identifying the faulty component. During all read and write operations, data parity is checked at these same levels, providing the highest degree of system integrity. Upon detection of any parity error, an interrupt is generated; the status word associated with the interrupt indicates the error type and point at which the error occurred.

6.5. STORAGE PROTECTION

To prevent inadvertent program reference to out of range storage addresses, the UNIVAC 1110 System includes a hardware storage protection feature. The controlling element in this feature is the storage limits register. The contents of the storage limits register are as follows:

INSTRUCTION AREA		DATA	
UPPER BOUNDARY	LOWER BOUNDARY	UPPER BOUNDARY	LOWER BOUNDARY
35 27	26 18	17 9	8 0

The storage limits register (SLR) can be loaded by the executive system to establish allowable operating areas for the program currently in execution. These areas are termed the program instruction (I) and data (D) areas. Before control is given to a particular program, the executive loads the SLR with the appropriate instruction and data boundaries.

Before each main storage reference, the appropriate CAU performs a limits check by comparing the relative address against the limits of either the I or D field of the storage limits register. An out of limits address generates a guard mode interrupt, thereby allowing the executive to regain control and take appropriate action.

The executive system can establish two different modes of storage protection (see 6.5.1 and 6.5.2) by means of control fields in the processor state register (PSR) (see 4.2). Normally, the executive itself operates in open mode; that is, the storage limits register may be loaded but the PSR is set to disregard this, and the executive can reference any location in main storage.

6.5.1. Privileged Mode

Another mode can be established in the PSR for privileged programs. This privileged mode protects against out-of-bounds writes. Privileged programs (such as real time programs or executive controlled subroutines) may enter nonalterable (re-entrant) subroutines, which are part of the executive. Though these privileged programs are assumed to be thoroughly checked out, the system is still fully protected against unexpected occurrences since write protection is in effect.

6.5.2. User Program Mode (Guard Mode)

In the user program mode, read, write, and jump storage protection is in effect. Therefore, user programs are limited to those areas assigned by the executive. If the user program reads, writes, or jumps to an out-of-limits address, an interrupt returns control to the executive for remedial action.

Read/jump protection allows the executive to stop the program at the point of error, terminate it, and provide diagnostic information to the programmer thereby minimizing wasted time and smoothing the checkout process.

6.6. RELATIVE ADDRESSING

Relative addressing is a feature of great significance in multiprogramming, time-sharing, and real time operations; it allows storage assignments for one program (the one going into execution) to be changed dynamically by the executive to provide continuous storage for operation of another program, and it permits programs to dynamically request additional main storage according to processing needs. An additional advantage is that systems programs stored in auxiliary mass storage may be brought in for operation in any available area without complicated relocation algorithms.

The full direct addressing capabilities of the UNIVAC 1110 System provides for the execution of instructions, the reading or writing of operands, and I/O data transfers throughout the full range of the system.

The UNIVAC 1110 System addressing mode provides an 18-bit relative addressing range of 262K words with any given PSR setting. The user may switch from two hardware PSR settings. Programs may be relocated on 512 block boundaries.

6.7. CHARACTER ADDRESSING

The UNIVAC 1110 System is provided with extended character addressing capabilities and with the ability to manipulate character addresses in essentially the same manner as full-word addresses. There are two character addressing modes:

- The UNIVAC 1110 System character instruction subsets:
  - Byte instructions
  - Byte/binary conversions
  - Decimal arithmetic instructions
- Instructions specifying:
  - Instruction f field values less than 70_8 (except 07_8, 33_8, 37_8)
  - Instruction j field values of 4_8 through 7_8 defined by addressing one of the four j registers located in R registers R6 through R9.
  - Bit D4 in the processor state register specifying character indexing mode

7. PERIPHERAL SUBSYSTEMS

7.1. GENERAL

Peripheral subsystems are attached to the independent input/output access unit (IOAU) through general purpose input/output channels; any peripheral subsystem may be attached to any I/O channel. The governing factor in the choice of an I/O channel for peripheral attachment is the transfer rate of the devices in the subsystem. Since the channel priority is based on channel number, real time equipment or equipment with very high transfer rates should be attached to the lower numbered channels which have the higher priority.

With this adaptable input/output arrangement, the UNIVAC 1110 System can communicate with many real time devices such as analog/digital converters, key sets, communication terminals, tracking and radar systems, display systems, and other information processing systems.

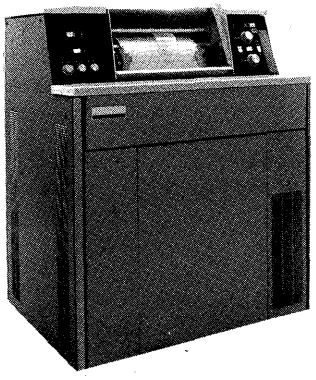
7.2. CENTRAL SITE EQUIPMENT

Central site equipment available for the UNIVAC 1110 System comprises six types of peripheral subsystems:

- Multiple High-Speed Printer Subsystem
- Punched Card Subsystem
- Magnetic Tape Subsystems
- Mass Storage Subsystems
- UNIVAC 9200/9300 Onsite/Remote Subsystems
- Communications/Symbiont Processor (C/SP)

The paragraphs which follow describe each subsystem, including associated components of the subsystem; the C/SP is described in Section 8.

7.2.1. UNIVAC Multiple High-Speed Printer Subsystem



The UNIVAC multiple high-speed printer subsystem provides the UNIVAC 1110 System with an output printing unit that is capable of printing single or multiple copies of data. Each line of output data may contain up to 132 printed characters. Printing operations which occur on a request/acknowledge basis allow the processor to perform other processing functions while the printer is printing data.

This subsystem contains a high-speed printer control unit connected to one or two high-speed printer mechanisms capable of printing from 1200 lines per minute for a full 63-character set to 1600 lines per minute for a 43 sequential character set. The printer contains 63 printable characters: the 26 letters of the alphabet, the ten arabic numerals, and 27 special characters. Different symbols may be factory supplied upon order.

CHARACTERISTICS	
Printing speed (with single-line spacing)	1200/1600 lines per minute maximum, depending upon the number of sequential characters to be printed on each line.
Line spacing speed	11.5 ms for spacing first line and for spacing each subsequent line as follows: 5.06 ms per line at 6 lines per inch 5.7 ms per line at 8 lines per inch
Character per line	132 characters (including spaces) per line.
Spacing of characters	0.1 inch along print line.
Ribbon feed	Bidirectional, self-reversing, self-correcting.
Type of ribbon	Fabric ribbon interchangeable with carbon Mylar* ribbon (optional) for "one-time" operation.
Vertical line spacing	Manually selected: Either 6 lines per inch or 8 lines per inch. Form loop control: As determined by code punched in form control tape.
Number of characters	Up to 63 different characters: standard font consists of alphabetic characters A–Z, numeric characters 0–9, 27 punctuation marks and symbols. Modified fonts available upon request.
Print format	Full print width of 132 characters can be placed anywhere on 16.5 inch form. With 22 inch width form, only central 13.2 inch portion can be used. Format variation under full control of programming.
Paper forms	Continuous forms with standard edge sprocket holes from 4 to 22 inches in width. Carbons may be attached or unattached with multicopy forms up to a maximum of six parts. Recommended pack thickness up to .0155 inch for optimum print quality.
Paper container	Maximum dimensions accommodated entirely within base of machine: 16 inches high, 16 inches long, and 22.5 inches deep.

*DuPont trademark for its polyester film

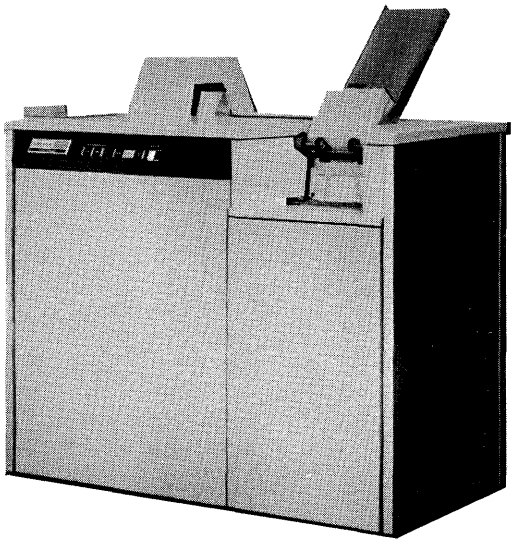
7.2.2. Punched Card Subsystem

The UNIVAC punched card subsystem consists of a UNIVAC 900 cpm Card Reader and a UNIVAC 300 Card Punch which are attached to a control unit on a single input/output channel of an IOAU.

The card reader uses column-parallel photodiode sensing with automatic photodiode checking, error cards being ejected into a separate stacker. A file feed device is standard. Data from the reader may be translated into Fieldata code before transfer to main storage or transferred directly in row or column binary.

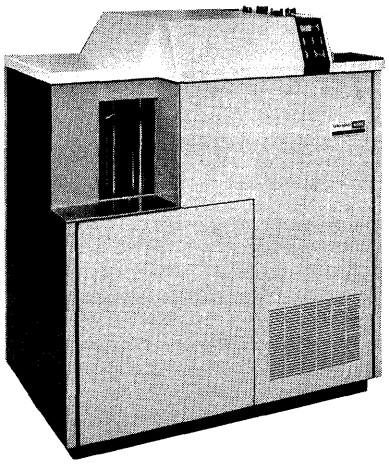
The card punch operates on a row-by-row basis and has an automatic check-read station. Incorrectly punched cards are passed to the error stacker and additional attempts to punch the data may be made under program control. Data may be punched in Fieldata code or in row and column binary.

7.2.2.1. UNIVAC Card Reader



CHARACTERISTICS	
Card reading speed	900 cards/minute
Input hopper capacity	3000 cards
Output stacker capacity	2100 cards
Reject Stacker capacity	100 cards
Read modes	Fieldata, column binary, row binary
I/O channels	1 shared with card punch

7.2.2.2. UNIVAC Card Punch



CHARACTERISTICS	
Card punching speed	300 cards/minute
Input hopper capacity	1000 cards
Output stacker capacity	2 stackers of 850 cards each
Punch modes	Fieldata, column binary, row binary
I/O channels	1 shared with card reader

7.2.3. UNISERVO Magnetic Tape Subsystems

Four magnetic tape subsystems are available with the UNIVAC 1110 System:

UNISERVO VIII-C Magnetic Tape Subsystem
UNISERVO 12 Magnetic Tape Subsystem
UNISERVO 16 Magnetic Tape Subsystem
UNISERVO 20 Magnetic Tape Subsystem

Two basic methods of operation are available:

- Single-Channel Operation

  One or more tape units are connected to a single I/O channel through the appropriate control units. Only one function on any one of the tape units may be active at any single instant.

- Simultaneous Operation

  Two or more tape units are connected to two I/O channels through the appropriate control units just as in dual-channel operation. Additional circuits in the control units and the tape units, however, permit simultaneous read/read, read/write, write/read, and write/write on any two tape units.

The following chart indicates the availability of the various methods of operation for the tape subsystems:

	SINGLE CHANNEL	SIMULTANEOUS DUAL ACCESS
UNISERVO VIII-C Subsystem	X	X
UNISERVO 12 Subsystem	X	
UNISERVO 16 Subsystem	X	X
UNISERVO 20 Subsystem	X	X

Magnetic tape subsystems may consist of up to 16 tape units with the appropriate control units. Systems are available for both 7- and 9-track operation. The 7- and 9-track options permit data recorded in the traditional industry-compatible form to be handled, and yet at the same time allow the upgrading of these records in line with the ASCII code and packed-decimal formats.

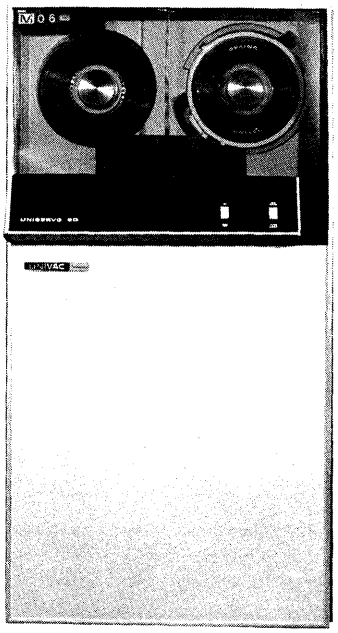
Those systems employing either UNISERVO 16 or UNISERVO 20 control units may be configured with mixed tape units; that is, systems can use a mixture of UNISERVO 12/16 tape units on the UNISERVO 16 control or a mixture of UNISERVO 12/16/20's and UNISERVO 20 control. Such a mixed system provides a useful flexibility by combining high-speed units with economic medium-speed units.

The UNISERVO 12 and 16 subsystems are connected to the UNIVAC 1110 System through the Multi-Subsystem Adapter (MSA). The MSA converts the byte output of the UNISERVO 12 and 16 tape units into a 36-bit format suitable for use in the UNIVAC 1110 System. On output, the MSA performs the word-to-byte conversion. The MSA also provides for data translation, function chaining, and command chaining by means of the appropriate features.

The UNISERVO 20 control unit includes the functional capabilities of the MSA as an integral part.

A Shared Peripheral Interface (SPI) feature is provided in the UNISERVO subsystems to permit access to up to four processor IOAU's.

7.2.3.1. UNISERVO 20 Magnetic Tape Subsystem



The UNISERVO 20 Magnetic Tape Subsystem represents the highest performance tape handling capability offered by UNIVAC. This subsystem consists of a control unit and from one to sixteen magnetic tape units. Another control unit may be used to achieve simultaneous dual access operation when appropriate features are present on the magnetic tape units. UNISERVO 12's and 16's may also be used with the UNISERVO 20 control unit, thereby providing tape handling flexibility in terms of price and performance. The UNISERVO 20, as well as the UNISERVO 12 and 16, provide tape compatibility with the IBM* 2400 Series Magnetic Tape Units which utilize the 9-track 1600 CPI phase encoded tape format.

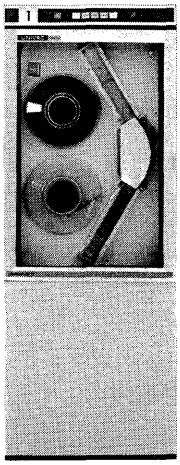
The UNISERVO 20 tape unit provides operational conveniences such as power window, automatic tape threading, and a wrap-around tape cartridge. This tape unit provides for the reading of tape during write operations for purposes of checking what has just been written. The 200-inch-per-second tape speed provides for a transfer rate of 320,000 frames per second.

The simultaneous dual access configuration provides for read/write, read/read, and write/write operation in any two individual UNISERVO 16's or 20's. In addition to doubling the performance of the subsystem, complete redundancy is achieved by virtue of individual power supplies for each control unit and independent access paths to each UNISERVO 16/20 tape unit. Data validity checking facilities include longitudinal redundancy check for 7 and 9 track NRZI tapes, vertical redundancy, and cyclic redundancy for 9 track NRZI tapes.

*IBM is a proprietary designation of the International Business Machines Corporation.

CHARACTERISTICS	
Transfer rate	320,000 frames/second
Recording density	1600 ppi
Tape speed	200 inches/second
Tape width	0.5 inch
Tape length (max.)	2400 feet
Thickness	1.5 mills
Block length	Variable
Space between block	0.6 inch
Tracks on tape	9 tracks, 8 data, 1 parity
Units per control	16
Standard features	Backward Read, Automatic Tape Threading, Power Window, Cartridge Loading, Quick Release Hub
Input/output channels required	1 or 2

7.2.3.2. UNISERVO 12 Magnetic Tape Subsystem



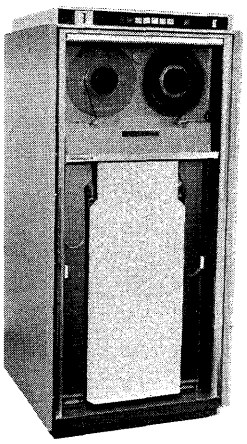
The UNISERVO 12 Magnetic Tape Subsystem is a low-cost medium performance subsystem with 7- or 9-track, 200, 556, or 800 ppi NRZI, and 1600 ppi phase encoding as the available tape formats. One master tape unit, with a power supply and control circuits, controls up to three slave units. Up to 16 tape units are synchronously controlled by a UNISERVO 12/16 control unit.

This subsystem offers a peak transfer rate of over 68,000 frames per second in the primary tape format. The 7- or 9-track NRZI formats with a transfer rate of over 34,000 frames per second also can be incorporated. Tape data validity checking facilities include read check while writing, longitudinal redundancy check, vertical parity check (9-track phase tapes), and cyclic redundancy check (diagonal, 9-track NRZI tapes).

"On the fly" single-track read error correction is standard for phase tapes. On 9-track NRZI tapes, single-track read error correction is provided by a second attempt at an operation after error detection and repositioning. This provides the ability to correct tape errors in either the forward or backward direction; this also simplifies the error correction programming routines and assists in the recovery of unusual error conditions which otherwise would result in a non-recoverable error. A programmable low gain read assists in the reading of tape records containing high noise levels.

CHARACTERISTICS	
Transfer rate	68,320 frames/second
Tape speed	42.7 inches/second
Tape direction Reading Writing	 Forward or backward Forward
Tape width	0.5 inch
Tape length (max.)	2400 feet (plastic)
Thickness	1.5 mils
Block length	Variable
Interblock gap	0.75 inch (7-track) 0.6 inch (9-track)
Interblock gap time (7-track)	17.6 milliseconds (nonstop) 23.6 milliseconds (start/stop)
Interblock gap time (9-track)	14.1 milliseconds (nonstop) 20.1 milliseconds (start/stop)
Reversal time	25 milliseconds
Rewind time	3 minutes (2400 feet)
Dual density	Feature available

7.2.3.3. UNISERVO 16 Magnetic Tape Subsystem

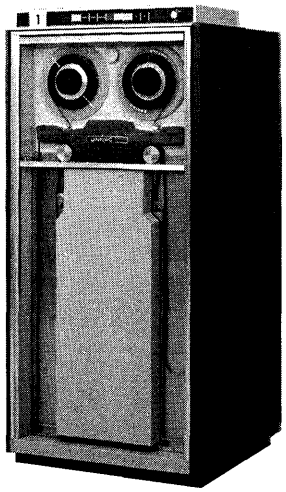


CHARACTERISTICS	
Transfer rate	192,000 frames/second
Tape speed	120 inches/second
Tape direction Reading Writing	 Forward or backward Forward
Tape width	0.5 inch
Tape length (max.)	2400 feet (plastic)
Thickness	1.5 mils
Block length	Variable
Interblock gap	0.75 inch (7-track) 0.6 inch (9-track)
Interblock gap time (7-track)	6.25 milliseconds (nonstop) 9.25 milliseconds (start/stop)
Interblock gap time (9-track)	5.0 milliseconds (nonstop) 8.0 milliseconds (start/stop)
Reversal time	10 milliseconds
Rewind time	2 minutes (2400 feet)
Dual density	Feature available
Dual access operation	Feature available

The UNISERVO 16 Magnetic Tape Subsystem is similar to the UNISERVO 12 Magnetic Tape Subsystem except that it has higher performance capabilities. The peak transfer rate is over 192,000 frames per second in the primary tape format (9-track, 1600 frames per inch, phase encoding), and 96,000 frames per second in the NRZI format.

The UNISERVO 16 dual access feature provides a more powerful subsystem in that it provides simultaneous read/read, read/write, and write/write operation on an individual tape unit basis. Furthermore, it provides complete subsystem redundancy by the addition of individual power supplies for each control unit and independent access paths to each tape unit. Data validity checking facilities include read check while writing, longitudinal redundancy check, and cyclic redundancy check (diagonal) on 9-track NRZI tapes, and vertical parity check on 9-track phase tapes. The UNISERVO 16 subsystem is a freestanding, independent tape unit, that is, the master/slave approach is not used.

7.2.3.4. UNISERVO VIII-C Magnetic Tape Subsystem



A UNISERVO VIII-C Magnetic Tape Subsystem has up to 16 magnetic tape units and incorporates either one or two control units attached to one or two input/output channels for single or dual access operation.

UNISERVO VIII-C units may be specified with 7- or 9-track mode. In 7-track mode, one parity and six data bits are recorded in each frame across the width of the tape. A single 6-bit alphanumeric character or a 6-bit binary value may be stored per frame. In 9-track mode one parity and eight data bits are recorded in each frame across the width of the tape.

Data packing density in 7-track mode is set either by the program or by a manual switch on each unit to either 200, 556, or 800 frames per inch. Physical tape speed is 120 inches per second giving maximum transfer rate of 24,000, 66,720 and 96,000 alphanumeric characters per second. Data packing density in 9-track mode is 800 frames per inch giving a maximum rate of 96,000 bytes per second.

A higher character transfer rate results if 6-bit characters are read or written in 9-track mode. This method of operation yields a transfer rate of 128,000 characters per second.

The rewinding rate is 240 inches per second; full reel of 2400 feet can be rewound in 120 seconds. The 800 frame per inch packing density is normally used, the 200 and 556 densities being used only for 7-track mode compatibility purposes.

Reading may take place with the tape moving either forward or backward, an ability valuable for saving rewind time especially during sort/merge operations. Writing takes place only when the tape is moving forward.

Data may be recorded in variable-length blocks under program control with character and block (horizontal and vertical) parity. A read-after-write head allows immediate verification of all data written. Under the control of the software input/output handler, repeated read and write operations are undertaken in an attempt to recover from an error.

Programming problems with this tape subsystem are insignificant since the control unit, combined with the executive input/output handler, deals with all operations except the system response to a nonrecoverable error.

UNISERVO VIII-C tape units are fully compatible with IBM 727, 729 Models I through VI, and 7330 units in 7-track mode, and with IBM 2400 Series Models 1 through 3 units in 7-track mode, and with industry-compatible units produced by other manufacturers. The UNISERVO VIII-C control unit can be furnished with a hardware translator to convert between tape code and Fieldata code, thus ensuring tape compatibility among installations.

CHARACTERISTICS	
Transfer rate	24,000, 66,720, and 96,000 alphanumeric characters per second
Recording density	200, 556, and 800 frames per inch.
Tape speed	120 inches per second
Tape width	0.5 inch
Tape length	2400 feet
Thickness	1.5 mils.
Block length	Variable
Space between block	0.75 inch (7-track) 0.6 inch (9-track)
Tracks on tape	7-tracks: 6 data, 1 parity Optional, 9-tracks: 8 data, 1 parity
Units per control	16
Standard feature	Backward Read
Processor input/output channels	1 or 2

7.2.4. Mass Storage Subsystems

To provide mass storage, three types of UNIVAC peripheral subsystems are available: FASTRAND mass storage (see 7.2.4.1), UNIVAC disc (see 7.2.4.2), and UNIVAC Flying Head drum (see 7.2.4.3).

7.2.4.1. FASTRAND Mass Storage Subsystems

The FASTRAND Mass Storage Subsystems (FASTRAND II and III) provide very large capacity random access storage. Great flexibility is provided by the availability of both a single- and a dual access subsystem. FASTRAND mass storage units include two large magnetic drums, which, like those used in the UNIVAC FH-432 and FH-1782 subsystems, employ flying heads. However, to reduce cost, only a limited number of read/write heads are used. These move laterally over 192 recording tracks.

There are 64 read/write heads per unit, gang-mounted on a common positioning mechanism. As a result, the subsystem positions all of the heads in a drum unit with one movement of its positioning mechanism in an average time of 57 milliseconds. The maximum head positioning time is 86 milliseconds; the minimum is 30 milliseconds. Average latency is half a drum revolution time (35 milliseconds).

Access time, therefore, varies from less than one millisecond (when a head is already positioned over the desired track and latency at its minimum) to 156 milliseconds (for maximum head movement and maximum latency). The average is 92 milliseconds which can usually be reduced by good system design and data layout.

An independent position control feature in each FASTRAND mass storage unit allows greater flexibility and decreases average access time. This is done in a multi-unit subsystem by concurrently prepositioning the heads in a number of drum units. Prepositioning the heads saves time because once the position instructions have been transmitted to one FASTRAND mass storage unit, the executive system can immediately initiate another operation on a different FASTRAND mass storage unit without waiting for completion of the positioning operation. Whenever the system reads from or writes on such a prepositioned unit, the 30 to 86 millisecond positioning delay is avoided. These are the mechanical design features which contribute to the FASTRAND mass storage unit's operating speed.

In any effort to reduce processing time, offline search is an important advantage. In this operation the IOAU instructs the FASTRAND mass storage subsystem to locate a specific piece of data, and then goes on with other processing while the storage search takes place. When the subsystem finds the data, it notifies the IOAU and sends it the data. Also, all other functions of the FASTRAND mass storage unit permit the computer to continue its work while records are being read from or written on the drums.

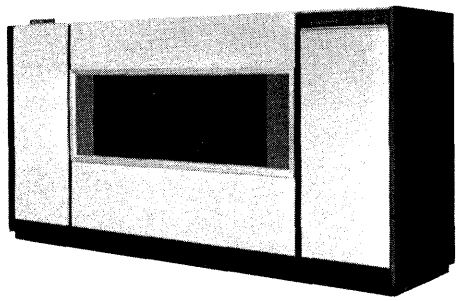
All data on a FASTRAND mass storage subsystem is recorded in 28-word groups known as sectors. Parity is recorded by sector and the parity bits are automatically checked.

A single FASTRAND mass storage subsystem can accommodate up to eight FASTRAND mass storage units. Any of the input/output channels can accommodate a FASTRAND mass storage subsystem. Like the UNIVAC FH-432/1782 drum subsystems, either single channel or dual access operation is possible, providing full-scale two-drum unit simultaneity.

The dual-access FASTRAND subsystem includes two complete independent control units which provides the parallel data paths that permit simultaneous operations on any two FASTRAND mass storage units in the subsystem.

Physically, the FASTRAND III mass storage units are similar to the FASTRAND II mass storage units. The basic difference is in their recording densities (over 1500 bits per inch for FASTRAND III mass storage units versus 1000 bits per inch for FASTRAND II mass storage units). Since the recording density is 50 percent greater, the storage capacity and word and character transfer rates of the FASTRAND III mass storage units are 50 percent greater than those of the FASTRAND II mass storage units.

FASTRAND II AND III MASS STORAGE UNITS



CHARACTERISTICS	
Storage capacity (per unit)	FASTRAND II: 22,020,096 36-bit words (132,120,576 alphanumeric characters) FASTRAND III: 33,030,144 36-bit words (198,180,864 alphanumeric characters)
Average access time	92 milliseconds
Recording density	FASTRAND II: 1000 bits per inch FASTRAND III: 1500 bits per inch
Tracks per inch	105
Drum speed	880 revolutions per minute
Moveable read/write heads	64
Character transfer rate	FASTRAND II: 157,696 characters per second FASTRAND III: 236,547 characters per second
Word transfer rate	FASTRAND II: 26,283 words per second FASTRAND III: 39,424 words per second
I/O channels	1 or 2 per subsystem
Number of units per subsystem	8

7.2.4.2. UNIVAC Disc Subsystems

Univac offers two types of disc storage subsystems for use on the UNIVAC 1110 System. They are:

UNIVAC 8414 Disc Subsystem
UNIVAC 8440 Disc Subsystem

The advantages achieved by implementing UNIVAC Disc subsystems include:

- substantially increased throughput performance
- provision for incremental growth
- expanded potential for online processing
- enhanced capabilities for real time and multiprogramming

Disc subsystems provide the UNIVAC 1110 System with an expandable, removable, direct access, external storage medium.

Data is transferred between the IOAU and the subsystem one word at a time for the 8414 subsystem. The Multi-Subsystem Adapter (MSA) translates the byte-oriented code of the disc into the 36-bit word format of the UNIVAC 1110 System.

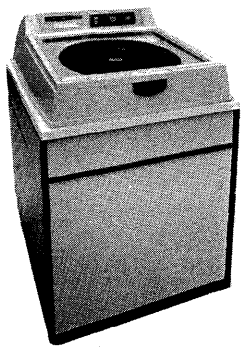
The MSA also provides for data translation, function chaining, and command chaining. These MSA capabilities are an integral part of the 8440 control unit.

The UNIVAC disc subsystems offer many processing advantages, especially in applications where rapid file processing and sort/merge routines are prevalent. The removability characteristics of the disc packs permit virtually unlimited off-line storage and easy interchange of information without conversion to other media.

The UNIVAC disc subsystems also provide for simultaneous dual access operation, prepositioning of access arms, and, with the 8440 subsystem, angular addressing. This implies:

- Simultaneous read/read, read/write, write/write concurrent with positioning functions.
- Alternate data and command paths available to any component of the system.

(1) UNIVAC 8414 Disc Subsystem



The UNIVAC 8414 Disc Subsystem offers a large storage capacity of up to 39.44 million 36-bit words of data online. A single disc pack provides for 5.0 million 36-bit words.

Each disc pack contains 11 discs. Twenty read/write heads are mounted on a single accessor mechanism which moves the 20 heads in unison between the periphery and the central area of the disc. The accessor mechanism can assume one of 203 tracks across the disc surface. This simultaneous head movement creates 203 addressable data recording cylinders in the disc pack, with three cylinders reserved as alternate tracks. Each cylinder contains twenty tracks, number 0 through 19. The addressing of an individual track in the pack is by track number (000–202) and by read/write head number (0–19).

Access to different tracks within a cylinder is faster than access to tracks in different cylinders since changing tracks requires only electronic switching whereas accessing a different cylinder requires physical movement of the accessor mechanism. There are 4060 (203 x 20) tracks in a disc pack assembly. Data capacity figures are based on 4000 tracks, thus allowing for 60 spare tracks in a disc pack assembly.

CHARACTERISTICS	
Number of drives per subsystem	2–8
Number of disc packs per drive	1
Number of R/W head accessor mechanisms	1
Number of R/W heads per disc pack	20
Number of tracks per disc surface	203
Number of recording surfaces per disc pack	20
Number of addressable tracks per surface	203
Number of addressable tracks per disc pack	4060
Number of words per sector	112
Number of sectors per track	11*
Capacity 36-bit words per disc pack	5.0 million*
Minimum access time	20 ms.
Average access time	60 ms.
Maximum access time	130 ms.
Disc pack speed	2400 rpm
Data transfer rate	69,444 words/second

*Using simulated FASTRAND format

(2) UNIVAC 8440 Disc Subsystem

The UNIVAC 8440 Disc Subsystem represents the highest performance disc subsystem offered by Univac. A single subsystem may provide up to 152 million 36-bit words of direct access storage with an average access time of 35 milliseconds.

Each removable disc pack consists of 11 discs with data recorded on 19 surfaces. The access mechanism is a combination of nineteen arms mounted side by side in two groups of ten each. The arms are mounted on a carriage block which is moved to any of the 406 cylinder positions. At each cylinder any one of the nineteen disc tracks can be accessed by selecting the related head. The vertical alignment of tracks can be thought of as a cylinder of tracks.

After a seek operation has moved the access mechanism to one of the 406 cylinders of data, head selection permits the reading or writing of data on any of the nineteen tracks in the cylinder. However, by virture of the angular positioning and priority control technique employed in 8440 control units, an angular address byte defines one of the 128 angular sectors around the circumference of the disc file. Thus, when the beginning of the addressed sector reaches the read/write head, the angular positioning is signaled and I/O orders can be executed with minimum record searching. By overlapping seeks between disc units, the execution of I/O orders will be arithmatically queued in the most efficient manner with priority considerations.

The access to different tracks within a cylinder is faster than access to tracks in different cylinders, since changing tracks requires only electronic switching whereas accessing a different cylinder requires physical movement of the accessor mechanism. There are 7714 (406 x 19) tracks in a disc pack assembly. Data capacity figures are based on 8000 tracks, thus allowing for 120 spare tracks in a disc pack assembly.

CHARACTERISTICS	
Number of drives per subsystem	1–8
Number of disc packs per drive	1
Number of R/W head accessor mechanism	1
Number of R/W heads per disc pack	19
Number of tracks per disc surface	406
Number of recording surfaces per disc pack	19
Number of addressable tracks per surface	406
Number of addressable tracks per disc pack	7714
Number of words per sector	112
Number of sectors per track	22*
Capacity 36-bit words per disc pack	19.0 million*
Minimum access time	10 ms.
Average access time	35 ms.
Maximum access time	70 ms.
Disc pack speed	2400 rpm
Data transfer rate	138,888 words/second

*Using simulated FASTRAND format

7.2.4.3. Flying Head Drum Subsystems

The UNIVAC flying head (FH) series of high-speed large-capacity magnetic drum storage units provide modular auxiliary storage essential for the operation of large and complex systems. These units vary from the ultra-fast UNIVAC FH-432 drum (with an average access time of 4.3 milliseconds) to the large capacity (12.5 million alphanumeric characters) UNIVAC FH-1782 drum which provides extensive fast access storage that can be used for large data files that have to be referenced frequently.

UNIVAC FH Magnetic Drum Subsystems have an individual read/write head for each track. Thus any word on an FH series drum is available to the system in an average access time of 4.3 milliseconds (FH-432) or 17.0 milliseconds (FH-1782).

Each word in all UNIVAC FH subsystems is individually addressable so that the fullest use can be made of premium storage. This enables offline search operations in which the control unit compares each word of any drum area, up to the capacity of the subsystem, with a designated identifier word. Upon finding a match, it supplies the address of the match or commences reading and transferring data to main storage. This entire process is carried out offline without any processor attention once the input/output search function has been initiated and the identifier word designated. This feature is frequently used in the scanning of large data tables when the exact location of an item is unknown.

The transfer rate of data to and from the FH drum subsystem is in line with the ultrafast computing power available. The standard rate is 1,440,000 alphanumeric characters per second. By means of a field option, drum transfer rates may be matched to system loads by interlacing to provide transfer rates of 720,000; 360,000; 180,000; or 90,000 alphanumeric characters per second.

Through the addition of shared peripheral interfaces (SPI), a single- or dual-access UNIVAC FH drum subsystem may be accessed by multiple I/O channels. This not only provides a safeguard in case of failure but also permits all IOAU's to access all drum units.

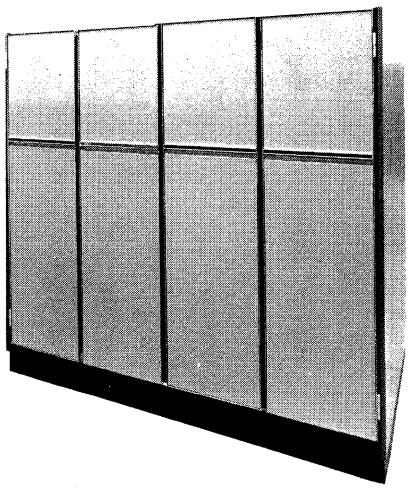
The FH-432/1782 drum subsystems may operate with either one or two control units, using one or two input/output channels. Availability of two channels permits simultaneous read/read, read/write, write/read and write/write operations on any two drum units of the subsystem (a search function is classified as a read function). As an additional reliability measure, each control unit of a dual access subsystem has its own power supply; therefore, in case of a failure of one of the power supplies, the subsystem can still operate on a single-channel basis.

These FH drum subsystems have many advantages in standard data processing as well as real time operation. This is especially true in applications where rapid file processing and sort/merge routines are more prevalent.

Large capacity with rapid access affords convenient intermediate storage. Instead of multiple tape units, the use of the drum subsystems frees the tape units for primary input/output demands.

Drum subsystems allow an extensive executive control system without undue utilization of storage. The short access time of the FH-432 drum permits lesser-used control segments to be stored outside of directly addressable storage. They can then be read into a common overlay area only when required. This arrangement greatly reduces the amount of storage required for the executive system.

(1) UNIVAC FH-432 Magnetic Drum Subsystem



The FH-432 Magnetic Drum Subsystem is designed for single-dual-channel operation. A minimum UNIVAC FH-432 Magnetic Drum Subsystem includes two drums (524,288 36-bit words of storage), a control unit, and power supplies, contained in two cabinets. To augment the systems, cabinets may be added, each containing one or two drums with a storage capacity of 262,144 36-bit words per drum. Of the 432 tracks on each drum, 384 are used for data; the remaining tracks are used for spares, parity, and timing functions. There are 2048 words of data per three tracks. Reading and writing are 3-bit parallel operations on all three tracks of a band simultaneously. Thus the maximum transfer rate is 240,000 words or 1,440,000 alphanumeric characters per second.

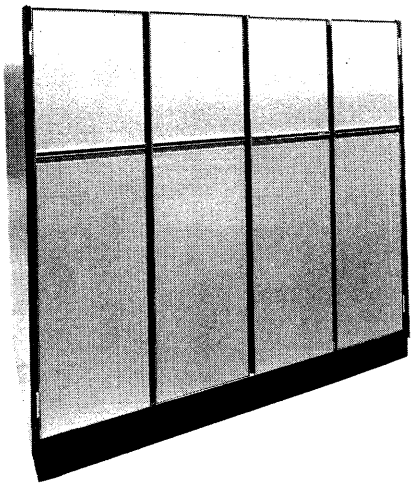
Up to eight FH-432 magnetic drums may be accommodated in a single subsystem, affording a maximum subsystem capacity of 2,097,152 words or 12,582,912 alphanumeric characters.

FH-432 drum units may be intermixed with FH-1782 drum units in the same subsystem to provide a powerful blend of ultrahigh speed and large capacity storage. This mixed subsystem is described in 7.2.4.3.

CHARACTERISTICS

Storage capacity	262,144 computer words of 36 data bits plus parity bits or 1,572,864 alphanumeric characters per drum
Average access time	4.3 milliseconds
Drum speed	7200 revolutions per minute
Number of read/write heads	432 – one per track
Character transfer rates	1,440,000, 720,000, 360,000 180,000, 90,000
Word transfer rates	240,000, 120,000, 60,000, 30,000, 15,000
I/O channels required	1 or 2 per subsystem
Number of drums per subsystem	2 to 8 (12,582,912 characters maximum)

(2) UNIVAC FH-1782 Magnetic Drum Subsystem



The UNIVAC FH-1782 Magnetic Drum is similar to the FH-432 drum except that the storage capacity is approximately eight times greater per drum; this increase is achieved partly by an increase in the number of data tracks to 1536 and partly by an increase in the diameter of the drum. Each track has its own read/write head, and average access time is 17 milliseconds.

A single FH-1782 drum stores 2,097,152 words equivalent to 12,582,912 alphanumeric characters. Up to eight FH-1782 drums can be accommodated in a single subsystem giving a subsystem capacity of 100,663,296 characters.

The character transfer rate is equal to that of the FH-432 drum; this arrangement enables FH-1782 drums to be associated with FH-432 drums in the same subsystem as described in (3).

CHARACTERISTICS	
Storage capacity	2,097,152 computer words of 36 data bits plus parity bits or 12,582,912 alphanumeric characters per drum.
Average access time	17 milliseconds
Drum speed	1770 revolutions per minute
Number of read/write heads	1782 (33 blocks with 54 heads per block)
Character transfer rates	1,440,000; 720,000; 360,000; 180,000; 90,000/sec.
Word transfer rates	240,000; 120,000; 60,000; 30,000; 15,000/sec.
I/O channels required	1 or 2 per subsystem
Number of drums per subsystem (maximum)	8 (total of 100,663,296 characters)

(3) UNIVAC FH-432/FH-1782 Magnetic Drum Subsystem

A valuable characteristic of this subsystem is the ability to associate, in the same subsystem, the ultrahigh speed FH-432 drum with the fast high-capacity FH-1782 drum. Any combination of eight drums may be mixed on a subsystem.

This subsystem arrangement is of significant importance in the UNIVAC 1110 storage configuration. An efficient blend can be made of high-speed storage for rapidly required software, program segments, tables, and indexes and greater access time but large capacity storage for less frequently used program segments, data files, and message assembly/disassembly areas. A judicious mix of speed, capacity, and economy can be planned and the mix can readily be altered as requirements change. Character transfer rates are identical for the FH-432 and FH-1782 drum units. The only functional difference in a data transfer is the variation in access time.

This subsystem is available in both single channel and dual access versions. The dual access version includes two electronically and logically independent control units each on a different I/O channel. This enables simultaneous operation of any two drums in the subsystem and provides the hardware redundancy required for multiprocessing.

7.2.5. UNIVAC 9200/9300 Onsite Subsystems

When a UNIVAC 9200/9300 subsystem is available at the central site, it is connected to the UNIVAC 1110 System by use of the intercomputer control unit (ICCU). When the subsystem is used for remote communication with the UNIVAC 1110 System, the required interface is the data communications subsystem (DCS). For further information concerning the use and the characteristics of the 9200/9300 subsystems, refer to 7.3.6.

7.3. COMMUNICATIONS SUBSYSTEMS

UNIVAC communications subsystems available for the UNIVAC 1110 System are:

- 1110 Communications Subsystem
- Communications/Symbiont Processor (C/SP)
- Data Communication Terminal (DCT) 500
- Data Communication Terminal (DCT) 1000
- Data Communication Terminal (DCT) 2000
- UNISCOPE 100 Visual Communication Terminal Subsystem
- 9200/9300 Onsite/Remote Subsystems
- Data Communication Subsystem (DCS)

The paragraphs which follow describe each subsystem including components, special features, and characteristics of the subsystem; the C/SP is described in Section 8.

7.3.1. UNIVAC 1110 Communications Subsystem

The UNIVAC 1110 Communications Subsystem enables the UNIVAC 1110 System to receive and transmit data by way of any common carrier at any of the standard rates of transmission up to 50,000 bits per second. It can receive data from or transmit data to low-speed, medium-speed, or high-speed lines in any combination.

As illustrated in Figure 7–1, the subsystem consists of two principal elements. The UNIVAC communications terminal module (CTM) makes direct connection with the communication facilities. The UNIVAC communications terminal module controller (CTMC) transmits data between the modules and the central processor. A communications terminal module controller may be connected to any IOAU channel, multiplexing up to 16 CTMs to that channel.

There are three basic types of input and output CTMs: low speed (up to 300 bits per second), a medium speed (up to 1800 bits per second), and high speed (2000 to 50,000 bits per second). Each is easily adjusted to the speed and other characteristics of the type of line with which it is to operate. Each CTM accommodates two full-duplex or two half-duplex communication lines.

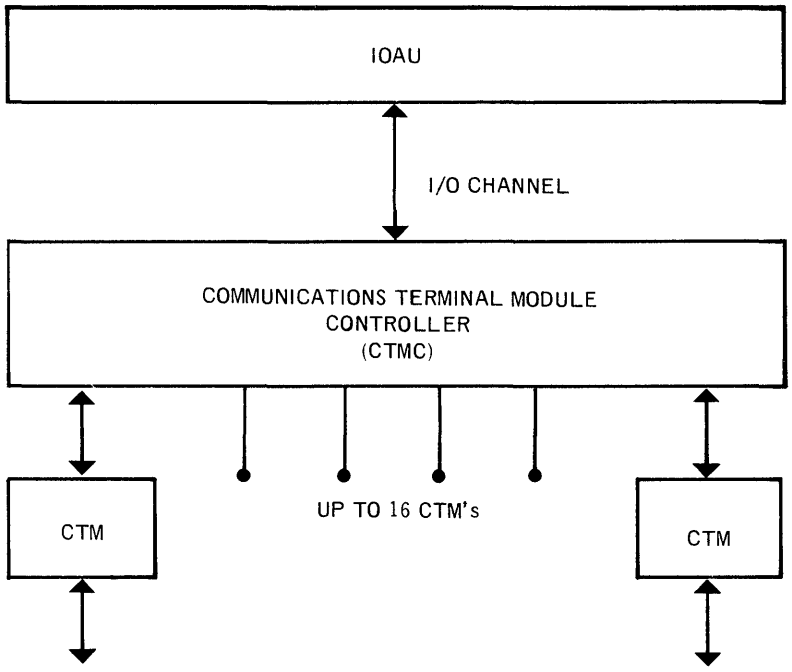


Figure 7–1. CTMC Subsystem, Block Diagram

In addition to the serial modules, there are also input and output parallel modules and a dialing module available. The parallel modules operate at speeds up to 75 8-bit characters per second. The automatic dialing modules enable the processor to establish communication with remote points through the common carrier's switching network.

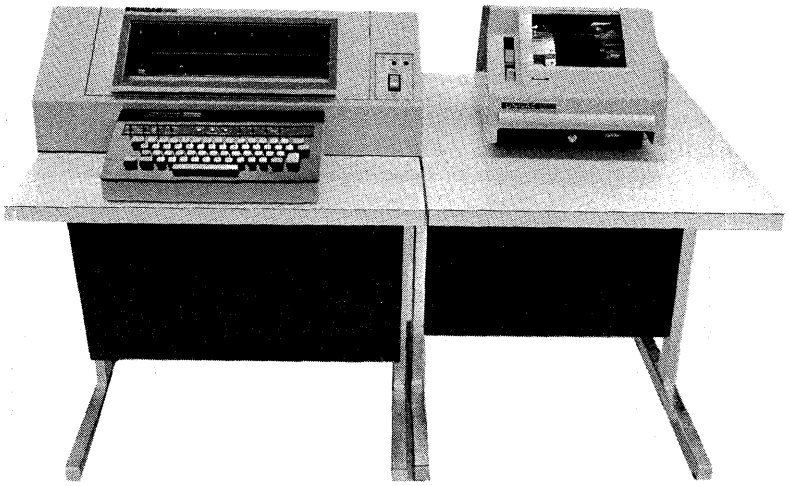
Characteristics of the six types of modules are summarized in the following table:

TYPE	SPEED	MODE	LEVEL
Low	To 300 BPS	Asynchronous bit serial	5, 6, 7, or 8
Medium	To 1800 BPS		
High	To 50,000	Synchronous bit serial	
Dialing	Variable	Bit parallel	4
Parallel out	To 75 CPS	Timing signal bit parallel	8
Parallel in			

NOTES:

BPS = bits per second
CPS = characters per second

7.3.2. UNIVAC Data Communication Terminal (DCT) 500



The UNIVAC Data Communication Terminal (DCT) 500 is a low cost, unbuffered, asynchronous keyboard/printer terminal similar in operation to a teletypewriter, and provides up to 132-column format and five carbons. The DCT 500 can replace existing teletypewriters with little or no changes in the software handlers for point-to-point communications networks over voice-grade telephone toll lines or private lines. In a multiparty polled environment the DCT 500 operates in accordance with ASCII procedures.

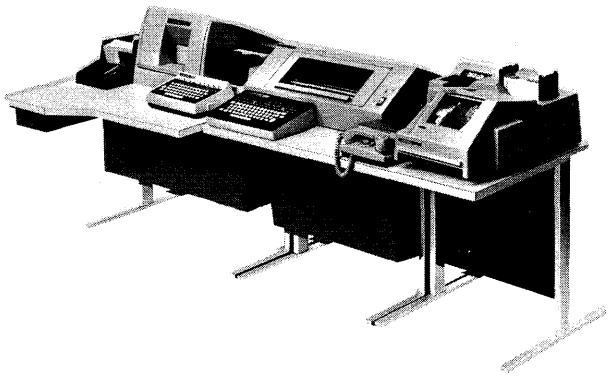
The DCT 500 can operate in a receive-only mode, a keyboard send/receive mode, or an automatic send/receive mode. The basic printer system (mimumum equipment) can be expanded to include a keyboard and a 1-inch paper tape read/punch unit at any time. Additional optional equipment is available to allow for multistation operation.

The optional features include:

- Automatic answering
- Master/slave operation
- Print monitor
- Internal modem
- Paper tape

CHARACTERISTICS	
Transmission code	8-level ASCII
Interfaces	EIA Standard RS-232/CCITT Internal Modem
Transmission mode	Half-duplex or Full-duplex (2- or 4-wire)
Transmission rate	110, 150, or 300 bits per second (selectable)
Printing rate	30 characters per second
Font selections	ASCII, EBCDIC A (Business)/H (Scientific)
Printable characters	63 plus space
Print positions per line	132 (adjustable tractor)
Paper tape reader/punch rate	50 characters per second

7.3.3. UNIVAC Data Communication Terminal (DCT) 1000



The Data Communication Terminal (DCT) 1000 is a fully buffered 30 character per second incremental printer which can be expanded to include a keyboard, card reader, card punch, paper tape reader/punch, and an auxiliary printer. The DCT 1000 transmits data or receives data from a local or remote computer or to a remote DCT 1000 in a conversational or batch mode.

7.3.3.1. Data Buffers

Two 160 character buffers are standard on the DCT 1000. These buffers facilitate the following:

- Automatic Blocking

  This eliminates complicated and time-consuming operator functions and minimizes training.

- Automatic Error Correction

  This eliminates manual correction protection procedures such as reloading cards and retyping input data.

- Error Free Output

  All messages are completely checked for character errors, block errors, duplicate blocks, or lost blocks. The result is that no errors are entered into the output medium.

- High Transmission Speeds

  The full capability of the line can be utilized since the transmission rate can be much higher than the I/O rate. On party line systems, this yields data throughput on a line which is the sum of the throughputs of the individual terminals.

7.3.3.2. Polling System

The DCT 1000 has complete polling and address recognition capabilities which allows the CPU to completely control up to 31 UNIVAC DCT 1000s on a single line. The terminals may be connected in a series string in different geographical locations or at a single point on the UNIVAC Terminal Multiplexer.

7.3.3.3. Operating Mode

The DCT 1000 operates in conversational or batch mode. The mode is under complete control of the central computer. In either mode, terminal operation is half duplex. At any one instant in time, data is either being transmitted or received.

Many DCT 1000 devices can be operated at full speed on a single voice-grade line. This is accomplished by combining the full potential of both polling and buffering.

7.3.3.4. Communications Interface Flexibility

The DCT 1000 can be tailored to complement the transmission facility which best fits the application. The following options are available:

Line Type	– Switched or Private
Private Line	– 2 Wire or 4 Wire
Modulation	– Synchronous or Asynchronous
Transmission Speed	– Asynchronous 300, 1200, or 1800 Baud Synchronous up to 4800 Baud
Interface	– EIA RS-232 (Synchronous or Asynchronous) MIL STD 188B (Synchronous)
Direct Connection	– To CTM
I/O Channel	– Direct to 1100, 400, or 9000 Series I/O channel by a terminal multiplexer

7.3.3.5. UNISCOPE 100 Compatibility

The DCT 1000 transmission control procedures are completely compatible with the UNISCOPE 100 terminal. Therefore, DCT 1000 and UNISCOPE 100 terminal devices can be intermixed on the same transmission line or on the same UNIVAC multiplexer. This mix and match capability yields an almost limitless number of configurations. Control can be achieved at the central site system with a single common handler.

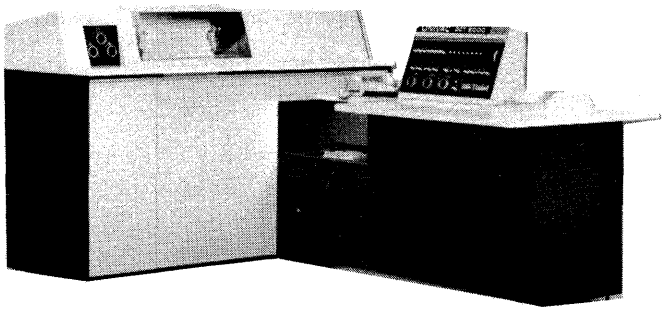
DCT 1000 (printer only) stations can be used to furnish hard copy for the UNISCOPE 100 terminal. The printing operation is not dependent on the display hardware and does not delay any operator functions at the display stations.

7.3.3.6. Offline Capability

When the DCT 1000 is not transmitting or receiving data, it need not be idle. The DCT 1000 can be used offline to generate paper tapes, list cards, or for media conversion. Additionally, while the DCT 1000 is receiving or transmitting data online, the punch can be used offline.

CHARACTERISTICS	
Card reading speed	40 cards per minute
Card punching speed	35 cards per minute
Printing speed	30 characters per second
Printing positions per line	132 (adjustable tractor)
Printable characters	63 plus space
Paper tape speeds	50 characters per second
Buffer storage	320 character capacity in two buffers, 160 characters each
Translator selections	ASCII Code EBCDIC H (Scientific) Code EBCDIC A (Business) Code Binary with additional feature
Transmission Method	Block by block
Transmission Mode	Half duplex; 2 or 4 wire (nonsimultaneous; two-way transmission)
Transmission Facilities	Voice-grade telephone toll exchange or private line
Transmission Rate	Synchronous 300, 1200, or 1800 bits per second; synchronous 4800 bits per second

7.3.4. UNIVAC Data Communication Terminal (DCT) 2000



The UNIVAC Data Communication Terminal (DCT) 2000 is a combination printer and reader/punch designed to transfer large quantities of data efficiently over voice-grade facilities. This terminal (tied into a network with computers such as the UNIVAC 1004 Card Processor or other DCT 2000 systems) can handle up to 250 blocks per minute. The DCT 2000 is also available without the combination card reader/punch for use as a printer terminal.

Ease of operation and the fact that no programming is required at the terminal location make the DCT 2000 simple to install and operate. Normally available ac power is all that is required to operate the unit, and the common carrier can simply make connection to his data communication facilities. Either a private line connection at a maximum rate of 2400 bits per second or a dial facility at a maximum rate of 2000 bits per second can be installed according to the user's requirements since the DCT 2000 and the common carrier equipment both meet the EIA RS-232 standard communications interface for industry.

The UNIVAC DCT 2000 consists of a bar printer, a card reader/punch (not needed when used only as a printer terminal), a control unit, and an operator's console; it is designed for:

- Reliability – through the use of the latest monolithic integrated circuits. Monolithic integrated circuits far exceed the reliability of ordinary transistorized circuits; furthermore, they produce less heat, use less power, and are less affected by fluctuating environmental conditions.

- Expandability – through the use of an input/output channel. The input/output channel permits the use of four additional input or output devices; for example, a paper tape punch might be added.

- Flexibility – through the use of eleven field-installable features (see Table 7–1).

CHARACTERISTICS	
Card reading speed	200 cards per minute
Card punching speed	75–200 cards per minute
Printing speed	250 lines per minute
Printing positions per line	80 or 128
Printable characters	63 plus space
Buffer storage	256 character capacity in two buffers, 128 characters each
Translation capabilities	Hollerith ASCII Hollerith to XS-3 (DLT compatible)
Transmission method	Block by block
Transmission mode	Half duplex; 2 or 4 wire (nonsimultaneous; two-way transmission)
Transmission facilities	Voice-grade telephone toll exchange or private line
Transmission rate	2400 bits per second (private line) 2000 bits per second (switched telephone network)
Transmission code	ASCII XS-3 (DLT compatible)

OPTION NAME	DESCRIPTION
Punch Check and Alternate Stacker	Allows a check of the actual punch die movement and diverts incorrectly punched cards to an error stacker while automatically repunching the data into another card.
128 Print Positions	Allows the basic 80 print-position line to be expanded to 128 print positions.
Unattended Operation	Allows data to be transmitted or received with no operator intervention necessary at the DCT 2000.
Transmit/Receive Monitor	Allows data that is being punched or read to be printed simultaneously.
Offline Listing	Allows data to be printed from cards when the DCT 2000 is not transmitting or receiving.
Peripheral I/O Channel	Allows four additional input or output devices to be attached to the DCT 2000.
Short Block Capability	Allows shorter messages to be handled, thereby increasing the throughput and message efficiency. Punching can increase to a maximum rate of 200 cpm.
Select Character Capability	Allows a transmitting DCT 2000 to select the peripheral in the receiving DCT 2000.
Telephone Alert	Allows voice communications between locations over the data facilities by providing signals through which the operators can make connection.
Error Detection and Retransmission	Allows automatic retransmission of a message when a character or message parity error is detected.
Form Control	Allows multiple line spacing and form feed under control of a special character in a message and a paper tape loop.

Table 7–1. UNIVAC DCT 2000 Field-Installable Options

7.3.5. UNISCOPE 100 Visual Communication Terminal Subsystem

The UNISCOPE 100 Visual Communication Terminal Subsystem is a low-cost, alphanumeric display designed for a broad range of applications which require direct operator interaction with a centralized computer system. Due to its modular construction, the UNISCOPE 100 terminal can operate either as a data entry or as a display device. It can be conveniently located at the central computer site or at a remote station where it is connected to the system by way of telephone lines.

The UNISCOPE 100 terminal is a self-contained unit consisting of a cathode-ray tube display screen, refresh storage, character generator, control logic, operator keyboard, and communication interfaces. Special interfaces for direct computer connection and hard copy output are also available. A variety of presentation formats are offered which provide a total display capacity of 480, 512, 960, or 1024 American National Standard Code for Information Interchange (ASCII) characters. Each of these units is capable of displaying the complete ASCII set of 96 characters which include upper and lower case alphabetics. Hardware editing features enable the operator to completely edit any message prior to transmitting it to the computer.

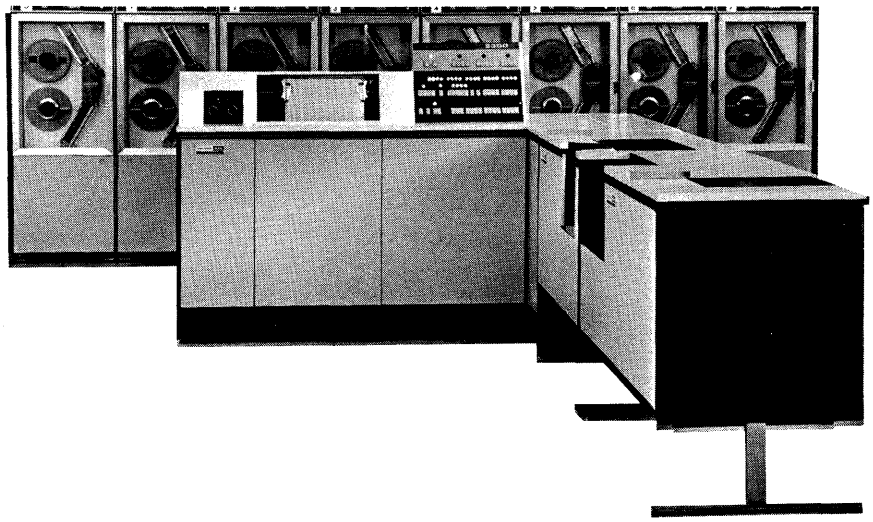
Up to 31 UNISCOPE 100 terminals may be connected to a single communication line or to a computer input/output channel by means of a multiplexer. This general-purpose multiplexer is available with all the communication line interfaces available on the UNISCOPE 100 terminal, thus permitting a mixture of single units and multiple units on one communication system. The multiplexer also provides broadcasting of output messages to multiple devices.

The keyboard has been functionally designed to approximate the conventional electric typewriter with its keyboard appearance, touch pressure, key travel, and slope characteristics. Typewriting speeds in excess of 80 words per minute can be accommodated by the keyboard. Because of its similarity to the standard typewriter, little additional training is required to operate it.

The keyboard includes cursor controls and editing keys, and the layout is right-left assignment balanced to efficiently distribute the work load. The keys are arranged for convenient function discrimination.

CHARACTERISTICS	
Display	480, 512, 960, or 1024 characters (80 per line, 6 lines; 32 per line, 16 lines; 80 per line, 12 lines; 64 per line, 16 lines) on 10-inch wide and 5-inch high screen. Characters are ASCII and also include both upper and lower case alphabetics; split screen
Keyboard	Alphanumeric and symbolic with eight cursor control keys and five editing keys
Storage	Magnetic core, 7.2-microsecond cycle time, 7-bit ASCII code with parity bit
Data transmission	Direct – up to 50,000 characters per second Line – up to 9600 bits per second Half or full duplex Party line polling Nonsignificant space suppression Block transmission Message segmentation
Power	Standard office receptacles

7.3.6. UNIVAC 9200/9300 Onsite/Remote Subsystems



The user of UNIVAC 9200/9300 or 9200 II/9300 II Systems can, without impairing to the slightest degree its capabilities as a self-contained data processor, demand the full and powerful facilities of the 1110 System. This is accomplished by use of the UNIVAC intercomputer control unit (ICCU) to connect the 9200/9300 to the 1110 System and the DCS (Data Communications Subsystem) to link a 9200/9300 System to communication lines. The DCS permits selection of communication speed for the most efficient use of the means of transmission: dial-switched voice line, private line, or broadband line.

CHARACTERISTICS

	9200	9200 II	9300	9300 II
System Orientation	Card/Disc	Card/Tape/Disc	Card/Tape/Disc	Card/Tape/Disc
Basic main storage	8192 bytes	8192 bytes	8192 bytes	16,384 bytes
Maximum main storage	16,384 bytes	32,768 bytes	32,768 bytes	32,768 bytes
Main storage cycle time	1200 nanoseconds per byte	1200 nanoseconds per byte	600 nanoseconds per byte	600 nanoseconds per byte
Add (decimal) instruction time (two 5-digit numbers)	104 microseconds	104 microseconds	52 microseconds	52 microseconds
Multiply, divide, and edit capability	Optional	Optional	Standard	Standard
Card reader Basic reader UNIVAC 1001 Card Controller	 400 cpm 1000/2000 cpm	 400/600 cpm 1000/2000 cpm	 600 cpm 1000/2000 cpm	 600 cpm 1000/2000 cpm
Card punch Column Row	 75–200 cpm Not available Read/punch optional	 75–200 cpm 200 cpm 250 cpm Read/punch optional	 75–200 cpm 200 cpm 250 cpm Read/punch optional	 75–200 cpm 200 cpm 250 cpm Read/punch optional
Print speed	250, 300, or 500 lpm	250, 300, 500, 600, 900, 1100, 1200, or 1600 lpm	600, 900, 1100, 1200, or 1600 lpm	600, 900, 1100, 1200, or 1600 lpm
Paper Tape Read Punch	 300 cps 110 cps	 300 cps 110 cps	 300 cps 110 cps	 300 cps 110 cps
Multiplexer channel transfer rate	85,000 bytes per second	85,000 bytes per second	85,000 bytes per second	85,00 bytes per second
Selector channel transfer rate	Not available	35,000 bytes per second	Not available	35,000 bytes per second
Data communication subsystem	Up to 8 duplex lines	Up to 8 duplex lines	Up to 8 duplex lines	Up to 8 duplex lines
Registers	8 for processor functions 8 for I/O functions	8 for processor functions 8 for I/O functions	8 for processor functions 8 for I/O functions	8 for processor functions 8 for I/O functions

The 9200/9300 equipment has storage large enough and internal speeds fast enough to utilize communication lines to the fullest. Furthermore, the DCS subsystem has provisions for error detection which are flexible enough to be adaptable to any reasonable requirements of the user.

When a 9200/9300 System is being used as a remote subsystem of an 1110 installation, the EXEC allows the remote user to send his program and data over a communication line and receive the complete output later, either at the point of origin or at some other designated location.

The minimum 9200/9300 System configuration consists of a central processor unit, 8K of main storage, and a printer. All other peripherals are options and must be specially ordered.

7.3.6.1. Card Reader

The card reader operates at a rate of 1000 cards per minute on a column by column basis. Read checking features are standard to assure correct input. Information read from the card is transferred to the processor in either image mode or translate mode. Translation is to one of three codes: EBCDIC, ASCII or Compressed Code.

The reader includes a self contained control unit and synchronizer that regulates flow of data and control signals to and from the reader mechanism. This control unit attaches to a standard UNIVAC 9000 Series multiplexer channel. The card reader reads three card sizes (80, 66, and 51 column) and has two output stackers each with a capacity of 2000 cards. The input hopper has a 2400 card capacity.

CHARACTERISTICS	
Card reading speed	1000 cpm
Input hopper capacity	2400 cards
Number of output stackers	2
Output stacker capacity	2000 cards per stacker
Read modes	Image mode – 160 6-bit characters per card. Translate mode – 80 characters per card. Three available codes: 8-bit EBCDIC code 8-bit ASCII code Compressed code
I/O channels	1 shared multiplexer subchannel
Optional features	51- or 66-column short card feeds

7.3.6.2. Upper/Lower Case Printer

The upper/lower case printer is a free-standing drum printer with a control unit and interface for connection to a standard UNIVAC 9000 Series I/O Channel. It contains a 94 character set consisting of upper and lower case alphabetics, numerals and 32 symbols to accommodate virtually every system application.

A 1000 line per minute printing rate is maintained with up to 87 contiguous characters whereas for those occasional applications requiring the entire 94 characters, the throughput is still an effective 840 LPM. A 14 character subset consisting of the ten numerics and four common symbols are repeated at 180 degree intervals on the drum. This innovation provides an impressive 2000 LPM throughput for those predominately numeric applications.

The load code function allows the user to operate with any desired seven or eight bit code.

CHARACTERISTICS	
Printing speed	2000 LPM for 14 character subset 1000 LPM for any 87 contiguous characters 840 LPM for entire 94 character set
Characters per line	132 character print positions
Printable characters	94 characters plus space: 26 upper case alphabetic 26 lower case alphabetic 10 numeric 32 punctuation marks and special symbols
Horizontal spacing	10 characters per inch
Vertical spacing	6 or 8 lines per inch
Form slew rate	33 ips for 6 or 8 lines per inch spacing
Form width	4 to 22 inches
Form length	1 to 22 inches
Number of form copies	Up to six part continuously sprocketed forms
Form advance	Loop control
Line advance	Single, double or triple spacing under program control
Forms advance	Up to 132 lines per command
Speed of form advance	11.5 + 5.05 (N–1) milliseconds at 6 lines per inch 11.5 + 3.8 (N–1) milliseconds at 8 lines per inch N = number of lines spaced

7.3.7. The UNIVAC Data Communication Subsystem (DCS)

The DCS subsystem provides communication capability for the UNIVAC 9000 series computer. Connected to a multiplexer channel, it enables synchronous data transmission at speeds up to 50,000 bits per second between the 9200/9300 computers and the 1110 System over standard communication circuits. The unit is physically small so that two of them can be mounted in space available in the 9200/9300 main frame.

The UNIVAC DCS susbystem is modular, permitting field modifications as demands for various options arise. As communication needs grow, different interfaces can be substituted to upgrade capabilities.

Its many features include the following:

- Automatic Error Checking

  The subsystem checks character and message parity by sending either odd or even parity bits. Longitudinal redundancy can be checked by hardware or by user software.

- Self Testing

  The hardware tests the DCS under program control connecting the output line to the input line to verify transmission and receipt of data.

- Unattended Answering

  The subsystem responds to incoming calls from dialed lines without operator intervention.

- Variable Message Length

  A message may be of any length, from one character up to available storage size.

- Compatibility

  The DCS is compatible with a number of UNIVAC systems including the 1110, 1108, 1106, 494, DCT 2000, and the 1004.

CHARACTERISTICS	
Speed and facilities	Dial voice lines – 2000 bits per second Private voice lines – 2400 bits per second Broadband lines – 50,000 bits per second
Data coding	Five to eight levels, plus parity
Checking	Odd or even message and character parity Longitudinal redundancy check is optional
Multiplexer sub-channel required required	One per subsystem

8. COMMUNICATIONS/SYMBIONT PROCESSOR

8.1. GENERAL

The Communications/Symbiont Processor (see Figure 8–1) is a high performance, internally programmed system which is intended to absorb the combined symbiont functions of communications control and paper peripheral control. Its high speed internal operation and flexible I/O channels provide the expedience necessary to effect high throughput rates and to interface to the virtual world of communications facilities and terminals.

Assuming control of all symbiont operations, the Communications/Symbiont Processor (C/SP) relieves the host computer of storage allocated to symbiont or handler programs and of time associated with numerious interrupt processing, data formatting, data editing, data translation, and other mundane tasks.

System throughput is increased and user turn-around time is decreased by virtue of the improved system performance offered by isolated, dedicated processor elements.

The concept of front-end processing offers efficiency by simplifying the interface between the host system and its peripheral subsystems. All paper peripherals and communication lines appear as a common intelligent subsystem to the host. The C/SP effectively buffers the host from the infinite variety of remote terminal and communications line transmission disciplines. It effectively accommodates all popular communication devices currently on the market.

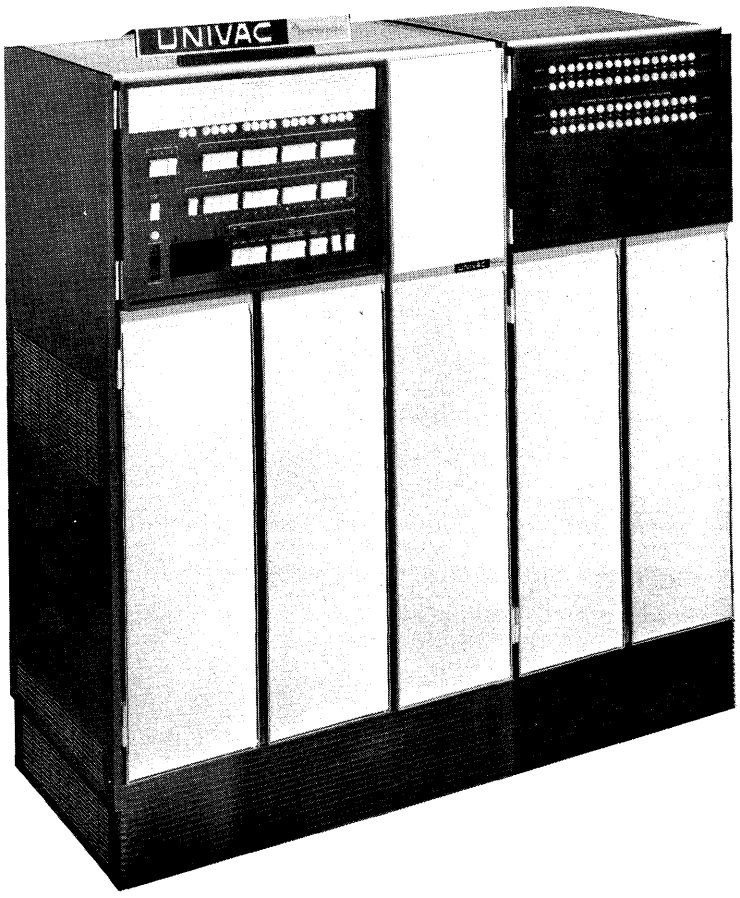


Figure 8–1. Communications/Symbiont Processor (C/SP)

8.2. HARDWARE

The C/SP hardware was designed with modularity and flexibility as primary goals. Realizing the wide implications of designing a multifunction subsystem, special emphasis was placed on high volume throughput. Special channels were designed to accommodate, with a high degree of efficiency, the varying needs of prime peripherals and of communication terminals. The basic configuration (see Figure 8–2) includes the following:

(1) Processor Cabinet

- Processor
- 16 General Purpose Registers
- Maintenance Panel
- Power Failure Interrupt Feature
- Interval Timer
- Special Device Channel
- 1100 Series Adapter Channel

(2) Storage Cabinet (one or two)

- 32K – 65K bytes storage
- Storage Protection Feature

(3) Optional Features include:

- Multiplexer Channel
- General Purpose Communications Channel

  (up to 32 full duplex or up to 64 half duplex Communications Line Terminals (CLT's); combinations permitted)
- Dialing Adapters (uses one CLT position)
- Asynchronous Timing Assemblies

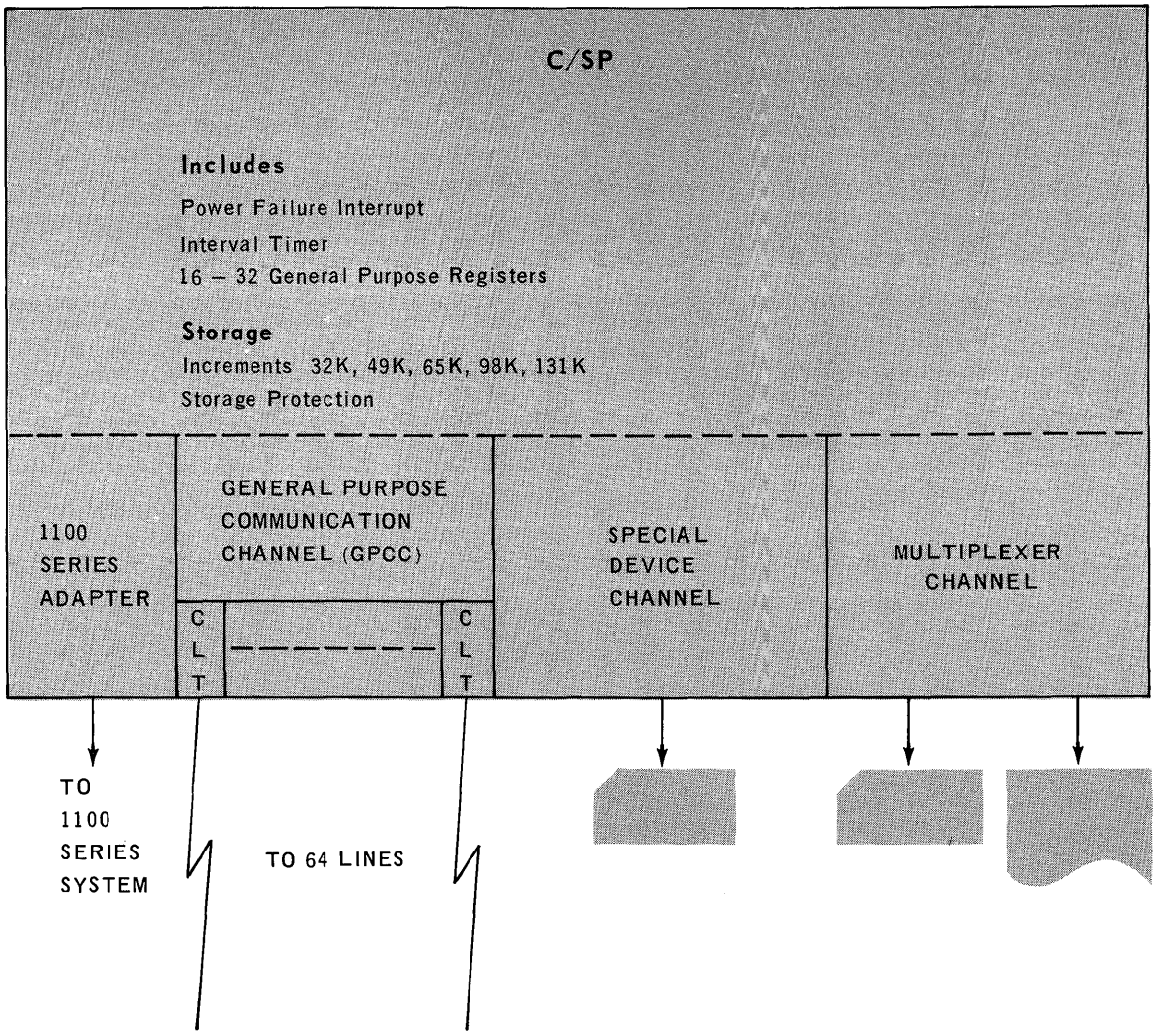


Figure 8–2. Communications/Symbiont Processor (C/SP) Configurator

8.2.1. Processor

The processor portion of the C/SP provides the flexibility that is required to control the I/O data flow and to perform message processing, as necessary, in an online peripheral or a communications environment.

8.2.1.1. Processor Characteristics

Major features of the processor include the following:

- 52 half-word and full-word instructions;
- sixteen 32-bit general purpose registers, external to storage;
- attached processor maintenance panel;
- I/O interrupt and data priority controls;
- variable interval timer;
- halfword basic data path;
- multilevel interrupt;
- 630 nanosecond cycle time;
- basic binary add (RX) instruction time of 2.52 μs (four cycles);
- binary add instruction (RR) time of 1.26 μs (two cycles).

8.2.1.2. Control Section

The control section regulates the sequence in which instructions are executed, interprets and controls the execution of each individual instruction, initiates cycling of main storage, performs required storage address modification and indexing, and determines the different processor modes of operation. All of the hardware aspects of interrupt handling, error checking, and protection are also performed by the control section.

8.2.1.3. Arithmetic Section

The arithmetic section of the processor performs all data manipulations including logical and numerical arithmetic, data comparisons, and shifting. The arithmetic section also performs single or double indexing of operand addresses. Arithmetic operations are performed in the twos complement form. A fixed-point arithmetic operand can be either a 32-bit full-word or a 16-bit half-word. The sign of a fixed-point operand is always the leftmost bit of the operand. When accessed from storage, a half-word fixed-point number is always expanded to a right-justified full-word; the sign is extended to the left.

Logical operations on fixed-length operands are performed in registers. Logical operations include comparing, translating, editing, bit setting, and bit testing.

8.2.1.4. Instruction Set

The C/SP utilizes 52 basic instructions that vary in format and length. The format, in general, is dictated by the operation to be performed and the location of the operands. Operands may be located in storage, in general purpose registers, or in the instruction itself. The length of an instruction is dictated by the format and is either a half-word or a full-word. All processor instructions must be on half-word boundaries in storage. Operand addresses in storage are on byte, half-word or full-word boundaries, depending upon the instruction. For example, if an operand for a particular instruction is a full-word, the operand address in storage must be on a full-word boundary.

Illustration of the four basic instruction formats that are used in the C/SP are provided in Figure 8–3 and are designated as follows:

RR (Register to Register) Instructions

RX (Register to Indexed Storage) Instructions

RS (Register to Storage) Instructions

SI (Storage and Immediate Operand) Instructions

Each format consists of an operation code (OP Code) and two or more fields which specify, among other things, the addresses of operands (in storage or in the general purpose registers). Each field is identified by a letter followed by a subscript numeral. The numeral, in general, denotes the operand (1, 2, or 3) to which the field applies.

The operation codes are expressed in hexadecimal (base 16); each code appears as two hexadecimal digits in the 8-bit OP Code field of each instruction.

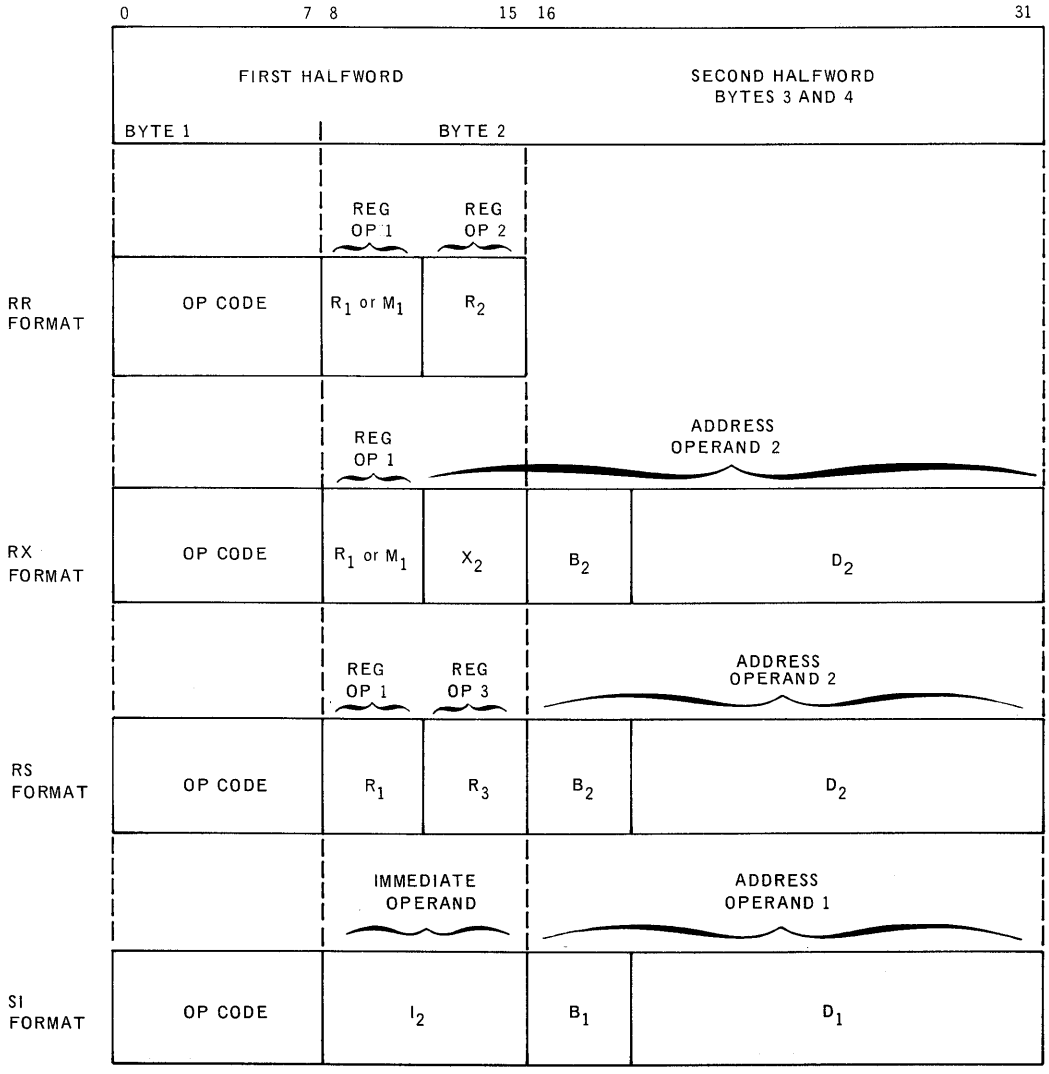


Figure 8–3. Basic C/SP Instruction Formats (Part 1 of 3)

FUNCTIONAL DESCRIPTION	MNEMONIC	OP CODE	FORMAT*	EXECUTION TIME (CYCLES)**
Add	A	5A	RX	4
Add half-word	AH	4A	RX	4
Add	AR	1A	RR	2
Compare	C	59	RX	4
Compare half-word	CH	49	RX	4
Compare	CR	19	RR	2
Divide half-word	DH	53	RX	22
Load	L	58	RX	4
Load half-word	LH	48	RX	4
Load	LR	18	RR	2
Multiply half-word	MH	52	RX	20
Shift left single	SLA	8B	RS	3 + N1 (Note 2)
Shift right single	SRA	8A	RS	3 + N1 (Note 2)
Subtract	S	5B	RX	4
Subtract half-word	SH	4B	RX	4
Subtract	SR	1B	RR	2
Store	ST	50	RX	5
Store half-word	STH	40	RX	4
LOGICAL				
AND	N	54	RX	4
AND	NI	94	SI	5 (Note 1)
AND	NR	14	RR	2
Compare logical	CL	55	RX	4
Compare logical	CLI	95	SI	5 (Note 1)
Compare logical	CLR	15	RR	2
Divide Polynomial	DP	81	RS	3 + N (Note 2)
Exclusive OR	X	57	RX	4
Exclusive OR	XI	97	SI	5 (Note 1)
Exclusive OR	XR	17	RR	2
Insert Character	IC	43	RX	3
Load Address	LA	41	RX	4
Move	MVI	92	SI	5 (Note 1)
OR	O	56	RX	4
OR	OI	96	SI	5 (Note 1)
OR	OR	16	RR	2
Shift left single logical	SLL	89	RS	3 + N1 (Note 2)
Shift right single logical	SRL	88	RS	3 + N1 (Note 2)
Store character	STC	42	RX	4
Test under mask	TM	91	SI	5 (Note 1)

Figure 8-3. Basic C/SP Instruction Formats (Part 2 of 3)

C/SP INSTRUCTION REPERTOIRE

FUNCTIONAL DESCRIPTION	MNEMONIC	OP CODE	FORMAT*	EXECUTION TIME (CYCLES)**
Branch & link	BAL	45	RX	4
Branch & link	BALR	65	RR	3
Branch on condition	BC	47	RX	2
Branch on condition	BCR	07	RR	1/2 NB/B
Branch on count	BCT	46	RX	4
Branch on count	BCTR	06	RR	2/3 NB/B
Branch on index high	BXH	86	RS	6
Branch on index low or equal	BXLE	87	RS	6
STATUS SWITCHING				
Halt & proceed	HPR	99	SI	***3 (If No Halt)
Load PSW	LPSW	82	SI	***5 (Note 1)
Set storage key	SSK	08	RR	***3
Set system mask	SSM	80	SI	***4 (Note 1)
Supervisor call	SVC	0A	RR	8
INPUT/OUTPUT				
Start I/O	SIO	9C	SI	***6–24

*RR – Register to Register
RX – Register to Indexed Storage
RS – Register to Storage
SI – Storage and Immediate Operand
**1 memory cycle = 630 nanoseconds; all RX instructions add one cycle if double indexing.
***Denotes Privileged Instruction

Note 1 – Number of cycles is one less if not indexing.

Note 2 – N = No. shifts up to 16, N_1 = No. shifts up to 31 (Module 16).

Figure 8–3. Basic C/SP Instruction Formats (Part 3 of 3)

8.2.1.5. Interval Timer

The interval timer, which utilizes a fixed-word location in storage, is a feature in the processor which provides interval timing and time of day information. Interval timer requests for service are made every six milliseconds. Interval timer requests may be serviced only at the end of a processor instruction execution prior to the processor staticizing the next instruction.

8.2.1.6. Interrupts

The interrupt system provides an automatic means of alerting the C/SP processor to exceptional or unexpected conditions, such as the end of I/O operations, program errors, machine errors, and similar occurrences and directs the processor to the appropriate program routine following their detection. The system permits the interruption of any task to process a task of higher priority. Among the features of the interrupt function are:

- automatic tabling of communications channel interrupts;
- a dynamically alterable priority structure;
- automatic dispatch by interrupt class;
- automatic program switching by interrupt class.

8.2.2. STORAGE

A high performance, plated-wire storage is an integral part of the C/SP. Storage may be located either within the processor cabinet, or in "stand-alone" cabinets, or both, depending upon the size of the storage capacity.

8.2.2.1. Main Storage Characteristics

Major features of main storage include the following:

- Capacity – 32,768 bytes minimum; 131,072 bytes maximum
- Cycle Time – 630 nanoseconds read/write cycle
- Operating Mode – Nondestructive readout
- Storage Data Path – 18 bits wide (2 8-bit bytes and 2 parity bits)
- Addressing – Zero time indexed base and displacement
  – Double indexed
- Storage Protection – Program and I/O Transfer
- Parity – Odd Parity (1 parity bit per byte)

8.2.2.2. Addressing

The addressing hardware accommodates a 17-bit address field which permits one-cycle addressing of 131,072 bytes. While the address field permits the addressing of each byte, the least significant bit of the address is not used to access the data from storage.

On a read cycle, the storage presents two bytes to the processor. If the particular reference requires byte addressing, the processor selects the appropriate byte based upon the least significant bit of the address field.

The capability for partial write is provided; that is, one byte may be written without altering the other byte in the storage half-word.

8.2.2.3. Storage Protection

In addition to the fixed storage assignment, there may be several programs resident in C/SP main storage at any one time. It becomes necessary to restrict storage accesses by a program to the storage limits assigned to it.

Associated with main storage are a maximum of 64 3-bit registers called key storage registers. The storage, beginning at address 0, is divided into a maximum of 64 blocks, each of which contains 2048 bytes. To each of these blocks is assigned a key storage register. The six most significant bits of a storage address are used to define the address of the key storage register associated with the block containing the storage address.

Storage is segmented by grouping together all blocks whose associated key storage registers have the same setting. Since there are three bit positions in a key storage register, a maximum of eight storage segments can be defined.

When a program is loaded, the program is assigned a unique program number. This number is then loaded into the key storage register that is associated with each 2048 byte block assigned to the program.

Storage protection against improper storage accessing is provided for during instruction execution and I/O transfers.

- Instruction Execution Protection

  When a program is scheduled for execution, the program number is loaded into a Program Status Word (PSW) register that is uniquely identified with the program currently in operation. On each access to storage during processing, the program number in this register is compared with the contents of the key storage register that is associated with the storage address. If a match is made, the storage access is allowed, otherwise an error interrupt occurs.

- I/O Transfer Protection

  The number of the program requesting the I/O transfer is presented to the I/O channel. Upon transfer, the contents of the key storage register that is associated with the address to which the transfer is to be made is checked against the program number associated with the I/O channel. Again, a match of the key storage register and program number permits the transfer to take place; otherwise, an error interrupt occurs.

8.2.2.4. Parity

The parity bit associated with each byte provides odd parity for that byte. Parity generation and checking are performed in storage. The parity bits are, however, presented to the processor or an appropriate channel on a read cycle.

8.2.3. Channels

All information transmission in and out of the C/SP is handled by channels. A channel controls the operation of input/output devices and the transfer of data between devices and storage.

Among the outstanding features of the C/SP channels are:

- direct interface to storage;
- independent operation;
- simultaneous operation;
- priority interchangeability.

The C/SP may contain up to seven channels, numbers 0 to 6. Priority of these channels increases in descending channel number order, channel 0 having the highest priority.

8.2.3.1. Channel Types

The C/SP is equipped with the following four types of channels.

- Special Device Channel (SDC) – The primary function of the SDC is to provide the means for local program loading and maintenance of the C/SP using a serial 80 column, 80 card per minute, card reader device.
- 1100 Series Adapter Channel – The 1100 Series Adapter Channel (inter-computer adapter channel) provides an interface for direction communication of the C/SP to an I/O Channel of a UNIVAC 1100 Series computer. The maximum transfer rate is in excess of 300,000 words (36 bits each) per second.
- Multiplexer Channel – This channel provides the capability of attaching all currently available UNIVAC 9000 Series peripheral devices, which operate on a corresponding channel to the C/SP. In addition, the high speed card reader and the ASCII printer can be connected via this channel.
- General Purpose Communications Channel (GPCC) – The GPCC and associated components are described in 8.2.3.2.

8.2.3.2. General Purpose Communications Channel (GPCC)

The GPCC performs such functions as multiplexing the various communications line terminals (CLT) so that one CLT may be serviced at a time, recognizing special characters and sequences of characters, checking character parity, coordinating all data transfers to and from storage, and executing other necessary operations.

The CLT's perform the function of assembly and disassembly of data characters for proper reception from and transmission to a communication line; detection of certain conditions of the communication line such as loss of carrier, a ringing indication, and others; and establishment of character synchronization.

The CLT's handle a wide range of communication with rates up to 50 kilobytes per second. However, the CLT's must be selected so that the total combined rate of service requests (one per byte) is no more than 50,000 per second. When the GPCC is operated at this maximum rate, somewhat less than 40 percent of all available storage cycles are utilized for this purpose.

The GPCC is the link between storage and the CLT's and provides the data path and control for CLT's as they communicate with storage. The single data path can be time-shared by as many as 64 positions, which need not have identical CLT's. A full duplex CLT uses two positions; a half duplex or simplex CLT utilizes only one position.

The GPCC is equipped to analyze each data character and/or sequence of characters which is transmitted through the GPCC and to act upon these characters in a manner that is a program-changeable function of the line to which the GPCC is connected. The GPCC also interfaces with the C/SP processor to service Start I/O (SIO) instructions and interrupts. Associated with the GPCC is a display panel which contains two active line indicators (output and input) for each CLT. The indicator is on when the corresponding CLT data line is in a spacing condition.

The multiplexer portion of the GPCC accepts up to 64 simultaneously presented service requests from the CLT's plus an external function (XF) request from another portion of the GPCC. The multiplexer selects one request by connecting the selected CLT to the GPCC. When all necessary information has been interchanged, the multiplexer is cleared and can immediately accept another request. The multiplexer can accommodate a maximum of 32 full duplex CLT's or 64 half duplex CLT's. An area in storage called a Buffer Control Word (BCW) is associated with each position on the multiplexer and is used to store status, control, and data address information for the particular position.

When information is to be transferred during an I/O operation, the CLT requests service from the GPCC and, when priority permits, sends the CLT address to the GPCC. The GPCC uses the address to select the associated BCW which is loaded into the channel and now controls the channel operation until the information is transferred. Then the BCW, in general changed by the channel operation (for example, address incrementation), is returned to main storage and the GPCC facilities are released. The BCW's then remember the current state of the various positions.

The C/SP is intended to operate in an environment which can involve many different line discipline procedures. Many such procedures have been long established and must be handled unchanged by the C/SP. To avoid a multiplicity of tailored CLT's operating through the GPCC, the BCW is permitted to access a Message Discipline Word (MDW). A chain of MDW's can be considered as a description of a given procedure, and the currently active MDW represents the position reached by a given line within that procedure. The chain of MDW's should not be modified once it has been loaded. Hence, all lines with the same line discipline procedure may share a common chain of MDW's. An MDW contains, for example, parameters which control character parity checking, special character recognition (single or multiple), special character insertion, and other operations.

At various points throughout a procedure, it is necessary to present certain information to the controlling program. Since this information is dynamically changing, it must be stored at the desired point immediately. A normal interrupt is not sufficient since the processor is not always in a condition to accept an interrupt request. Four interrupt lists are provided in storage where Communication Interrupt Words (CIW's) are stored. These lists are controlled by CIW list controls which are in fixed storage locations. A list is selected by a BCW or MDW and is usually assigned on a priority basis.

8.3. PROGRAMMED SYSTEMS SUPPORT

The software support provided for the C/SP is designed to provide complete flexibility for implementing a symbiont subsystem and for handling communications configurations with all types of terminal hardware while maintaining an expedient user interface. Coding efficiency is achieved by the utilization of a powerful instruction set at the assembly level. System macros are also provided to facilitate the user's requirements.

Software to integrate the C/SP effectively with the host computer system is supplied; an assembler and simulator to write and debug user own code on the larger system also are included.

The software package is divided into three segments:

(1) resident programs and routines;

(2) programs to operate under the host computer executive system;

(3) modification to host computer elements.

Each of these segments is discussed in further detail in the following paragraphs.

8.3.1. Resident Programs

The resident program software elements are defined as programs which reside entirely or partially in C/SP main storage during their execution. Resident programs include:

- Operating System
- Diagnostic Routines
- Intercomputer Handler

8.3.1.1. Operating System

The C/SP operating system comprises various program modules which are specified by the user at system generation. When supplied elements are used, the following are included:

- Terminal Management Supervisor (TMS)
- Message Control Program (MCP)
- Terminal Management Control Routine (TMCR)
- Communication Control Routines (CCR)

System interface is provided for inclusion of user versions of, or additions to, any element specified under the operating system. Figure 8–4 illustrates the functional relationship of these elements.

(1) Terminal Management Supervisor (TMS)

The TMS is the nucleus of the C/SP operating system and controls all program switching, I/O queuing, and interrupt handling. The modular design of the TMS is based upon the functional divisions within the scope of the TMS activity; this modularity permits the system to be custom-made at systems generation time for a particular installation.

(2) Message Control Program (MCP)

The MCP uses facilities of the TMS, CCR's, and host handler to accomplish message transmission between on site peripherals, remote sites and the host computer.

The MCP is responsible for remote initialization, accomplishing line connection on dial-in and initiation of dial-out, remote sign-on, message routing, message editing, message translation, function dispatching, initiation of polling, and initiation of communication hardware diagnostic programs. Dynamic buffering that is performed to support I/O operations with the remote site or with the host computer is accomplished by the MCP.

The system MCP is designed to support batch, real time, and demand functions. In addition, certain user own-code options are available to enable a user to implement various preprocessing capabilities within the framework of the MCP. The user also has the capability to implement user coded MCP's that operate concurrently with the system MCP as a separate problem program.

The MCP is loaded at system initialization time following the system boot from the host computer. The MCP operates in a supervisory state, but under a different protect key from the supervisor, and as the highest priority problem program of the system.

(3) Terminal Management Control Routine (TMCR) and Communication Control Routines (CCR)

The routines or programs for controlling the communication terminals on the C/SP are divided into a general handler TMCR and the individual CCR's.

TMCR is a series of routines which perform functions that are common to all CCR's. Some of the functions performed by the TMCR are dial and test mode, CIW preprocessing, and interfacing with the MCP and TMS.

CCR's are designed to compensate for the peculiarities of a given communication service or procedure. Where sufficient commonality exists, a CCR is reentrant and may support more than one type of terminal.

In the normal operation of the communication devices attached to the C/SP, any MCP executes I/O packets in the form of supervisor calls (SVC instructions pointing to a packet). The C/SP TMS passes control to the TMCR to determine which CCR the packet refers to. The appropriate CCR executes the indicated I/O functions desired and passes back appropriate status information.

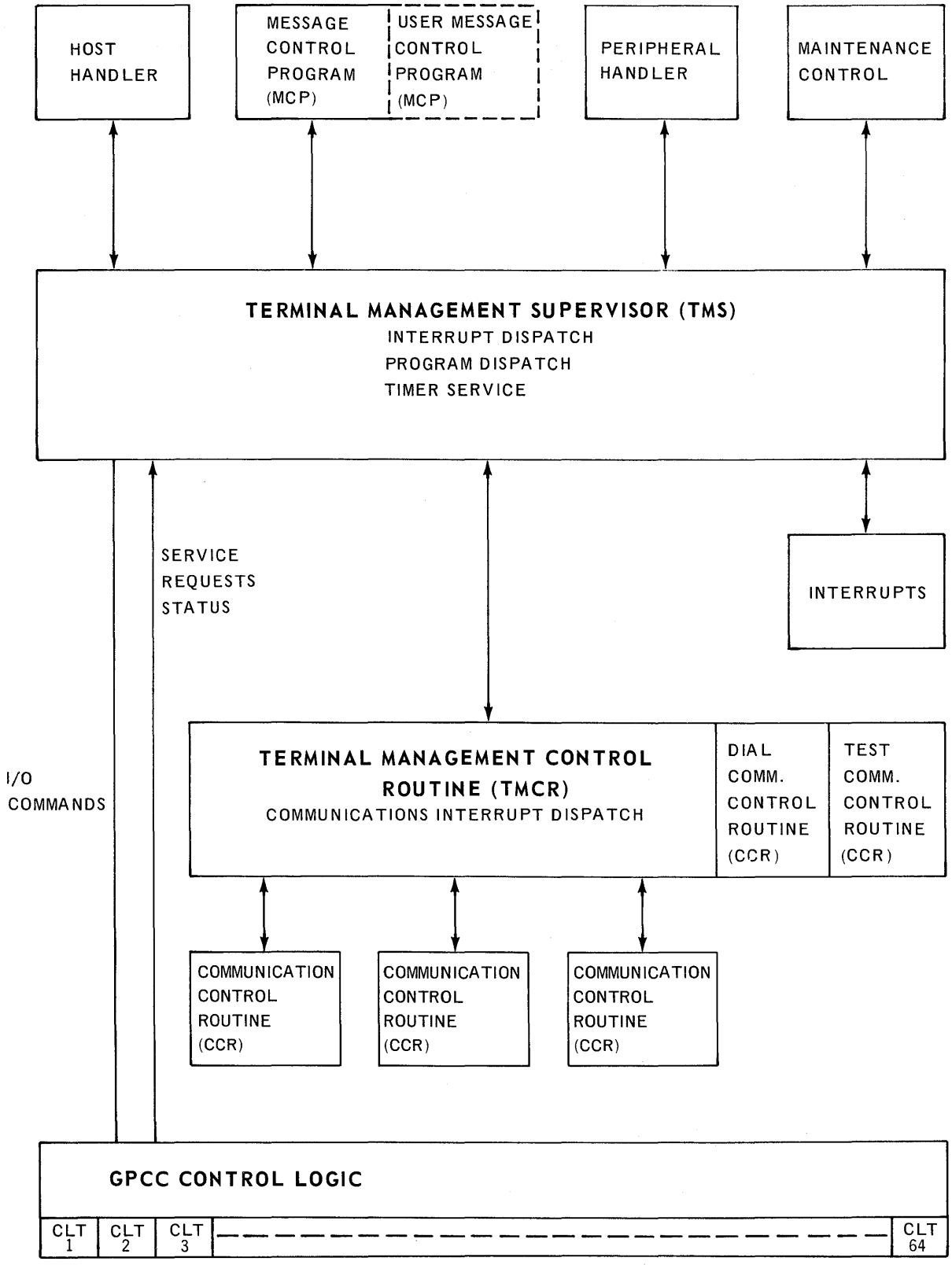


Figure 8–4. C/SP Operating System

Since many of the operations performed are common to all CCR's, the routines necessary for their execution are contained in TMCR.

The TMCR maintains the CIW list and receives control from the TMS whenever an interrupt occurs on the GPCC. After initial processing of the interrupt, control is passed to the appropriate CCR. A Master Status Block (MSB) is maintained for each type of device, service, or procedure which is established at the time of system generation. TMCR and CCR's update information in this block.

The CCR's establish and control the Message Discipline Words (MDW), Buffer Control Words (BCW), and the Line Status Blocks (LSB). Actual hardware I/O control is performed by the CCR's and status information is returned to the MCP which issued the packet.

The following communication devices, services, and procedures are supported with standard CCR's:

- UNISCOPE 100 Display Terminal
- Teletypewriter/UNIVAC Data Communication Terminal DCT 500
- UNIVAC Data Communication Terminal DCT 1000
- UNIVAC Data Communication Terminal DCT 2000
- UNIVAC 1004/9000 Systems
- Binary Synchronous Communication Devices

8.3.1.2. Diagnostic Routines

Diagnostic routines are included to function with the C/SP operating system and provide an indication of the cause of a malfunction through the host computer primary output stream.

8.3.1.3. Intercomputer Adapter Handler (ICA)

The ICA handler is resident within the C/SP and controls all commands between the C/SP and the host computer.

The ICA handler is the same for the remote C/SP as for an on-site C/SP, with the site identification (ID) in the data as the first two words. The first two words also contain the necessary option indicators. Error status is communicated through data transmissions.

8.3.2. Programs Under Host Computer

The following programs operate under control of the host computer operating system and are described in this section:

- C/SP Assembler
- C/SP Element Collector
- C/SP Simulator
- C/SP Service Routines
- C/SP Symbionts

8.3.2.1. C/SP Assembler

The C/SP assembler is a one-phase, two-pass bootstrap assembler. The assembler is an efficient, easy-to-use software that satisfactorily handles most of the programming problems encountered by the user. Each machine instruction and data form has simple, convenient representations in assembly language. The assembler translates this language into a form that can be executed by the computer. The rules that govern the use of the language are uncomplicated and can be easily applied by the programmer.

8.3.2.2. C/SP Element Collector

The element collector provides a means of collecting independent relative binary elements to produce an absolute program for execution in the C/SP. A relative binary element is an output of the assembler, as a result of translating C/SP assembler instructions. An absolute program is a program unit with no unresolved references which can be relocated in C/SP storage as an executable program.

8.3.2.3. C/SP Simulator

The C/SP Simulator is a UNIVAC 1100 Series assembler language user program which runs under the operating system. The simulator accepts C/SP object code, simulates execution on the C/SP processor, and provides diagnostic printout to aid in debugging the C/SP program.

8.3.2.4. C/SP Symbionts

Several symbionts are added to the host operating system to support existing batch/demand processing and to accommodate additional capabilities which are associated with the C/SP system. The main difference between the C/SP symbionts and other symbionts is that the C/SP symbionts are not device oriented and all I/O looks the same to the C/SP symbiont.

The batch/demand input symbiont links to the executive as is currently done by all input symbionts. The output symbiont links directly, bypassing the stacking of output. The C/SP stacks output requests under its own stacking scheme; this is necessary because the C/SP controls the output terminals.

The host computer communication symbiont and host computer storage access symbiont link directly to the I/O symbionts and are controlled by the I/O symbionts. These symbionts are necessary for communications between the host computer executive system, console typewriter, and storage devices.

The C/SP remote symbiont communicates with a remote C/SP on a CTMC and transfers the data to and from the regular I/O symbionts and to the real-time programs.

8.3.3. Modified Host Computer Elements

The C/SP software package includes routines and elements of the host computer operating system which are modified for operation with the C/SP as well as additions to the operating system. These elements include the following:

- C/SP Handler
- Executive Return Additions
- Parameter Table Additions
- Interrupt Answering Routine
- Interrupt Processing Routine

9. PROGRAMMED SYSTEMS SUPPORT

9.1. UNIVAC 1110 OPERATING SYSTEM

The UNIVAC 1110 operating system has been designed to meet the total computing requirements of today's advanced users, and to allow the change and growth required to meet tomorrow's challenge. This operating system is the outgrowth of UNIVAC's many years of experience in multiprogramming, multiprocessing, communications, and real time oriented systems, and provides a system that contains the facilities required in complex environments, yet is easy to operate and use.

The UNIVAC 1110 Operating System, while even more comprehensive than the highly successful EXEC 8 based 1106/1108 Operating System, retains full compatibility with EXEC 8, thereby allowing an 1106/1108 user to upgrade to a UNIVAC 1110 System without the conversion effort required in the past when changing systems.

A complete set of software, ranging from high level language compilers to basic mathematical functions, is included in the UNIVAC 1110 Operating System. The six major categories are:

- Executive System
- System Processors
- Language Processors
- Utility Processors
- Libraries
- Applications

9.2. THE EXECUTIVE SYSTEM

To take full advantage of the speed and hardware capabilities of the UNIVAC 1110 System and to make effective use of a given hardware configuration, a comprehensive internal operating environment has been created.

This environment permits the concurrent operation of many programs; it allows the system to react immediately to the inquiries, requests, and demands of many different users at local and remote stations; it accords with the stringent demands of real time applications; it can store, file, retrieve, and protect large blocks of data; and it makes the best use of all available hardware facilities, while minimizing job turnaround time.

Only through central control of all activities of the UNIVAC 1110 System can this environment of the combined hardware and software systems be fully established and maintained to satisfy the requirements of all applications. The responsibility for efficient, flexible, centralized control is borne by the executive system, which controls and coordinates the functions of the complex internal environment. By presenting a relatively simple interface to the programmer it allows him to use the UNIVAC 1110 System easily, while relieving him of concern for the internal interaction between his program and other co-existent programs.

9.2.1. Multiple Modes of Operation

The technical capabilities of the UNIVAC 1110 Executive System cover a great variety of data processing activities. Its design is such that no penalties are imposed upon any one of these activities by the support provided for the others, and an installation not interested in making use of the full range of activities may specify capabilities to be eliminated at system generation time.

9.2.1.1. Batch Processing

Foremost among these capabilities is the support provided for batch processing. The system is designed to ease run preparation and submission, to shorten job turn-around time, and to reduce the need for operator intervention and decisions.

9.2.1.2. Demand Processing (Time-Sharing)

Complementing the batch processing capabilities of the UNIVAC 1110 Executive System are its time-sharing capabilities, the simultaneous accommodation by the executive system of requests and demands from users at numerous remote inquiry terminals, operating in a demand (or conversational) mode. All facilities available to the batch processing user are also available in the demand mode, the primary difference being that the executive system permits the user additional flexibility in the statement and control of individual runs. The demand user may communicate directly with either the executive or a user program or he may communicate with a conversational processor, such as Conversational FORTRAN or BASIC.

9.2.1.3. Real Time

The executive system is also designed to be applicable to programs which have real time requirements. The UNIVAC 1110 Communications Subsystem, together with efficient scheduling and interrupt processing features of the executive system, provide an environment satisfactory for any real time program.

9.2.1.4. Multiprogramming and Multiprocessing

Runs may come from many sources, remote and central. These various runs, through the executive system's use and control of efficient multiprogramming and multiprocessing techniques may, at any given moment, be in different stages of activity; input, processing, and output may all be occurring simultaneously, thus ensuring efficient operation.

9.2.2. Techniques for Utilization of Mass Storage

The executive system is designed to ensure effective and efficient utilization of the mass storage devices. The consequence is an unprecendented ability to relieve operators and programmers of the responsibility of maintaining and handling cards and magnetic tapes, thus eliminating many of the errors which heretofore have accompanied the use of large scale software systems. At the same time, the overall operating efficiency is considerably improved.

Permanent data files and program files are maintained on the mass storage devices, with full facilities for modification and manipulation of these files. Security measures are established by the executive system to ensure that files are not subject to unauthorized use. Provisions are also made within the executive system for automatic relocation of infrequently used files to magnetic tape, as unused mass storage space approaches exhaustion. When the use of files relocated in such a manner is requested, they are retrieved and restored, under control of the executive system, with no inconvenience to the user.

9.2.3. The Primary Functional Areas of the Executive System

The UNIVAC 1110 Executive System is composed of many different routines which perform many different functions. These functions and routines are summarized in the following paragraphs.

9.2.3.1. Executive Control Language

In the executive system the user has a simple means of directing the execution of the individual activities of a run and of relaying operational information concerning the run to the executive. This is accomplished through a set of control commands, capable of performing all of the functions desirable or necessary in a modern executive system. The command language is open ended and easily expanded, so that features and functions may be added as the need arises.

The basic format of an executive control statement is quite simple, and is adaptable to a large number of input devices. Statements are not restricted to card image format, and may be of variable lengths. Each statement consists of a heading character (@), for recognition purposes, followed by a command and a variable number of expressions. The end of a statement is indicated by the end of a card, a carriage return, or an equivalent signal, depending on the type of input device.

9.2.3.2. The Supervisor

The supervisor is the UNIVAC 1110 Executive System component that controls the sequencing, setup, and execution of all runs. It is designed to control the execution of a large number of programs without any interaction among them.

The supervisor contains three levels of scheduling: coarse scheduling, dynamic allocation of storage space, and CAU dispatching. Runs entering the UNIVAC 1110 System are sorted into information files and these files are used by the supervisor for run scheduling and processing. Control statements for each run are retrieved and scanned by the control command interpreter in the supervisor to facilitate the selection of runs for setup by the coarse scheduler. The coarse scheduling of each run primarily depends on two factors: the priority of the run, and its facility requirements.

The dynamic allocator takes runs set up by the coarse scheduler and allots storage space according to the needs of the individual tasks of a run. Each run may be thought of as being made up of tasks, where a task is a single operation of a system processor or the execution of a user program. All tasks for a given run are processed serially but not necessarily consecutively; if there are several runs, the tasks of separate runs are interleaved.

When time-sharing of storage is appropriate, the dynamic allocator initiates storage swaps. This involves writing one program on drum storage and replacing it temporarily in storage with another program. Such action is taken only to provide reasonable response time to remote demand-processing terminals.

The CAU Dispatching routine is a third level of scheduling; it selects among the various tasks currently occupying storage whenever it is appropriate to switch the commitment of one of the CAU's from one task to another. Under normal circumstances, a batch program is allowed to use a CAU either until it becomes interlocked against some event or until some higher priority program is freed of all of its interlocks.

9.2.3.3. Time Slicing

In order to accommodate demand processing, periodic time slices must be assigned to particular routines. This is done by the timing routine, which interlocks the currently running program and, at specified intervals, examines a separate queue of periodically scheduled routines. Based on the required duty cycle of the demand routines, their priorities, and the priorities of other ready routines, the demand routine is moved to the ready queue with an adjusted priority. In this manner, even low-priority demand routines are given at least occasional use of a CAU.

9.2.3.4. Storage Compacting

Certain hardware features make feasible the dynamic relocation of programs residing in main and/or extended storage – a necessity for effective multiprogramming. At program termination, the assigned storage is returned to the pool of available storage.

Storage is compacted only if a requirement exists for contiguous storage and if compacting can meet this requirement. Compacting is never performed unnecessarily. Instead the storage contents control routine always fits programs into gaps in the in-use storage, if possible.

9.2.3.5. Facilities Assignment

Available facilities and their disposition are indicated to the system at system generation time; thereafter, the executive system assigns these facilities, as needed and as available, to fulfill the facilities requirements of all runs entering the UNIVAC 1110 System. The executive system maintains current inventory tables, that indicate what facilities are available for assignment, and which runs are using the currently unavailable facilities.

9.2.3.6. The File-Control System

The UNIVAC 1110 File-Control System affords the highest degree of operational flexibility in storing and retrieving data, without concern for the physical characteristics of the recording devices. Thus, most files are made insensitive to input/output media characteristics, as the system adjusts the interface between the file and the device. Security measures ensure that files are not subject to unauthorized use or destruction. Facilities are provided to roll out files from mass storage devices to magnetic tape, as well as reconstruct such files on the mass storage devices when the user calls for them.

Comprehensive utility routines are available for manipulation of files and for informing the user of current status and structure of his files. Provisions are made for random storage and retrieval-access of data, under the direction of the user. User program files and data files are maintained and processed in the same environment.

9.2.3.7. Operator Communications

The executive system has been designed for operation with a minimum of operator intervention. However, some functions frequently in use are beyond the scope of the executive system, while others demand operator concurrence. In addition, certain information must be presented automatically to the operator, while other information must be available to answer operator requests.

Operator functions are required for a large variety of activities. The UNIVAC 1110 Executive System groups them into four classes, thus equally dividing operator duties in a multi-operator installation. These four functional classes are system control, input/output activity, communications activity, and hardware confidence activity. These functions may be associated with as many as four system consoles or as few as one, depending on the complexity and layout of the installation.

The executive system displays information such as current system load and operator requests associated with I/O setup and I/O interlocks. The operator can request other information, such as backlog status. If the display area becomes filled up, the executive defers lower priority displays.

9.2.3.8. Diagnostic System

A comprehensive diagnostic system within the UNIVAC 1110 Executive System aids in checking out user programs. Both allocation time and assembly time commands trigger snapshot dumps. Postmortem dumps are available through an executive control statement.

9.2.3.9. Input/Output Device Handlers

The input/output device handlers control the activities of all I/O channels and peripheral equipment attached to the UNIVAC 1110 System.

9.3. SYSTEM PROCESSORS

The system processors of the UNIVAC 1110 Operating System are programs which provide for the utilitarian functions required to edit program linkages, modify and maintain files, and provide diagnostic information on the status of the system and programs upon termination.

9.3.1. Collector

The collector is designed to provide the user with the means of collecting and linking relocatable subprograms to produce an absolute program in the form ready for execution under the control of the executive system.

9.3.2. FURPUR

FURPUR (file/program utility processor) consists of a set of file maintenance routines which provide the flexibility in management and manipulation of cataloged or temporary files containing data or programs.

9.3.3. Postmortem and Diagnostic Processor

The postmortem and diagnostic processor (PMD) reads the diagnostic tables which are output from the collector, and determines which parts of the program text are related to specific banks, segments, elements, and location counters. PMD dumps the banks of a program serially, producing label headings for each bank and its constituent segments, elements, and location counters.

9.3.4. DATA and ELT Processors

The DATA and ELT processors are used to insert data descriptions and elements in a library.

9.3.5. Secure Processor

The secure processor, also known as the file administration processor, uses a source language structure which allows the user to define specific tasks with simple COBOL-like statements. The processor's primary functions are to produce backup tapes for cataloged files, and to provide a recovery mechanism for these files in case of system failure.

9.3.6. Text Editor (EDIT Processor)

The EDIT processor is a text editor which enables a user to modify and/or move character strings in either program files or data files.

9.4. LANGUAGE PROCESSORS

9.4.1. The UNIVAC 1100 Series Assembler

The UNIVAC 1100 Series Assembler translates a symbolic language composed of brief expressions to machine-language relocatable object coding for the UNIVAC 1110 System.

The symbolic language includes a wide variety of sophisticated operators which allow the development of a desired object code based on information generated at assembly time. The instruction operation codes are assigned mnemonic codes which describe the hardware function of each instruction. By the use of assembler directives, the programmer can generate data words, values, or instructions based on specific conditions at assembly time. Multiple location counters make it possible to prepare for program segmentation and control address generation during assembly of a source code program. The assembler produces a relocatable binary output in a form suitable for processing by the loading mechanism of the system. If requested, it supplies a listing of the original symbolic coding and an edited octal representation of each word generated. Flags indicate errors in the symbolic coding detected by the assembler.

9.4.1.1. Symbolic Coding Format

When writing instructions using assembly language, the programmer is primarily concerned with three fields: a label field, an operation field, and an operand field. It is possible to relate the symbolic coding to its associated flowchart, if desired, by appending comments to each instruction line or program segment.

All fields except the label field, which begins in column 1, are in free form providing the greatest convenience possible for the programmer. Consequently, the programmer is not hampered by the necessity of considering fixed form boundaries in the design of his symbolic coding. Appendix C lists the instructions recognized by the assembler.

9.4.1.2. Assembler Directives

The symbolic assembler directives control or direct the assembly processor just as operation codes control or direct the central processor. They are represented by mnemonics written in the operation field of a symbolic line of code. Their flexibility is the key to the power of the assembler. The directives are used to equate expressions, to adjust the location counter value, and to afford the programmer special control over the generation of object coding.

9.4.1.3. Additional Features

Facilities for interprogram communication permit separately assembled programs (or subprograms) to be linked together at load time. A label followed by an asterisk is defined as an external label which can be referenced by other programs, as well as by the program in which it is defined. A job to be executed may be composed of many subprograms (or elements). The recompilation of any element does not necessitate the recompilation of the remaining elements which compose the job. The program is constructed, using the technique above, at or before the time of execution.

9.4.2. FORTRAN V

FORTRAN V is an algebraic language designed primarily for scientific and engineering computations. It closely resembles the language of mathematics. It is the logical outgrowth of the earlier FORTRAN languages and is generally compatible with them (although the earlier languages are not a proper subset of FORTRAN V). The FORTRAN V language has been extended to provide more flexibility in data handling and to make programming easier. FORTRAN V, being an outgrowth of the earlier FORTRAN languages (in particular, UNIVAC 1107 FORTRAN IV and IBM FORTRAN IV as announced in IBM form C-28-6274-1) accepts these languages as compatible, although the reverse is not necessarily true.

FORTRAN V has all the features of the proposed ANSI FORTRAN IV language plus many valuable extensions which significantly increase the power and flexibility of the language, particularly in the areas of data handling. For further information, consult the *UNIVAC 1108 Multi-Processor System FORTRAN V Programmers Reference, UP-4060,* (current version).

9.4.2.1. Language Extensions and Enhancements

The following extensions and enhancements are currently available in UNIVAC 1107 FORTRAN IV and are included in UNIVAC 1108 FORTRAN V.

- The PARAMETER statement assigns specified integer values to specified variables at the time of compiling; PARAMETER I = 2 replaces I with the integer 2 whenever it occurs in the source program. This facilitates the assignment of different values to frequently used parameters in different compilations of the same program.

- The ABNORMAL statement permits increased optimization of object programs. Where common subexpressions occur within a program unit, it is obviously desirable to evaluate each subexpression only once. Where the common subexpressions contain function references, however, there is a possibility that the function will produce a different result upon successive references with the same arguments. Because of this possibility, most FORTRAN systems are forced to reevaluate subexpressions containing function references at each occurrence. UNIVAC FORTRAN V permits all functions that can produce different results from identical sets of arguments to be designated ABNORMAL. All common expressions except those that reference ABNORMAL functions are evaluated only once. When the ABNORMAL statement does not appear in a program, all function references except library functions are considered ABNORMAL and are reevaluated at each occurrence, as in most other FORTRAN Systems.

- Non-standard subroutine returns (of the form RETURN k) are permitted where k specifies the subroutine argument to which a return is made.

- In conjunction with RETURN statements, statement labels may be used as subprogram arguments.

- A variable may have up to seven subscripts.

- Internal subprograms are permitted; that is, main and internal subprograms are part of the same program unit, which requires only one compilation.

- Variables of different types may occur in the same expression with two exceptions:

  Logical variables may not be mixed with other types.

  Double-precision and complex variables cannot be mixed.

- Extended subscript expressions are permissible, having the form $\pm M_1 \pm M_2 \ldots \pm M_i$ where M is of the form $K_1 * K_2 * K_3 \ldots K_j$. The K's represent either an integer constant, an unsubscripted integer variable, or a parameter variable. No more than one K may be a DO index.

- Forward and backward DO loops (that is, increasing and decreasing index variable) are permitted.

- A generalized assigned GO TO may be used; the assigned GO TO need not have a list of possible assignments.

The following language extensions and enhancements, not available with FORTRAN IV or earlier versions, are now available with FORTRAN V:

- A string of consecutive bits, called a field, may be defined and operated on by making use of the FLD (e_1,e_2,e_3) intrinsic function, where e_1 and e_2 determine a field of the expression e_3. e_1 and e_2 are integer expressions which give the starting position (e_1) and the length (e_2) of the field being defined. The FLD function may be used for extraction and insertion of bit fields.

- The NAMELIST statement which is nonexecutable, provides data-characteristic information at object time, and may be used instead of a LIST on an INPUT/OUTPUT statement and the associated FORMAT statement. A NAMELIST name (1–6 alphanumeric characters) is defined by its appearance in a NAMELIST statement, and thereafter may appear only in an input or output statement that requires a list.

- The DEFINE statement is of the form DEFINE $R(a_1, \ldots, a_n) = e$ or DEFINE $R = e$, where R and a_i are variable names and e is any expression not involving any undefined R's. The DEFINE statement generates in-line code when a procedure is involved, eliminating the overhead of a subroutine and enabling the optimizing capabilities to apply. Such a statement provides the following benefits:

  – All statement functions of FORTRAN IV operate more efficiently at object time.

  – Mathematical equivalence between arrays and variables can be attained.

  – Subscripting of subscripts is in effect, permitted to any level.

  – Any legal FORTRAN V expression can be treated as a subscript expression.

  – Dynamic storage allocation can be achieved.

- The INCLUDE statement is of the form INCLUDE n, LIST where n is the name assigned to a set of FORTRAN statements, previously filed with the procedure definition processor which are to be included in the program at this point. The word LIST is optional and if entered, the "Included" statements will be listed whenever the source program is listed. Thus, a frequently used number of statements (e.g. specification statements or a set of internal subprograms) may be added to the source code from an internally available element.

- The IMPLICIT type statement of the form IMPLICIT type (a_1, $a_2 \ldots a_n$), type (b_1, $b_2 \ldots b_n$) where type is INTEGER, REAL, LOGICAL, DOUBLE PRECISION or COMPLEX and the a_i represent alphabetic characters or a range of alphabetic characters. The IMPLICIT type statement allows the user to declare the type of variables by specifying that variables beginning with certain designated letters (the a_i) are of a certain type.

- The ENTRY statement is of the form ENTRY name (a_1,a_2, ..., a_n) where name is the name of an entry point and where the a_i are dummy arguments. The entry statement permits an entry to an internal or external subroutine or function by a CALL statement or a function reference to an ENTRY statement. Entry is made at the first executable statement following the ENTRY statement.

- The compile time-interpretive DELETE statement provides the programmer with a simple facility to prevent compilation of a section of source code. It is of the form DELETE n or DELETE n,V where n is a statement label and V is the integer 0 or 1 (or is a name assigned the value 0 or 1 by the PARAMETER statement). V=0 implies that the DELETE statement is not effective while V=1 implies that DELETE is effective.

- FORTRAN V processes double precision quantities in UNIVAC 1110 double precision format, within:

  - The FORTRAN V compiler
  - Mathematical function routines (where appropriate)
  - The I/O conversion routines
  - Compiled FORTRAN V programs

- The arithmetic type of the argument to library and intrinsic functions is used by the compiler to determine the correct function routine to be called (that is, SQRT, DSQRT, or CSQRT for REAL, DOUBLE PRECISION or COMPLEX arguments respectively.)

- The UNIVAC 1110 FORMAT control has been augmented by the addition of new FORMAT control specifications of the form:

  Gw.d Used for input and output of any of the five types of variables. If the output item is REAL, E or F editing code is used depending on magnitude.

  Tw Causes the pointer in an input or output record to point to the w^{th} character in the record.

  Lw Is logical field specification.

- A master space character (7–8 keypunch) will cause the compiler to ignore all subsequent information on the line. The space thus ignored may be used for comments.

- Hollerith strings may have the form '$c_1,c_2,\ldots, c_i$' where c_i is any Hollerith character, including blank.

- In a typeless expression the computer word (36 bits) is considered as a bit string. Permissible typeless expressions are alphanumeric constants and Boolean functions. The Boolean functions are AND (e_1, e_2), OR(e_1, e_2), BOOL(e_1), COMPL(e_1), and XOR(e_1, e_2).

- An additional input statement of the form READ (unit, format, ERR=n, END=m) is included. The END and ERR clauses can only be indicated in the third or fourth argument positions. Control changes to statement number n, if an input error is encountered. Control goes to statement number m, if an end-of-file is encountered.

- Miscellaneous Extension

  – Free-field input is specified by an empty FORMAT statement:

```
      READ (5,100)A,B,C
  100 FORMAT ( )
```

  – In a subroutine or function subprogram the maximum dimension for an array may be transferred as an argument. In FORTRAN V, the information may be provided via COMMON.

  – An array may be dimensioned in an explicit type statement by including the dimension parameters in parentheses.

  – The EDIT statement provides the user the option of suppressing and restoring compiler listings for any part of the program, overriding control card listing options. Valid forms of the statement are:

```
START EDIT SOURCE
START EDIT CODE
STOP EDIT SOURCE
STOP EDIT CODE
```

  – FORTRAN V accepts an & (2–8 punch) preceding a label as an argument for use in a subroutine or function subprogram to indicate the transmission of a statement label.

  – Data will be accepted in the explicit type statement or DIMENSION statements; for example, REAL A/1. 51/,B(2,3)/6*1.01/.

9.4.2.2. Compiler Organization

The UNIVAC 1110 FORTRAN V source language processor accepts FORTRAN statements and produces a highly efficient relocatable object code element. FORTRAN, like all other UNIVAC 1110 processors, generates its own code and does not require an assembler pass.

The FORTRAN V compiler is modular and consists of six phases. Although the phases have been separated on the basis of general operations performed on the source program, not every phase processes the entire program.

The result of this is a compiler which, while quite rapid as a processor, also produces an object program optimized with respect to both storage requirements and execution time.

The compilation process involves the successive execution of the six phases summarized in the following list:

Phase 1 transforms the FORTRAN V program source code into an internal format. Files and tables of relational information, implicit in the source program but not easily accessed, are constructed.

Phase 2 deals with storage assignments for variables and performs an analysis of loops.

Phase 3 deals with arithmetic optimization and index register optimization.

Phase 4 deals with loop optimization.

Phase 5 deals with code generation and storage assignment for those quantities not assigned storage by Phase 2.

Phase 6 the final phase, completes the generated instructions in a relocatable binary format and optionally edits all output, including error messages.

The compiler performs several types of optimization on a source program:

- Local Optimization

  This involves the reduction of expressions involving nothing but constants to a single constant.

- Inter-statement Arithmetic Optimization

  This optimization has three forms: the elimination of redundancies in loading of index and arithmetic registers, the recognition of common sub-expressions from previous statements, and the removal from a loop of these computations within a loop structure which are constant relative to the loop.

- Inter-statement Indexing Optimization

  This involves a study of the DO-loop structure, entries and exits from loops, the form of subscripts and the loop parameters.

9.4.3. Conversational FORTRAN V

An important element of the programmed systems support provided with a UNIVAC 1110 System is the Conversational FORTRAN processor which provides a dynamic and efficient means for constructing, debugging, and modifying a program.

The prominent characteristic of the system is that it enables the user to program from a remote device with a minimum amount of preplanning. He can think freely on line by constructing and testing routines in a nonsequential trial-and-error manner. The time in which the user engages the conversational system is considered a "session". During a session, the system responds on a statement-by-statement basis. Each statement is translated, verified and if desired, executed immediately. Once this is done, the system sends a reply to the device. The user then reacts to the system's message. If he is using the system as a desk calculator, the message will most likely be the result of a requested computation. If he is constructing a program, it may be diagnostic information indicating that the statement contained an error. In either case, the user now converses with the system as to the next step to be taken.

9.4.3.1. System Features

Some of the features provided by the conversational system are:

- The user has immediate and sustained access to the machine.
- The user has the ability to construct, execute, and alter statements or complete routines; to change values of variables; to rename variables; and to request information selectively.
- The user can store complete routines or portions of routines, take checkpoints during execution of a complex of routines, and load source-statements from optional devices.
- The user may continue his session at the device after an extensive time lapse.
- The user is provided with diagnostic messages and logical analysis to allow modification and debugging to take place at the same level as routine construction.

9.4.3.2. System Concepts

A session is the time in which the user engages the services of the conversational processor. Normally a session is identified by a user-selected name. During a session, the user is free to use the processor in the manner that is most conducive to the accomplishment of his objective. He may construct, execute, and save single statements, groups of statements, or a complete program.

The environment at the device is the session the user is currently engaged in and all those routines needed for solution of a particular problem. These routines can be assembled from either the user's library or they can be system routines provided by the processor.

All conversational operations are performed during the current session at the device. When the user is constructing, executing, testing, or modifying statements, he is considered to be performing operations on an active image. The user has the ability to obtain, from storage, source statements which then become part of the active image. Only one image may be active at any time at a device.

9.4.3.3. Conversational FORTRAN Processor and the Executive System

The conversational FORTRAN processor (CFOR) is but one of a number of source-language processors made available to the user by the executive system. The action it performs is integrated with the multitude of activities controlled by the executive. For this reason, specific attention was given to the choice of services it renders so as to eliminate duplication of facilities already offered by the system. This is particularly applicable to the service language portion of the programming language. It is assumed that the user will employ the normal job-control language statements of the executive to perform housekeeping services.

9.4.3.4. Conversational FORTRAN Language

Conversational FORTRAN consists of procedural statements defined by the UNIVAC 1108 FORTRAN V language, and service statements which allow the user to regulate the system during construction, execution, and modification of a program.

The conversational FORTRAN compiler analyzes the FORTRAN statements introduced by the user at his terminal and immediately checks these for errors. If there are any, it identifies the error for the user so that it can be corrected. If the FORTRAN statement is without error, it is stored in an intermediate form for interpretive execution at a future time; or the statement may be executed immediately and its result stored for later use.

The conversational FORTRAN language encompasses the American National Standards Institute (ANSI) FORTRAN. It is also defined by FORTRAN V and any programs constructed by the UNIVAC 1110 Conversational FORTRAN compiler may also be compiled by the batch compiler.

9.4.4. COBOL

Three COBOL processors are provided with the UNIVAC 1110 Operating System:

- UNIVAC 1100 Series COBOL, based on the Department of Defense publication *COBOL Edition 1965.*
- UNIVAC 1100 Series ANSI COBOL, based on the American National Standard COBOL, X3.23–1968.
- UNIVAC 1100 Series ANSI/ASCII COBOL, based on the American National Standard COBOL, X3.23–1968, with major extensions including ASCII/EBCDIC/FD code capabilities.

9.4.4.1. UNIVAC 1100 Series COBOL

UNIVAC 1100 Series COBOL is based on the Department of Defense publication *COBOL Edition 1965.* The UNIVAC 1100 Series COBOL accepts atatements written in the COBOL language and produces a program ready for execution. For further information, consult the *UNIVAC 1106 System/1108 Multi-Processor System COBOL Under EXEC II and EXEC 8 Supplementary Reference, UP-7626* (current version).

The major features implemented on the UNIVAC 1110 System are:

- COMPUTE verb and arithmetic expression in conditional statement
- Table handling (SEARCH)
- Segmentation
- Mass storage
- COPY
- Characters used in arithmetic expression (+, –, *, /, **, =), and in relational expressions (=, >, <)
- Literals up to 132 characters
- The ENTER verb
- The LOCK option on the CLOSE verb
- The ADVANCING option on the WRITE verb
- The REVERSED option of the OPEN verb
- Operands used in arithmetic can be up to 18 digits long.
- AND and OR connectors in compound conditions
- Parentheses in compound conditions
- All abbreviations of conditional statements
- The OBJECT-COMPUTER paragraph
- The APPLY clause
- RERUN
- The DATA-COMPILED clause
- Library provisions
- Multiple results from arithmetic verbs

The following features, implemented in UNIVAC 1100 Series COBOL, are special UNIVAC extensions to the COBOL language:

COBOL sub-program communication

MONITOR

Dynamic date

Common storage

Page control

A report generator, not part of the COBOL processor, is available for COBOL programs.

(1) The Processor

The COBOL processor is a six-phase compiler operating within the UNIVAC 1110 System minimum configuration. The compiler is completely modular in relocatable elements and it is handled as any program in the system for easy expandability and maintenance. Likewise, the COBOL processor produces as its output relocatable binary elements stored on the drum or mass storage, which are indistinguishable from other elements in the system. Other outputs from the compiler include extensive diagnostic messages, source language listings, machine language listings, and special cross reference listings of name definitions and their references. The machine language listing consists of side by side procedure division statements and the corresponding generated symbolic machine code.

The compiler diagnostics are of two categories:

Warning – A minor source language error has been detected which does not affect the program being produced. This type of diagnostic is identified by the word ERROR preceding the actual message.

Fatal – A major source language error has been detected which very likely will adversely affect the program being produced. The compiler will continue to process the source language but will flag the program produced as in error. This type of diagnostic is identified by the word ERROR* preceding the actual message.

(2) Special Features

- Segmentation – COBOL programs can be segmented by use of priority numbers on procedural sections.
- Monitor – Provides dynamic program checkout facilities.
- Library – The procedure definition processor is available to store environment, data and procedure division descriptions so they can be retrieved by the COPY and INCLUDE verb.
- Rerun – The programmer can specify rerun after any number of records have been processed or when an end of reel is encountered.
- Common Storage – Since COBOL programs can be chained (an executive function), intermediate data results can be maintained between programs using the common storage provision of UNIVAC COBOL. The elements sharing common storage may be from another processor such as FORTRAN V.
- Overpunched Sign Convention – Tapes and cards prepared in the overpunched sign convention can be processed.

9.4.4.2. UNIVAC 1100 Series ANSI COBOL

UNIVAC 1100 Series ANSI COBOL is based on the American National Standard COBOL, X3.23–1968. In addition to the standard features, UNIVAC 1100 Series ANSI COBOL contains several useful extensions to the standard.

UNIVAC 1100 Series ANSI COBOL contains the following modules of the ANSI standard:

- NUCLEUS level 2
- TABLE HANDLING level 3
- SEQUENTIAL ACCESS level 2
- RANDOM ACCESS level 2
- SORT level 2
- SEGMENTATION level 2
- LIBRARY level 2

This is the full ANSI standard with the exception of the report writer.

The principal extensions to the ANSI standards are:

- CODASYL oriented subprogramming feature using the CALL and ENTRY verbs with LINKAGE SECTION.
- UNIVAC subprogramming feature (USE FOR ENTRY POINTS, GO TO, and PERFORM).
- UNIVAC page control features.
- COMMON-STORAGE SECTION for using blank common. This feature may be used with subprogramming to provide communication between COBOL/COBOL and COBOL/FORTRAN programs.
- Indexed-sequential file handling capability (COBOL verbs using the ISFMS indexed-sequential handler).
- A new COBOL file handler with superior input/output capabilities.
- Special printer forms handling (change forms messages for operator action).
- READY and RESET debugging verbs.
- MONITOR debugging verb with the ability to suspend and reactivate its function.
- Multiple receving fields for ADD and SUBTRACT verbs.

9.4.4.3. UNIVAC 1100 Series ANSI/ASCII COBOL

UNIVAC 1100 Series ANSI/ASCII COBOL is based on the American National Standard COBOL X3.23–1968, various CODASYL conventions, and contains significant UNIVAC extensions.

The following major features are included in UNIVAC 1100 Series ANSI/ASCII COBOL:

- Full ANSI standards
- Communications (CODASYL)
- Asychronous Processing (CODASYL)
- Inter-Programming Communications and Data Storage (CODASYL)
- Vertical Form Control (CODASYL)
- Indexed Sequential File Capability
- Reentrant Code
- UNIVAC 1108 Floating Point Usage
- ASCII, EBCDIC, and FD Code Handling

9.4.5. NU ALGOL

The NU ALGOL language allows the mathematician or engineer to prepare programs for the UNIVAC 1110 System without the necessity of becoming familiar with the details of the internal machine operation. The NU ALGOL compiler then generates, from this pseudo mathematical source language, efficient coding in a relocatable binary format acceptable to the executive for execution, the filing system for cataloging and filing, or both.

UNIVAC NU ALGOL is an extended hardware representation of ALGOL 60 designed to employ the UNIVAC 1110 System and associated peripheral equipment efficiently. Certain extensions to basic ALGOL have been made. It can handle the powerful input/output logic; it has the ability to name strings; and it is capable of performing complex and double precision arithmetic. It is a four pass compiler that generates more efficient code than available in the previous UNIVAC 1100 Series ALGOL System.

9.4.6. BASIC

BASIC is a reentrant processor which accepts source language statements from remote users, checks the syntax and issues diagnostics immediately when a syntax error is detected. By use of a service language, the user can direct the compiler to execute his source statements. The compiler responds by executing, interpretively, the intermediate language that is output from the syntax analysis. The result is displayed at the remote point so that the user can observe and respond if he desires.

9.4.7. Procedure Definition Processor (PDP)

The Procedure Definition Processor (PDP) accepts source language statements defining assembler, COBOL, or FORTRAN procedures and builds an element in the user-defined program file. These procedures may subsequently be referenced in an assembly or compilation without definition.

9.4.8. JOVIAL

JOVIAL is a block structure symbolic algorithmic language which is procedure-oriented, describing data processing in terms of the logical operations required.

A program written in JOVIAL basically consists of a list of statements which are commands structured in the form of a sentence and terminated by a dollar sign.

9.5. UTILITY PROCESSORS

9.5.1. LIFT

LIFT is a source language translator which accepts a FORTRAN II source language program as input, performs a translation, and prepares a source language program acceptable to the FORTRAN V compiler. There is a need for translation since FORTRAN II is not a proper subset of FORTRAN V; that is, there are statement types in FORTRAN II that are not acceptable to FORTRAN V. LIFT itself, written in FORTRAN V, is fully integrated with the executive system.

There are nine areas of incompatibility between FORTRAN II and FORTRAN V, and the basic purpose of LIFT is to generate FORTRAN V source statements which replace the unacceptable FORTRAN II statements.

(1) The F card

(2) Functions

(3) Boolean statements

(4) Double-precision and complex statements

(5) COMMON statements

(6) Arithmetic statement functions

(7) Dimension statements

(8) Hollerith literals

(9) Implicit multiplication

There are also five types of FORTRAN II statements that, although acceptable to the FORTRAN V processor, are converted to their FORTRAN V equivalents. LIFT offers two features that ease the transfer between computers: the ASSIGN and REPLACE card options. The ASSIGN card allows a temporary change to be made to the I/O assignment table, and the REPLACE card allows the user to have every occurrence of a variable name replaced with another variable. The standard output produced by LIFT consists of a listing of the FORTRAN II program, an annotated list of the translated program, and a symbolic program element suitable for use as input to any FORTRAN V compiler.

9.5.2. SLEUTH I Translator

The SLEUTH I translator accepts programs written in SLEUTH I, and converts them into a format accepted by the UNIVAC 1100 Series Assembler.

9.5.3. CUR to FUR Conversion

This processor converts magnetic tapes created by the UNIVAC 1107 Complex Utility Routine (CUR) to magnetic tapes acceptable as input to UNIVAC 1100 Series Program file. The processor will accept UNIVAC 1107 symbolic elements, COBOL library elements, and procedure elements, converting them to the proper formats.

9.5.4. FLUSH

FLUSH (Flowcharting Language for User's Simplified Handling) is a routine which accepts assembler format input to produce a flowchart. FLUSH can:

- Be instructed through the use of parameters, called options, which are punched in the comments field of the instruction statement, or
- Perform an analysis based on the assembler instruction statements.

9.5.5. SSG Processor

The SSG processor is a general purpose symbolic stream generator. Any variety of symbolic streams, varying from a file of data to a runstream which configures an executive system, may be generated. Directions and models for the building of the desired stream images are conveyed to SSG through a skeleton which is written in SYMSTREAM, an extensive manipulative language.

9.5.6. CULL Processor

CULL is a processor which produces an alphabetically sorted, cross-referenced listing of all symbols in a specified set of symbolic elements. Each symbol processed by CULL may contain up to 12 alphanumeric characters plus the dollar sign. Provisions are included, via options, to selectively include or exclude defined symbols from the output.

9.5.7. UNADS (UNIVAC Automated Documentation System)

UNIVAC Automated Documentation System (UNADS) accepts the contents of a document and composes that document according to the user's specifications. The user may specify the format of the document, the method of composition of the document body, the table of contents, the index contents, and other related activities.

The user's specifications are called commands (contained in the UNADS command language) and macros (defined by the user in terms of UNADS commands). These commands and macros are intermixed with text. The UNADS commands are English language type directives and, although they are easy to read, they are lengthy. Accordingly, the system intends that the overwhelming majority of user specifications will occur in the form of macros which may be defined as being equivalent to a few or many UNADS commands.

The system also has a facility which permits the user to call upon large collections of macro definitions at the start of runs, using only a short expression similar to a macro. This facility enables the user to devote his efforts to preparation of text and easily directing its composition.

9.6. LIBRARIES

The UNIVAC 1110 Operating System includes the following libraries:

- Assembler Procedure Library
- FORTRAN Library
- COBOL Library
- ALGOL Library
- JOVIAL Library
- Services Library
- Diagnostic Library
- Item Handler
- Processor Interface Routines
- Sort/Merge
- MATH-PACK
- STAT-PACK

9.6.1. Sort/Merge

The UNIVAC Sort/Merge package is fully modular, with every functional unit completely self-contained. This permits the various units to be individually adapted to their own particular tasks, enabling them to be associated in the most effective form, and allowing updating and augmentation.

The package is not a generator of specialized sort/merge routines; rather, the user calls and adapts the independent modules for all his specific sorting needs by presenting his parameter values on control cards at load time.

In the internal sort the replacement selection method, which takes advantage of any inherent sequence in the original data, is used. Strings may be written upon magnetic tape or drum. The UNIVAC FH-432 and FH-1782 Drums, because of their high transfer rates and rapid access, minimize processor waiting time and thus greatly speed efficient sort operations. Any random access unit areas may be defined by the supervisor and these are automatically used by the subsystem if advantage can be gained thereby.

The input data to be sorted may be stored on magnetic tape, punched cards or magnetic drum. User own coding may be inserted on the first and final passes of the sort and merge operation and may also replace the standard comparison routines. Sorting generally requires the use of two magnetic tape units, although additional units can be employed to give faster times.

Keys may be in multiple form and can be recorded, modified, and packed. Standard collating sequences are intrinsically provided for, but the user may define any collating sequences he requires, up to a maximum of seven, and any combination of these may be utilized in the same run. Fixed or variable length items can be handled.

The sort/merge package normally uses 20,000 words of storage, 262,000 words of magnetic drum storage, and magnetic tape units of any kind as required; but the user may specify more central and drum storage, and additional magnetic tape units to increase efficiency and speed.

9.6.2. MATH-PACK

MATH-PACK provides the UNIVAC 1110 System with a comprehensive library of 78 fundamental mathematical subprograms coded in FORTRAN V. The purpose of this library is to present to the mathematician, the scientist, and the engineer many of the more frequently used tools of numerical analysis. These subroutines and function subprograms are designed to speed up and simplify solutions to problems encountered in many areas of scientific research.

The subprograms are grouped into 14 categories:

(1) Interpolation
(2) Numerical integration
(3) Solution of equations
(4) Differentiation
(5) Polynomial manipulation
(6) Matrix manipulation: real matrices
(7) Matrix manipulation: complex matrices
(8) Matrix manipulation: eigenvalues and eigenvectors
(9) Matrix manipulation: miscellaneous
(10) Ordinary differential equations
(11) Systems of equations
(12) Curve fitting
(13) Pseudo-random number generators
(14) Specific functions

Each of these classes contains subroutines and function subprograms that are generally useful for problems commonly encountered by mathematicians, scientists, and engineers.

9.6.3. STAT-PACK

STAT-PACK provides the UNIVAC 1110 System with a comprehensive library of 91 fundamental statistical subprograms coded in FORTRAN V. The purpose of this library is to present to the statistician, the scientist, the operations research specialist, and the engineer many of the more frequently used tools of statistical analysis. These subroutines and function subprograms are designed to speed up the preparation of solutions to statistical problems encountered within many areas of scientific research.

The subprograms are grouped into 13 categories:

(1) Descriptive statistics
(2) Elementary population statistics
(3) Distribution fitting and plotting
(4) Chi-square tests
(5) Significance tests
(6) Confidence intervals
(7) Analysis of variance
(8) Regression analysis
(9) Time series analysis
(10) Multivariate analysis
(11) Distribution functions
(12) Inverse distribution functions
(13) Miscellaneous

Each of these classes contains subroutines and function subprograms that are generally useful for problems commonly encountered by statisticians, scientists, and engineers.

9.7. APPLICATIONS

9.7.1. APT III (Automatically Programmed Tools)

APT (Automatically Programmed Tools) is a system for the computer-assisted programming of numerically controlled machine tools, flame cutters, drafting machines, and similar equipment. It is production-oriented, written to take full advantage of numerically controlled techniques in engineering and manufacturing with the least expenditure of effort, time, and money.

APT enhances most of the usual advantages found in numerical control: reduced lead time, greater design freedom and flexibility, lower direct costs, greater accuracy, improved production forecasting, lower tooling costs, better engineering control of the manufacturing process, and simplified introduction of changes.

The APT III program represents over one hundred man-years of development and testing. After extensive experience with our earlier program, APT II, the Aerospace Industries Association made a new start and wrote APT III from the beginning, during the calendar year 1961. At the completion of this package, APT III was turned over to the Illinois Institute of Technology Research Institute for further development, under the APT long-range program. The use of certain parts of APT requires membership in this long-range program.

Univac Division participated in the original writing of APT III and has been a member of the APT long-range program from its beginning. Numerical control specialists are continually working to keep the UNIVAC 1110 APT program in the forefront of the art. As implemented on the UNIVAC 1110 System, APT III will continue to conform to the latest APT long-range program specifications.

9.7.2. PERT

The UNIVAC PERT system is a generalized applications program for project/program planning and control. The system is modular in nature and has TIME and COST modules. The modular design of this program allows separate processing of the time networks and the cost structure while simultaneously providing for integrated time and cost reporting.

The PERT/TIME module is both activity and event oriented. The input to this module is provided by a deck of cards (may be in the executive system input file) which describes the network to the system. These cards are then processed to perform network computations and, if needed, update a TIME master file, created by a previous run. As a final step, the PERT/TIME module generates various useful reports, such as activity reports, event reports, milestone reports, and others.

The PERT/COST system is based upon the *DOD/NASA Guide to PERT/COST System Design.* The DOD/NASA design is based upon the concepts of costing work packages rather than individual network activities. A work package is a discrete unit of work required to complete a specific job or process. The work packages of a research and development project are directly related to activities or groups of activities on the project network. The work package is the basic unit of the PERT/COST system for which actual project costs are collected and compared with estimates for purposes of cost control.

The input to the COST module consists of a deck of cards (may be any input file using the executive system) describing the work breakdown structure, the actual, budgeted and estimated costs for the work packages, and a table of labor and overhead rates. This information is then processed to accumulate costs through a cost breakdown tree and, if needed, is integrated with the TIME output. Required cost reports at desired summarizing levels are then generated.

The PERT system is oriented towards alphanumeric characters. All the necessary information, such as event codes, charge and summary numbers, responsible organizations, performing organizations, and resource codes, can be designated in any arbitrary fashion as a string of alphanumeric characters. Event codes, for example, do not have to be numeric, nor do they have to be in any particular sequence. The program module performs all necessary ordering internally. The system converts all computed times to calendar dates and makes necessary adjustments for vacations and holidays, if any.

9.7.3. GPSS III (General Purpose System Simulator)

The General Purpose System Simulator (GPSS III) provides for test and evaluation of a system by simulating operations on a model of that system. GPSS III is an interpretive program which accepts free format statements in which the user describes the model to be simulated. These statements specify the actions which are to be taken upon GPSS III entities. Entities of GPSS III are given symbolic names thereby allowing the model to be developed in terms of the real world system.

There are four entity classes in GPSS III: dynamic, equipment, statistical and operational. The dynamic entities (transactions) are the units of traffic. Dynamic entities compete for equipment entities within a model. This competition is based on transaction priority and the scheduled time at which an activity is to take place. To insure proper sequencing of activities, GPSS III maintains a simulation clock.

Equipment entities are of three types: facilities, storage, and switch. Facilities process only one transaction during any increment of simulation time. Storage has the property of capacity and is capable of simultaneously processing as many transactions as can be accommodated by the defined capacity. Switch is a binary indicator which can be used to record some system condition (physical or logical) that is instrumental in deciding when or how tasks are to be executed.

Statistical entities of GPSS III allow the input of numerical information, establish numerical relationships among system variables, and facilitate output of the effects of equipment competitions.

Operational entities allow operations to be performed on groups of entities (for example, searching, modifying) and provide means to influence the event scheduling technique of GPSS III.

9.7.4. FMPS (Functional Mathematical Programming System)

Functional Mathematical Programming System (FMPS) is an advanced mathematical programming language. Its features include:

- Procedures commonly used to solve linear programming (LP) problems (CRASH, OPTIMIZE, INVERT).
- A user oriented, FORTRAN like control language through which the user can establish the sequence of operations to be performed, control the treatment of exception conditions, and adjust tolerances if desired.
- Can be used as a self standing package or, if desired, as part of a user designed optimization package.
- A system to accommodate future extensions (for instance, integer programming) while retaining the current control language and general organization.
- Input procedures used to read matrix data from cards or tape in standard FMPS format or in various share formats (LP 90/94, UNIVAC 1108 LP, or CDC CDM4).
- Report writer which displays the solution or input matrix in an application oriented format.
- Selective output by masking operation.
- Generalized matrix generator.
- Procedures for saving the basis, restoring the basis, and procedures for obtaining error estimates and sensitivity analysis on the solution.
- Procedures which can be called during the solution process to perform non-linear adjustments of the matrix coefficients. These procedures are especially useful in problems pertaining to gasoline blending.
- Bounded variable code.

- Generalized upper bounded operating mode. This is a particularly valuable feature for transportation, weighted distribution, and multi-plant distribution LP problems.
- Mixed integer programming.
- Parameteric programming.
- Separable programming.
- Non-linear procedures through the use of the FMPS capability for recursive matrix modification.

FMPS makes use of most of the features of the FORTRAN V language and the UNIVAC 1100 Series Assembler.

9.7.5. SIMULA 67

SIMULA 67 is a general purpose programming language with a built-in simulation capability similar to, but stronger than that of SIMULA I. SIMULA 67 is an extension of ALGOL 60 and one of its features is that it is easily structured towards specialized problem areas, and hence will be used as a basis for special application languages. The central concept in SIMULA 67 is the object. An object is a self-contained program, having its own local data and actions defined by a class declaration. The class declaration defines a program (data and action) pattern, and objects conforming to that pattern are said to belong to the same class. SIMULA 67 contains one problem oriented class as part of the language; the class "SIMULATION". When used as a prefix to a block, features corresponding to those of SIMULA I are available. Transcription of SIMULA I programs into SIMULA 67 is accomplished through the use of the prefix "SIMULATION".

9.7.6. DMS 1100 (Data Management System)

Data Management System (DMS 1100) is a system which handles structured data files stored on mass storage devices. Its goals are to provide independence of data and programs, device independence, reduced or eliminated redundant copies of data, equal ease of access to data using any of several search keys, and the ability to select from, and use, any of several access techniques in the database. All of these goals are met in the shared database environment with database integrity assured by the system. Locks are provided at the file and record levels, as is the ability to save data to enable recovery in the face of hardware or software errors.

The system functional specifications are a modified version of the "October 1969 Report of the CODASYL Data Base Task Group". DMS 1100 comprises three independent modules; these are a data description language (DDL) translator, a data manipulation language preprocessor (DMLP) and a data management routine (DMR).

The DDL builds a set of data description tables which are stored in a standard cataloged file and are available to the remainder of the system as required. The DDL is used by the data base manager to control record structure, record relationships, file and record placement in the data base on physical devices, and definition of the access techniques to be used by the system in retrieving the records.

The DMR executes at application program run time and is driven by commands written in the applications program. The program can be written using either ASM or COBOL.

The DMLP is a COBOL language preprocessor which accepts a source program made up to ANSI standard COBOL source language and the data management commands. The output is a ANSI standard COBOL symbolic program which is then translated and mapped in a standard fashion. The resultant object program executes using the DMR.

Using the ASM interface and a set of interface subroutines, the data management system can be called from programs written in any language.

APPENDIX A. NOTATIONAL CONVENTIONS

Abbreviations and symbols frequently used in the description of the instruction repertoire are given below:

A	Arithmetic register. A control register in the GRS specified by the a field of an instruction.
a	Arithmetic register designator. In input/output instructions, "a" designates an I/O channel.
ACW	Access control word
BD	9-bit D-Base Storage Block Number
BDX	6-bit extension of BD
BI	9-bit I-Base Storage Block Number
BIX	6-bit extension of BI
BS	7-bit Program Effective Switch point

PSR base displacement fields

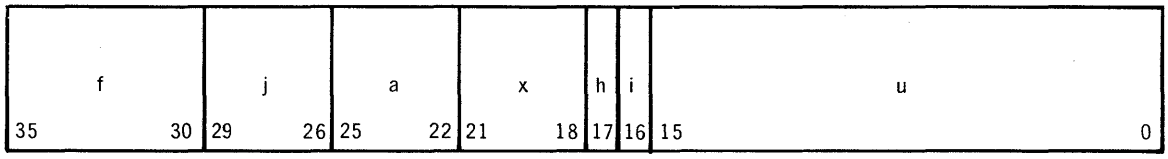
CAU	Command/arithmetic unit
CBR	Chain base register
CSR	Channel select register
E	A byte string whose starting word address is formed by summing the instruction u field, the instruction specified index register and J register zero. ($u + X + J0_{ow}$). Field 0_b in J0 points to the byte within word.
F	A byte string whose starting word address is ($u + (X + 1) + J1_{ow}$); Field 0_b in J1 points to the byte within word.
f	Function code, bits 35–30 of instruction word.

G	A byte string whose starting word address is (u + (X + 2) + $J2_{ow}$) Field 0_b in J2 points to the byte within word.
GRS	General register stack
h	h designator bit 17 of the instruction word. A value of 1 normally specifies incrementation of an index register.
IOAU	Input/output access unit
i	i designator bit 16 of the instruction word. A value of 1 normally specifies indirect addressing.
J	J register, GRS addresses 106_8–111_8 or 126_8–131_8
j	Partial word designator or function code extension, bit positions 29–26 of the instruction word.
M	The byte count contained in field BB2 in staging register 3
MD	Memory descriptor
MDP	Memory descriptor pointer
N	The byte count contained in field BB1 in staging register 3, GRS addresses 105_8 or 125_8
NI	Next instruction
P	Program address register
PSR	Processor state register
PSRE	Processor state register extension
PSRU	Processor state register utility
PSRUE	Processor state register utility extension
R	R register, GRS addresses 100_8 – 137_8
Ra	R register specified by the a-field of an instruction.
SLR	Storage limits register
SLRU	Storage limits register utility
SR1	Staging register 1 (R3), GRS addresses 103_8 or 123_8
SR2	Staging register 2 (R4), GRS addresses 104_8 or 124_8
SR3	Staging register 3 (R5), GRS addresses 105_8 or 125_8

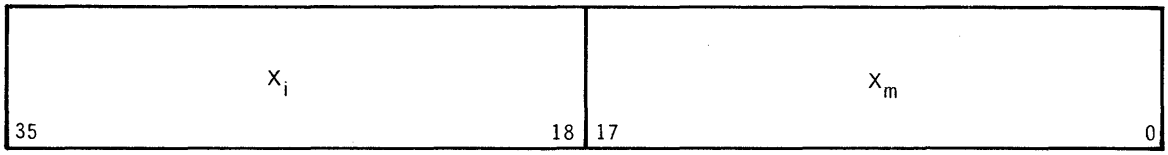
U	The effective address or value of the operand after application of indexing and indirect addressing.
u	The base address of the operand (or the base of the actual operand) as coded in u field of an instruction.
X	Index register
X_a	Index register specified by the a-field of an instruction
X_i	Increment portion of an index register
X_m	Modifier portion of an index register
X_x	Index register specified by the x-field of an instruction
x	Index register designator
()	Contents of
()'	Complement of contents of
|()|	Absolute value or magnitude
()c	Floating point biased exponent
$()_{17\text{-}00}$	Subscripts indicate the bit positions involved. A full word is normally not subscripted. Subscripts are also used to designate octal or decimal notation.
AND	Symbol denoting logical product, or logical AND
OR	Symbol denoting logical sum, or inclusive OR
XOR	Symbol denoting logical difference, or exclusive OR
	Direction of data flow

APPENDIX B. SUMMARY OF WORD FORMATS

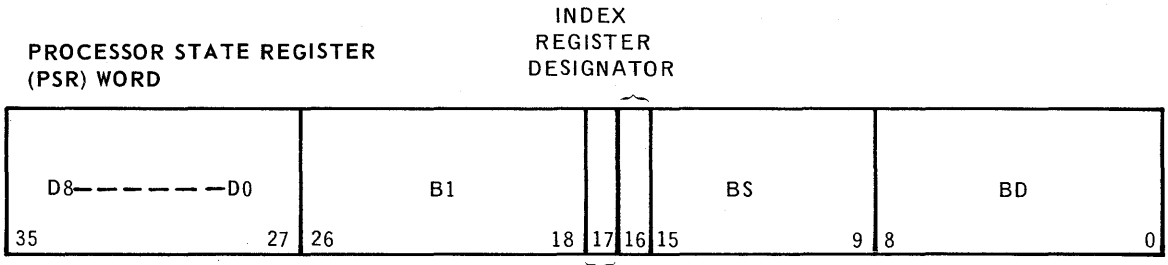
INSTRUCTION WORD



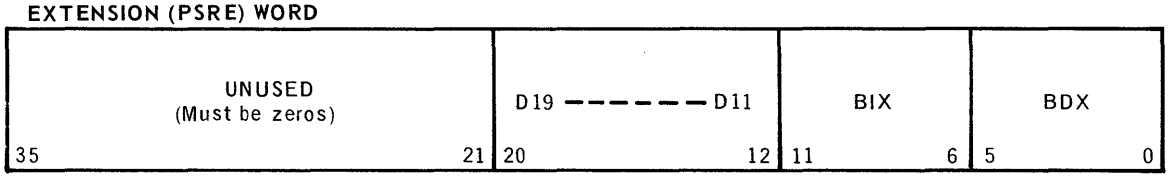
INDEX REGISTER WORD



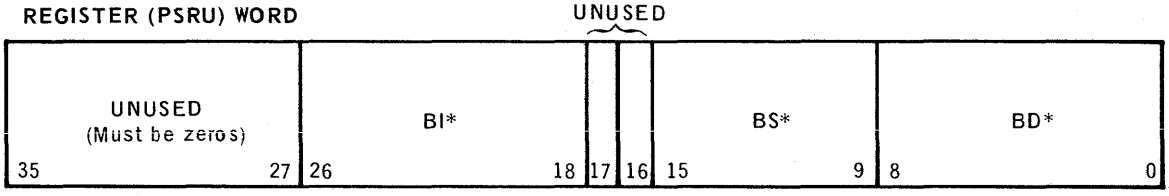
PROCESSOR STATE REGISTER (PSR) WORD



PROCESSOR STATE REGISTER EXTENSION (PSRE) WORD

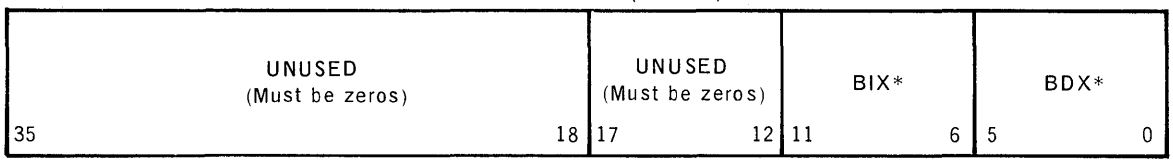


UTILITY PROCESSOR STATE REGISTER (PSRU) WORD

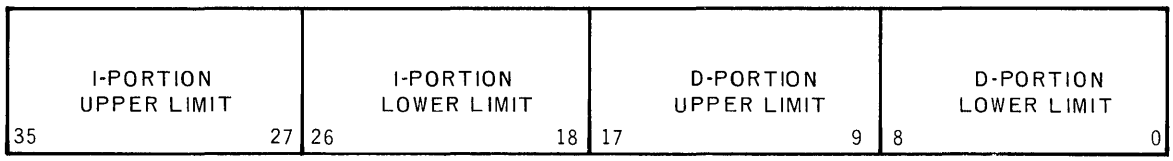


*Same as for corresponding PSR and PSRE fields.

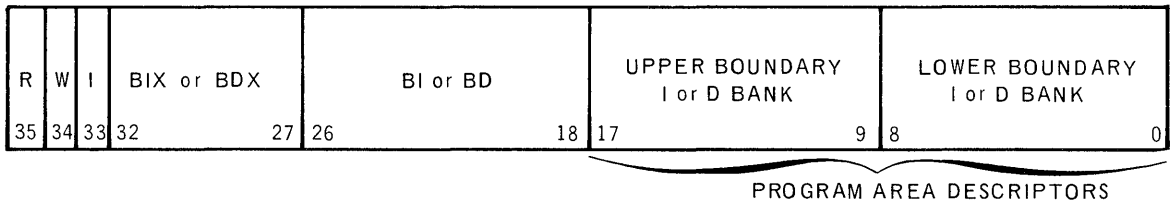
UTILITY PROCESSOR STATE REGISTER EXTENSION (PSRUE) WORD



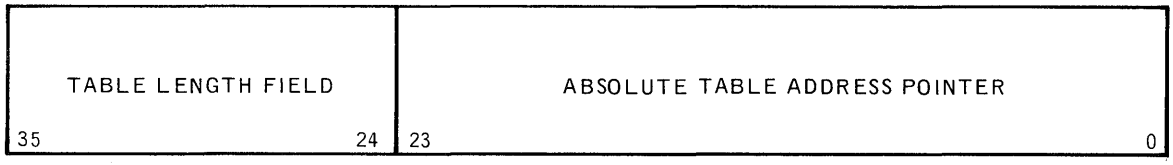
STORAGE LIMITS REGISTER (SLR) WORD **



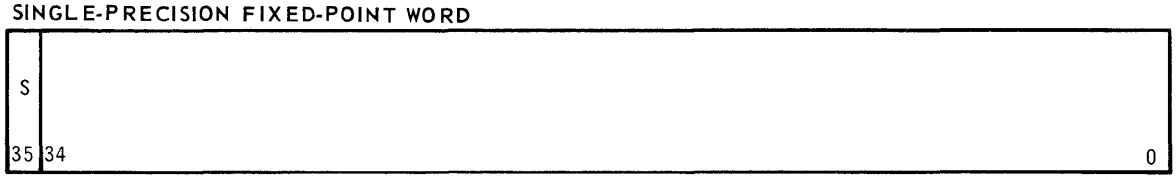
MEMORY DESCRIPTOR (MD) WORD



MEMORY DESCRIPTOR POINTER WORD



SINGLE-PRECISION FIXED-POINT WORD

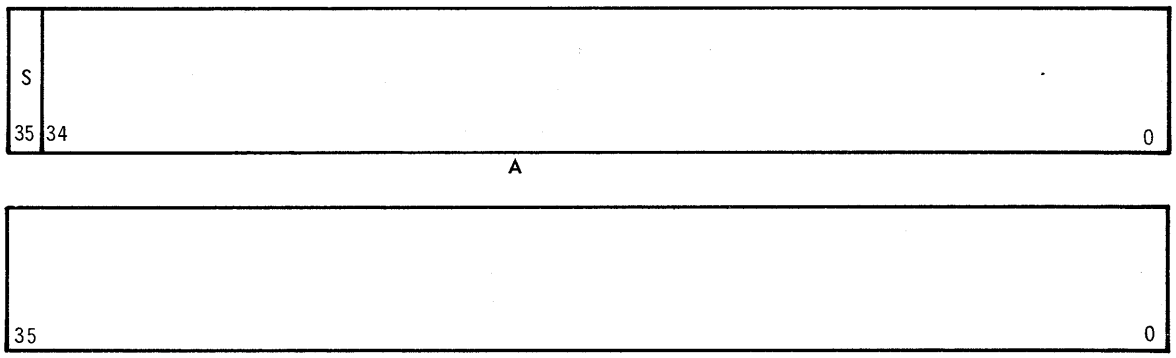


---

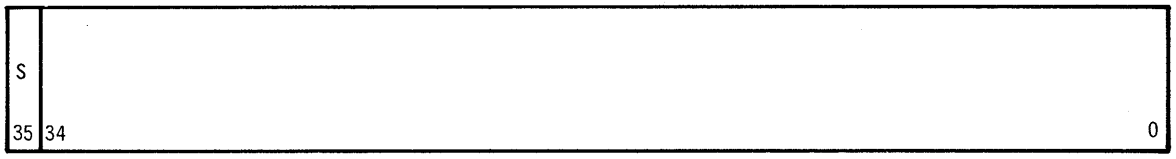
*Same as for corresponding PSR and PSRE fields.

**Format of utility storage limits register (SLRU) word is identical to SLR word format.

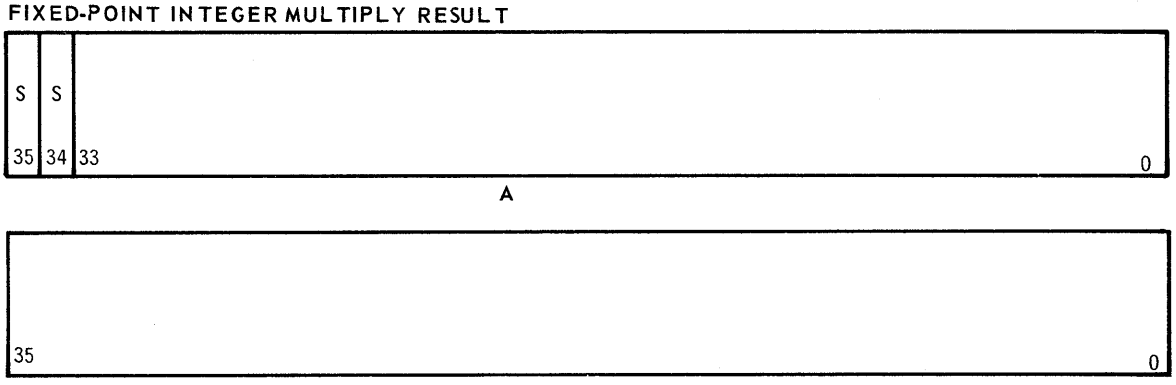
DOUBLE-PRECISION FIXED-POINT WORD



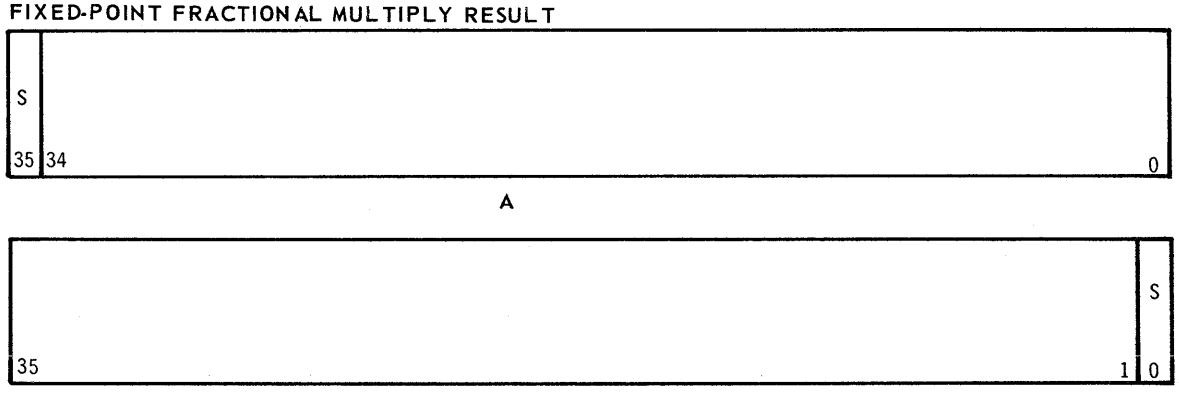
FIXED-POINT MULTIPLY SINGLE INTEGER RESULT



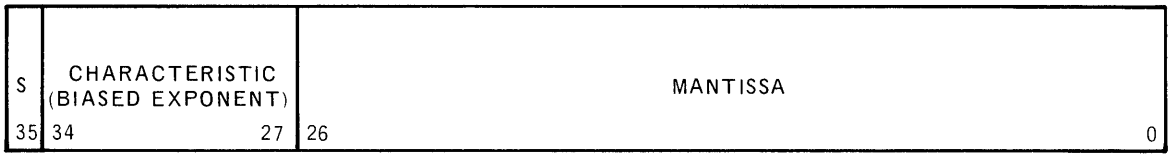
FIXED-POINT INTEGER MULTIPLY RESULT



FIXED-POINT FRACTIONAL MULTIPLY RESULT

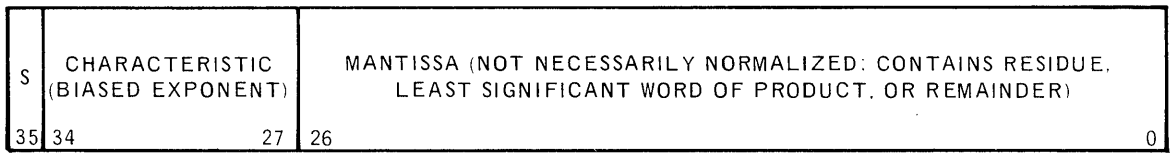


SINGLE-PRECISION FLOATING-POINT OPERAND



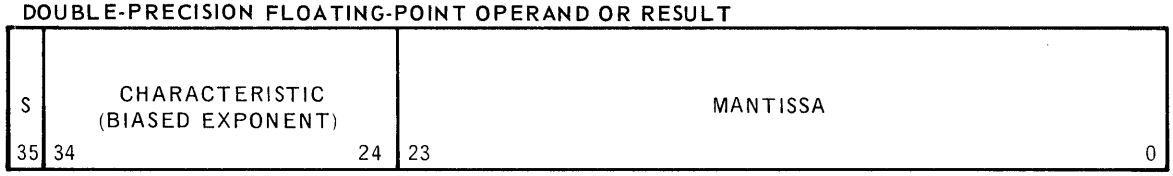
SINGLE-PRECISION FLOATING-POINT RESULT

A



A + 1

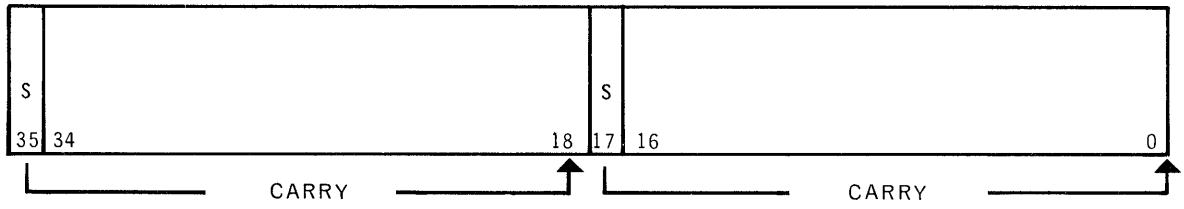
DOUBLE-PRECISION FLOATING-POINT OPERAND OR RESULT



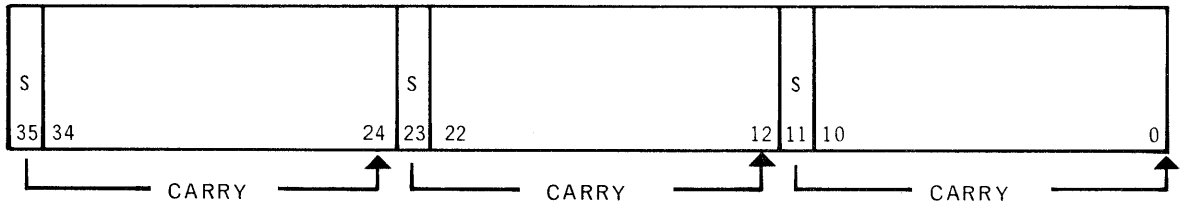
A

A + 1

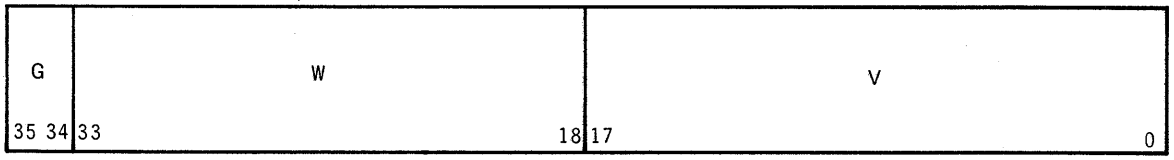
ADD HALVES WORD FORMAT



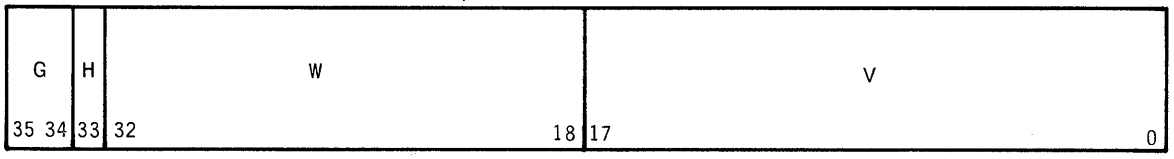
ADD THIRDS WORD FORMAT



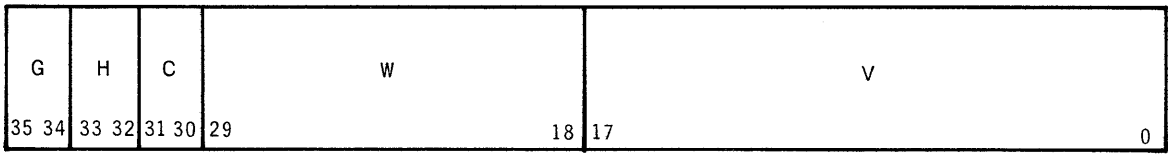
ISI ACCESS CONTROL WORD



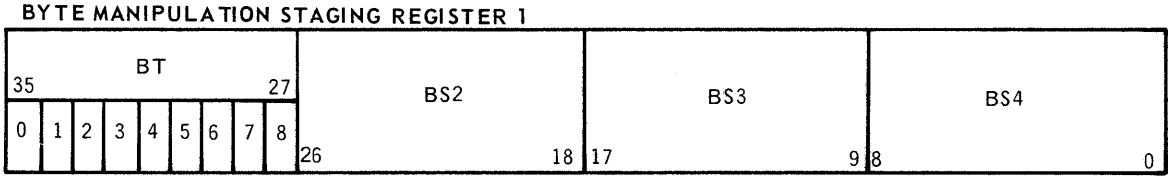
ESI ACCESS CONTROL WORD (half-word)



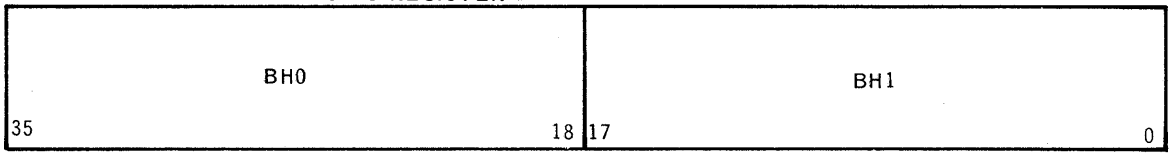
ESI ACCESS CONTROL WORD (quarter-word)



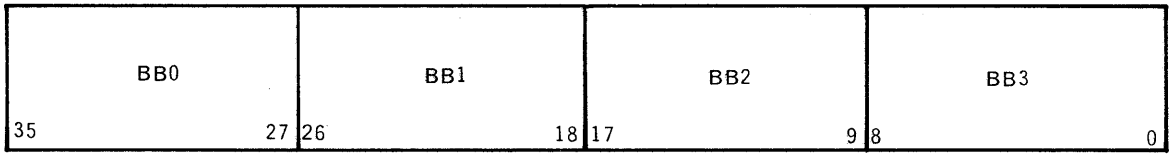
BYTE MANIPULATION STAGING REGISTER 1



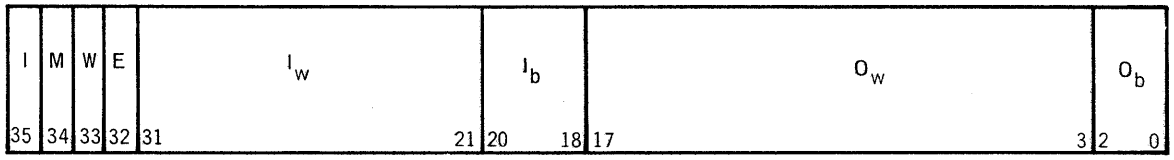
BYTE MANIPULATION STAGING REGISTER 2



BYTE MANIPULATION STAGING REGISTER 3



J REGISTER



APPENDIX C. INSTRUCTION REPERTOIRE BY FUNCTION CODE

FUNCTION CODE (OCTAL)		MNEMONIC	INSTRUCTION	DESCRIPTION*
f	i			
00	–	–	Invalid Code	Causes invalid instruction interrupt to address 241_8
01	0–15	S, SA	Store A	(A) → U
02	0–15	SN, SNA	Store Negative A	–(A) → U
03	0–15	SM, SMA	Store Magnitude A	|(A)| → U
04	0–15	S, SR	Store R	(R_a) → U
05	0–15	SZ	Store Zero	ZEROS → U
06	0–15	S, SX	Store X	(X_a) → U
07	00	SIA	Store Input Access Control Word	(A) → IACR; channel number per U_{5-0}
07	01	SOA	Store Output Access Control Word	(A) → OACR; channel number per U_{5-0}
07	02	SIP	Store Input Pointer Word	(A) → ICPR; channel number per U_{5-0}
07	03	SOP	Store Output Pointer Word	(A) → OCPR; channel number per U_{5-0}
07	04	LIA	Load Input Access Control Word	(IACR) → A channel number per U_{5-0}
07	05	LOA	Load Output Access Control Word	(OACR) → A channel number per U_{5-0}
07	06	LIP	Load Input Pointer Word	(ICPR) → A channel number per U_{5-0}
07	07	LOP	Load Output Pointer Word	(OCPR) → A channel number per U_{5-0}
07	10	LCB	Load Chain Base Register	If a = 0, $(U)_{14-0}$ → CBR; IOAU number 0 (for channel 0–23) If a = 1, $(U)_{14-0}$ → CBR; IOAU number 1 (channel 24–47)

*See Appendix A for notational conventions.

FUNCTION CODE (OCTAL)		MNEMONIC	INSTRUCTION	DESCRIPTION*
f	i			
07	11	LPI	Load Processor Interrupt Pointer	If a = 0, $(U)_{1-0}$ → processor interrupt pointer register of IOAU number 0 (for channels 0–23) If a = 1, $(U)_{1-0}$ → processor interrupt pointer register of IOAU number 1 (for channels 24–47)
07	12	LDJ	Load D – Bank Base and Jump	Transfer the portions of the MD word specified by the MDP plus the MD index specified by (Xa) to PSR, SLR; jump to address U (on new PSR, SLR)
07	13	LIJ	Load I – Bank Base and Jump	Transfer the portions of the MD word specified by MDP plus the MD index specified by (Xa) to PSR, SLR; jump to address U (on new PSR, SLR)
07	14	LPD (a = 0)	Load PSR Designators	U_{3-0} → PSR; Bit 0 = D4 Bit 1 = D5 Bit 2 = D8 Bit 3 = D10
07	14	SPD (a = 1)	Store PSR Designators	PSR D-bits → U
07	16	LBR (a = 0)	Load Breakpoint Register	(U) → Breakpoint Register
07	16	SJS (a = 1)	Store Jump Stack	(Jump History Stack) → U through U + 3 NOTE: The SJS instruction is noninterruptible
07	17	–	Invalid Code	Causes invalid instruction interrupt to address 241_8
10	0–17	L, LA	Load A	(U) → A
11	0–17	LN, LNA	Load Negative A	–(U) → A
12	0–17	LM, LMA	Load Magnitude A	|(U)| → A
13	0–17	LNMA	Load Negative Magnitude A	–|(U)| → A

*See Appendix A for notational conventions.

FUNCTION CODE (OCTAL) f	FUNCTION CODE (OCTAL) j	MNEMONIC	INSTRUCTION	DESCRIPTION*
14	0–17	A, AA	Add To A	(A) + (U) → A
15	0–17	AN, ANA	Add Negative To A	(A) − (U) → A
16	0–17	AM, AMA	Add Magnitude To A	(A) + |(U)| → A
17	0–17	ANM, ANMA	Add Negative Magnitude To A	(A) − |(U)| → A
20	0–17	AU	Add Upper	(A) + (U) → A + 1
21	0–17	ANU	Add Negative Upper	(A) − (U) → A + 1
22	0–15	BT	Block Transfer, Repeat	$(X_x + u) \rightarrow X_a + u$; repeat K times
23	0–17	L, LR	Load R	$(U) \rightarrow R_a$
24	0–17	A, AX	Add To X	$(X_a) + (U) \rightarrow X_a$
25	0–17	AN, ANX	Add Negative To X	$(X_a) - (U) \rightarrow X_a$
26	0–17	LXM	Load X Modifier	$(U) \rightarrow X_{a_{17-0}}$; $X_{a_{35-18}}$ unchanged
27	0–17	L, LX	Load X	$(U) \rightarrow X_a$
30	0–17	MI	Multiply Integer	(A) . (U) → A, A + 1
31	0–17	MSI	Multiply Single Integer	(A) . (U) → A
32	0–17	MF	Multiply Fractional	(A) . (U) → A, A + 1
33	00	BM	Byte Move	Transfer N bytes from source string to receiving string. Truncate or fill receiving string as required
33	01	BMT	Byte Move With Translate	Translate and transfer N bytes from source string to receiving string. Truncate or fill receiving string as required
33	02	BTT	Byte Translate and Test	Translate and test N bytes against (A); if not equal terminate instruction with J0 pointing unequal byte and $N \neq 0$
33	03	BTC	Byte Translate and Compare	Translate and compare N bytes from string E to M bytes from string F; terminate instruction on not equal or when both M and N have been reduced to zero; when: (A) > 0; string E > F (A) = 0; string E = F (A) < 0; string E < F

*See Appendix A for notational conventions.

FUNCTION CODE (OCTAL)		MNEMONIC	INSTRUCTION	DESCRIPTION*
f	i			
33	04	BC	Byte Compare	Compare N bytes from string E to M bytes from string F; terminate instruction on not equal or when both M and N are zero
33	05	BPD	Byte to Packed Decimal Convert	Convert N bytes in string E to packed decimal in string F
33	06	PDB	Packed Decimal to Byte Convert	Convert N packed decimal digits in string E to bytes in string F
33	07	EDIT	Edit	Edit byte string E and transfer to string F under the control of string G
33	10	BI	Byte to Binary Single Integer Convert	Convert N bytes in string E into a signed binary integer in register A
33	11	BDI	Byte to Binary Double Integer Convert	Convert N bytes in string E into a signed binary integer in registers A and A + 1
33	12	IB	Binary Single Integer to Byte Convert	Convert the binary integer in A to byte format and store in string E
33	13	DIB	Binary Double Integer to Byte Convert	Convert the binary integer in A and A + 1 to byte format and store in string E
33	14	BF	Byte to Single Floating Convert	Convert N bytes in string E into a single length floating point format in register A
33	15	BDF	Byte to Double Floating Convert	Convert N bytes in string E into a double length floating point format in registers A, A + 1
33	16	FB	Single Floating to Byte Convert	Convert the single length floating point number in A to byte format and store in string E
33	17	DFB	Double Floating to Byte Convert	Convert the double length floating point number in A and A + 1 to byte format and store in string E

*See Appendix A for notational conventions.

FUNCTION CODE (OCTAL)		MNEMONIC	INSTRUCTION	DESCRIPTION*
f	i			
34	0–17	DI	Divide Integer	(A, A + 1) ÷ (U) → A; REMAINDER → A + 1
35	0–17	DSF	Divide Single Fractional	(A) ÷ (U) → A + 1
36	0–17	DF	Divide Fractional	(A, A + 1) ÷ (U) → A; REMAINDER → A + 1
37	00	QB	Quarter-Word Byte to Binary Compress	Discard $(A)_{35}$, $(A)_{26}$, $(A)_{17}$, and $(A)_{8}$; place the remaining bits in A_{31-0}; $(A)_{31} \rightarrow A_{35-32}$.
37	01	BQ	Binary to Quarter-Word Byte Extend	Discard $(A)_{35-32}$; place the remaining bits in A_{34-27}, A_{25-18}, A_{16-9}, and A_{7-0}; zero fill A_{35}, A_{26}, A_{17}, and A_{8}.
37	02	QBH	Quarter-Word Byte to Binary Halves Compress	Discard $(A)_{35}$, $(A)_{26}$, $(A)_{17}$, and $(A)_{8}$; place the remaining bits in A_{33-18} and A_{15-0}; $(A)_{33} \rightarrow A_{35-34}$; $(A)_{15} \rightarrow A_{17-16}$.
37	03	BHQ	Binary Halves to Quarter-Word Byte Extend	Discard $(A)_{35-34}$ and $(A)_{17-16}$; place the remaining bits in A_{34-27}, A_{25-18}, A_{16-9}, and A_{7-0}; zero fill A_{35}, A_{26}, A_{17}, and A_{8}.
37	04	QDB	Quarter-Word Byte to Double Binary Compress	Discard A_{35}, A_{26}, A_{17}, A_{8}, $A+1_{35}$, $A+1_{26}$, $A+1_{17}$, and $A+1_{8}$; place the remaining bits in A_{27-0} and A + 1; $(A)_{27} \rightarrow A_{35-28}$.
37	05	DBQ	Double Binary to Quarter-Word Byte Extend	Discard $(A)_{35-28}$; place the remaining bits from A and A + 1 in A_{34-27}, A_{25-18}, A_{16-9}, A_{7-0}, $A+1_{34-27}$, $A+1_{25-18}$, $A+1_{16-9}$, and $A+1_{7-0}$; zero fill A_{35}, A_{26}, A_{17}, A_{8}, $A+1_{35}$, $A+1_{26}$, $A+1_{17}$, and $A+1_{8}$.
37	06	BA	Byte Add	Add the N bytes in string E to the M bytes in string F and place the results in string G
37	07	BAN	Byte Add Negative	Subtract the N bytes in string E from the M bytes in string F and place the result in string G
37	10–17	–	Invalid Code	Causes invalid instruction interrupt to address 241_{8}
40	0–17	OR	Logical OR	(A) OR (U) → A + 1
41	0–17	XOR	Logical Exclusive	(A) XOR (U) → A + 1

*See Appendix A for notational conventions.

FUNCTION CODE (OCTAL) f	i	MNEMONIC	INSTRUCTION	DESCRIPTION*
42	0–17	AND	Logical AND	(A) AND (U) → A + 1
43	0–17	MLU	Masked Load Upper	[(U) AND (R2)] OR [(A) AND (R2)] → A + 1
44	0–17	TEP	Test Even Parity	Skip NI if (U) AND (A) has even parity
45	0–17	TOP	Test Odd Parity	Skip NI if (U) AND (A) has odd parity
46	0–17	LXI	Load X Increment	(U) → $(X_a)_{35-18}$; $(X_a)_{17-0}$ unchanged
47	0–17	TLEM	Test Less Than or Equal to Modifier	Skip NI if (U) ≤ $(X_a)_{17-0}$;
		TNGM	Test Not Greater Than Modifier	always $(X_a)_{17-0}$ + $(X_a)_{35-18}$ → $X_{a_{17-0}}$
50	0–17	TZ	Test Zero	Skip NI if (U) = ±0
51	0–17	TNZ	Test Nonzero	Skip NI if (U) ≠ ±0
52	0–17	TE	Test Equal	Skip NI if (U) = (A)
53	0–17	TNE	Test Not Equal	Skip NI if (U) ≠ (A)
54	0–17	TLE	Test Less Than or Equal	Skip NI if (U) ≤ (A)
		TNG	Test Not Greater	
55	0–17	TG	Test Greater	Skip NI if (U) > (A)
56	0–17	TW	Test Within Range	Skip NI if (A) < (U) ≤ (A + 1)
57	0–17	TNW	Test Not Within Range	Skip NI if (U) ≤ (A) or (U) > (A + 1)
60	0–17	TP	Test Positive	Skip NI if $(U)_{35} = 0$
61	0–17	TN	Test Negative	Skip NI if $(U)_{35} = 1$
62	0–17	SE	Search Equal	Skip NI if (U) = (A), else repeat
63	0–17	SNE	Search Not Equal	Skip NI (U) ≠ (A), else repeat
64	0–17	SLE	Search Less Than or Equal	Skip NI if (U) ≤ (A), else repeat
		SNG	Search Not Greater	
65	0–17	SG	Search Greater	Skip NI if (U) > (A), else repeat
66	0–17	SW	Search Within Range	Skip NI if (A) < (U) ≤ (A + 1), else repeat

*See Appendix A for notational conventions

FUNCTION CODE (OCTAL)		MNEMONIC	INSTRUCTION	DESCRIPTION*
f	j			
67	0–17	SNW	Search Not Within Range	Skip NI if (U) $\leq$ (A) or (U) > (A + 1), else repeat
70	**	JGD	Jump Greater and Decrement	Jump to U if (Control Register $_{ja}$) > 0; go to NI if (Control Register$_{ja}$) $\leq$ 0; always (Control Register$_{ja}$) $-1 \rightarrow$ Control Register $_{ja}$
71	00	MSE	Mask Search Equal	Skip NI if (U) AND (R2) = (A) AND (R2), else repeat
71	01	MSNE	Mask Search Not Equal	Skip NI if (U) AND (R2) $\neq$ (A) AND (R2), else repeat
71	02	MSLE	Mask Search Less Than or Equal	Skip NI if (U) AND (R2) $\leq$ (A) AND (R2), else repeat
		MSNG	Mask Search Not Greater	
71	03	MSG	Mask Search Greater	Skip NI if (U) AND (R2) > (A) AND (R2), else repeat
71	04	MSW	Masked Search Within Range	Skip NI if (A) AND (R2) < (U) AND (R2) $\leq$ (A + 1) AND (R2), else repeat
71	05	MSNW	Masked Search Not Within Range	Skip NI if (U) AND (R2) $\leq$ (A) AND (R2) or (U) AND (R2) > (A + 1) AND (R2), else repeat
71	06	MASL	Masked Alphanumeric Search Less Than or Equal	Skip NI if (U) AND (R2) $\leq$ (A) AND (R2), else repeat
71	07	MASG	Masked Alphanumeric Search Greater	Skip NI if (U) AND (R2) > (A) AND (R2), else repeat
71	10	DA	Double Precision Fixed-Point Add	(A, A+1) + (U, U+1) $\rightarrow$ A, A+1
71	11	DAN	Double Precision Fixed-Point Add Negative	(A, A+1) – (U, U+1) $\rightarrow$ A, A+1
71	12	DS	Double Store A	(A, A+1) $\rightarrow$ U, U+1
71	13	DL	Double Load A	(U, U+1) $\rightarrow$ A, A+1
71	14	DLN	Double Load Negative A	–(U, U+1) $\rightarrow$ A, A+1
71	15	DLM	Double Load Magnitude A	|(U, U+1)| $\rightarrow$ A, A+1
71	16	DJZ	Double Precision Jump Zero	Jump to U if (A, A+1) = $\pm$0; go to NI if (A, A+1) $\neq \pm$0
71	17	DTE	Double Precision Test Equal	Skip NI if (U, U+1) = (A, A+1)
72	00	–	Invalid Code	Causes invalid instruction interrupt to address 241_8

*See Appendix A for notational conventions

**The a and j fields together serve to specify any one of the 112 control registers.

FUNCTION CODE (OCTAL)		MNEMONIC	INSTRUCTION	DESCRIPTION*
f	i			
72	01	SLJ	Store Location and Jump	(P) – BASE ADDRESS MODIFIER [BIX/BI or BDX/BD] → U_{17-0}; jump to U+1
72	02	JPS	Jump Positive and Shift	Jump to U if $(A)_{35} = 0$; go to NI if $(A)_{35} = 1$; always shift (A) left circularly one bit position
72	03	JNS	Jump Negative and Shift	Jump to U if $(A)_{35-1} = 1$; go to NI if $(A)_{35-1} = 0$; always shift (A) left circularly one bit position
72	04	AH	Add Halves	$(A)_{35-18} + (U)_{35-18} \rightarrow (A)_{35-18}$; $(A)_{17-0} + (U)_{17-0} \rightarrow A_{17-0}$
72	05	ANH	Add Negative	$(A)_{35-18} - (U)_{35-18} \rightarrow (A)_{35-18}$; $(A)_{17-0} - (U)_{17-0} \rightarrow A_{17-0}$
72	06	AT	Add Thirds	$(A)_{35-24} + (U)_{35-24} \rightarrow A_{35-24}$; $(A)_{23-12} + (U)_{23-12} \rightarrow A_{23-12}$; $(A)_{11-0} + (U)_{11-0} \rightarrow A_{11-0}$
72	07	ANT	Add Negative Thirds	$(A)_{35-24} - (U)_{35-24} \rightarrow A_{35-24}$; $(A)_{23-12} - (U)_{23-12} \rightarrow A_{23-12}$; $(A)_{11-0} - (U)_{11-0} \rightarrow A_{11-0}$
72	10	EX	Execute	Execute the instruction at U
72	11	ER	Executive Return	Causes executive return interrupt to address 242_8
72	12	–	Invalid Code	Causes invalid instruction interrupt to address 241_8
72	13	PAIJ	Prevent All I/O Interrupts and Jump	Prevent all I/O interrupts and jump to U
72	14	SCN	Store Channel Number	If a = 0, number of the interrupting I/O channel → U_{3-0}; CAU number → U_{5-4} If a = 1, number of the interrupting I/O channel → U_{3-0}; CAU number → U_{5-4}

*See Appendix A for notational conventions.

FUNCTION CODE (OCTAL)		MNEMONIC	INSTRUCTION	DESCRIPTION*
f	i			
				If a = 2, number of the interrupting I/O channel → U_{5-0}; CAU number → U_{14-12} If a = 3, number of the interrupting I/O channel → U_{5-0}; CAU number → U_{14-12}
72	15	LPS (a = 0)	Load Processor State Register	(U) → PSR
72	15	LMP (a = 1)	Load Main Processor State Register	(U, U+1) → PSR, PSRE
72	15	LUP (a = 2)	Load Utility Processor State Register	(U, U+1) → PSRU, PSRUE
72	16	LSL (a = 0)	Load Storage Limits Register	(U) → SLR
72	16	LUS (a = 1)	Load Utility Storage Limits Register	(U) → SLRU
72	16	SL (a = 2)	Store Storage Limits Register	(SLR) → U
72	16	SUL (a = 3)	Store Utility Storage Limits Register	(SLRU) → U
72	17	–	Invalid Code	Causes invalid instruction interrupt to address 241_8
73	00	SSC	Single Shift Circular	Shift (A) right circularly U places
73	01	DSC	Double Shift Circular	Shift (A, A+1) right circularly U places
73	02	SSL	Single Shift Logical	Shift (A) right U places; zerofill
73	03	DSL	Double Shift Logical	Shift (A, A+1) right U places; zerofill
73	04	SSA	Single Shift Algebraic	Shift (A) right U places; signfill
73	05	DSA	Double Shift Algebraic	Shift (A, A+1) right U places; signfill

*See Appendix A for notational conventions.

FUNCTION CODE (OCTAL)		MNEMONIC	INSTRUCTION	DESCRIPTION*
f	i			
73	06	LSC	Load Shift and Count	(U) → A, shift (A) left circularly until $(A)_{35} \neq (A)_{34}$; NUMBER OF SHIFTS → A+1
73	07	DLSC	Double Load Shift and Count	(U, U+1) → A, A+1; shift (A, A+1) left circularly until $(A, A+1)_{71} \neq (A, A+1)_{70}$; NUMBER OF SHIFTS → A+2
73	10	LSSC	Left Single Shift Circular	Shift (A) left circularly U places
73	11	LDSC	Left Double Shift Circular	Shift (A, A+1) left circularly U places
73	12	LSSL	Left Single Shift Logical	Shift (A) left U places; zerofill
73	13	LDSL	Left Double Shift Logical	Shift (A, A+1) left U places; zerofill
73	14	III (a=0 through 5)	Initiate Interprocessor-Interrupt	Initiate interprocessor-interrupt per a; a = 0: Interrupt CAU Number 0 a = 1: Interrupt CAU Number 1 a = 2: Interrupt CAU Number 2 a = 3: Interrupt CAU Number 3 a = 4: Interrupt CAU Number 4 a = 5: Interrupt CAU Number 5
73	14	EDC ($a=11_8$)	Enable Day Clock	Enable day clock
73	14	DDC ($a=12_8$)	Disable Day Clock	Disable day clock
73	14	SEC ($a=13_8$)	Set and Enable Storage Reference Counters	
73	14	EC ($a=14_8$)	Enable Storage Reference Counters	
73	15	SIL	Select Interrupt Locations	$(U)_{8-0}$ → MSR

*See Appendix A for notational conventions.

FUNCTION CODE (OCTAL)		MNEMONIC	INSTRUCTION	DESCRIPTION*
f	i			
73	16	LCR (a=0)	Load Channel Select Register	$(U)_{5-0} \rightarrow$ CSR; if $(U)_9 = 1$, condition channel for back-to-back data transfer
		LLA (a=1)	Load Last Address Register	$(U)_{8-0} \rightarrow$ LAR
73	17	TS (a=0)	Test and Set	If $(U)_{30} = 1$, interrupt; if $(U)_{30} = 0$, take the next instruction. Always set $(U)_{35-30}$ to 01_8
73	17	TSS (a = 1)	Test and Set and Skip	If $(U)_{30} = 0$, skip; if $(U)_{30} = 1$, take next instruction. Always set $(U)_{35-30}$ to 01_8
73	17	TCS (a = 2)	Test and Clear and Skip	If $(U)_{30} = 0$, take next instruction; if $(U)_{30} = 1$, skip. Always clear $(U)_{35-30}$
74	00	JZ	Jump Zero	Jump to U if $(A) = \pm 0$; go to NI if $(A) \neq \pm 0$
74	01	JNZ	Jump Nonzero	Jump to U if $(A) \neq \pm 0$; go to NI if $(A) = \pm 0$
74	02	JP	Jump Positive	Jump to U if $(A)_{35} = 0$; go to NI if $(A)_{35} = 1$
74	03	JN	Jump Negative	Jump to U if $(A)_{35} = 1$; go to NI if $(A)_{35} = 0$
74	04	JK J	Jump Keys Jump	Jump to U if a=0 or if a= lit select jump indicator; go to NI if neither is true
74	05	HKJ HJ	Halt Keys and Jump Halt Jump	Stop if a=0 or if [a AND lit select stop indicators] $\neq 0$; on restart or continuation, jump to U
74	06	NOP	No Operation	Proceed to next instruction
74	07	AAIJ	Allow All I/O Interrupts and Jump	Allow all I/O interrupts and jump to U
74	10	JNB	Jump No Low Bit	Jump to U if $(A)_0 = 0$; go to NI if $(A)_0 = 1$
74	11	JB	Jump Low Bit	Jump to U if $(A)_0 = 1$; go to NI if $(A)_0 = 0$

*See Appendix A for notational conventions

FUNCTION CODE (OCTAL)		MNEMONIC	INSTRUCTION	DESCRIPTION*
f	i			
74	12	JMGI	Jump Modifier Greater and Increment	Jump to U if $(X_a)_{17-0} > 0$; go to NI if $(X_a)_{17-0} \leq 0$; always $(X_a)_{17-0} + (X_a)_{35-0} \rightarrow X_{a_{17-0}}$
74	13	LMJ	Load Modifier and Jump	(P) – BASE ADDRESS MODIFIER [BIX/BI or BDX/BD] $\rightarrow (X_a)_{17-0}$; Jump to U
74	14	JO	Jump Overflow	Jump to U if D1 of PSR = 1; go to NI if D1 = 0
74	15	JNO	Jump No Overflow	Jump to U if D1 of PSR = 0; go to NI if D1 = 1
74	16	JC	Jump Carry	Jump to U if D0 of PSR = 1; go to NI if D0 = 0
74	17	JNC	Jump No Carry	Jump to U if D0 of PSR = 0; go to NI if D0 = 1
75	00	LIC	Load Input Channel	For channel [a OR CSR]: (U) → IACR; set input active; clear input monitor
75	01	LICM	Load Output Channel and Monitor	For channel [a OR CSR]: (U) → IACR; set input active; set input monitor
75	02	JIC	Jump Input Channel Busy	Jump to U if input active is set for channel [a OR CSR]; go to NI if input active is clear
75	03	DIC	Disconnect Input Channel	For channel [a OR CSR]: clear input active; clear input monitor
75	04	LOC	Load Output Channel	For channel [a OR CSR]: (U) → OACR; set output active; clear output monitor; clear external function (ISI only)
75	05	LOCM	Load Input Channel and Monitor	For channel [a OR CSR]: (U) → OACR; set output active; set output monitor; clear external function (ISI only)
75	06	JOC	Jump Output Channel Busy	Jump to U if output active is set for channel [a OR CSR]; go to NI if output active is clear
75	07	DOC	Disconnect Output Channel	For channel [a OR CSR]: clear output active; clear output monitor; clear external function

*See Appendix A for notational conventions

FUNCTION CODE (OCTAL)		MNEMONIC	INSTRUCTION	DESCRIPTION*
f	i			
75	10	LFC	Load Function in Channel	For channel [a OR CSR]: (U) → OACR; set output active (ISI only), external function, and force external function; clear output monitor (ISI only)
75	11	LFCM	Load Function in Channel and Monitor	For channel [a OR CSR]: (U) → OACR; set output active (ISI only), external function, force external function, and output monitor (ISI only)
75	12	JFC	Jump on Function in Channel	Jump to U if force external function is set for channel [a OR CSR]; go to NI if force external function is clear
75	13	–	Invalid Code	Causes invalid instruction interrupt to address 241_8
75	14	AACI	Allow All Channel External Interrupts	Allow all external interrupts
75	15	PACI	Prevent All Channel External Interrupts	Prevent all external interrupts
75	16	ACI	Allow Channel Interrupts	If a = 0, allow interrupts on channels 23–0 specified by one bits in $(U)_{23-0}$ If a = 1, allow interrupts on Channels 47–24 specified by one bits in $(U)_{23-0}$
75	17	PCI	Prevent Channel Interrupts	If a = 0, prevent interrupts on Channels 23–0 specified by one bits bits in $(U)_{23-0}$ If a = 1, prevent interrupts on Channels 47–24 specified by one bits in $(U)_{23-0}$
76	00	FA	Floating Add	(A) + (U) → A; REMAINDER → A+1
76	01	FAN	Floating Add Negative	(A) – (U) → A; REMAINDER → A+1
76	02	FM	Floating Multiply	(A) . (U) → A; REMAINDER → A+1
76	03	FD	Floating Divide	(A) ÷ (U) → A; REMAINDER → A+1

*See Appendix A for notational conventions

FUNCTION CODE (OCTAL)		MNEMONIC	INSTRUCTION	DESCRIPTION*
f	j			
76	04	LUF	Load and Unpack Floating	$\|(U)\|_{34-27} \rightarrow (A)_{7-0}$, zero-fill; $(U)_{26-0} \rightarrow A+1_{26-0}$ signfill
76	05	LCF	Load and Convert To Floating	$(U)_{35} \rightarrow (A+1)_{35}$; [NORMALIZED $(U)]_{26-0} \rightarrow A+1_{26-0}$; if $(U)_{35} = 0$, $(A)_{7-0} \pm$ NORMALIZING COUNT $\rightarrow (A+1)_{34-27}$; if $(U)_{35} = 1$, ones complement of $[(A)_{7-0} \pm$ NORMALIZING COUNT$] \rightarrow A+1_{34-27}$
76	06	MCDU	Magnitude of Characteristic Difference To Upper	$\|\|(A)\|_{35-27} - \|(U)_{35-27}\| \rightarrow (A+1)_{7-0}$; ZEROS $\rightarrow (A+1)_{35-8}$
76	07	CDU	Characteristic Difference To Upper	$\|(A)\|_{35-27} - \|(U)_{35-27}\| \rightarrow (A+1)_{7-0}$; SIGN BITS $\rightarrow (A+1)_{35-8}$
76	10	DFA	Double Precision Floating Add	$(A, A+1) + (U, U+1) \rightarrow A, A+1$
76	11	DFAN	Double Precision Floating Add Negative	$(A, A+1) - (U, U+1) \rightarrow A, A+1$
76	12	DFM	Double Precision Floating Multiply	$(A, A+1) \cdot (U, U+1) \rightarrow A, A+1$
76	13	DFD	Double Precision Floating Divide	$(A, A+1) \div (U, U+1) \rightarrow A, A+1$
76	14	DFU	Double Load and Unpack Floating	$\|(U, U+1)_{70-60}\| \rightarrow (A)_{10-0}$; $(U, U+1)_{59-36} \rightarrow (A+1)_{23-0}$; $(U, U+1)_{35-0} \rightarrow A+2$
76	15	DLCF	Double Load and Convert To Floating	$(U)_{35} \rightarrow A+1_{35}$; [NORMALIZED $(U, U+1)]_{59-0} \rightarrow A+1_{23-0}$ and A+2; If $(U)_{35}$, $(A)_{10-0} \pm$ normalizing count $\rightarrow A+1_{34-24}$ If $(U)_{35} = 1$, 1's complement of $[(A)_{10-0} \pm$ normalizing count$] \rightarrow A+1_{34-24}$

*See Appendix A for notational conventions

FUNCTION CODE (OCTAL)		MNEMONIC	INSTRUCTION	DESCRIPTION*
f	i			
76	16	FEL	Floating Expand and Load	If $(U)_{35} = 0$, $(U)_{35-27} + 1600_8 \rightarrow A_{35-24}$; If $(U)_{35} = 1$, $(U)_{35-27} - 1600_8 \rightarrow A_{35-24}$; $(U)_{26-3} \rightarrow A_{23-0}$; $(U)_{2-0} \rightarrow A+1_{35-33}$; $(U)_{35} \rightarrow A+1_{32-0}$
76	17	FCL	Floating Compress and Load	If $(U)_{35} = 0$, $(U)_{35-24} - 1600_8 \rightarrow A_{35-27}$; If $(U)_{35} = 1$, $(U)_{35-24} + 1600_8 \rightarrow A_{35-27}$; $(U)_{23-0} \rightarrow A_{26-3}$ $(U+1)_{35-33} \rightarrow A_{2-0}$
77	0–17	–	Invalid Code	Causes invalid instruction interrupt to address 241_8

*See Appendix A for notational conventions

UP-7841