# UNIVAC® 1108

MULTI-PROCESSOR SYSTEM

SYSTEM DESCRIPTION

# CONTENTS

FIGURES

# 1. INTRODUCTION

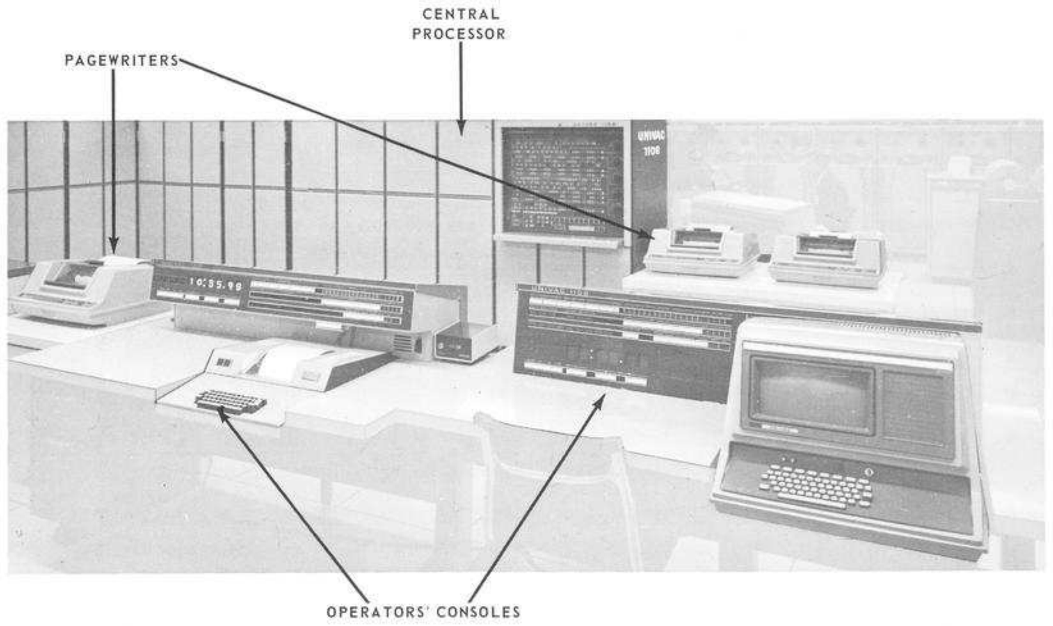


*Figure 1–1. The Central Processor and Operator's Console*

The UNIVAC 1108 System – the logical, program compatible successor to the UNIVAC 1107 Computer – is an integration of system-oriented hardware design, imaginative development, and programming technology. The result is a system that effectively knows no application boundaries. It is equally effectual in real-time, scientific, or data processing environments, and is capable of adjusting dynamically to any one or a mixture of these environments.

All system operations are coordinated and controlled by a versatile executive system having full real-time, multiprogramming, and multiprocessing capabilities, but possessing the simplicity of a monitor system.

The following description introduces the UNIVAC 1108 Multi-Processor System organization, hardware components, and programming system.

# 2. SYSTEM DESCRIPTION

## 2.1. GENERAL

The UNIVAC 1108 System is a general purpose, high performance unit- and multi-processor system, making good use of the latest advances in computer design, systems organization, and programming technology. Its modular structure permits the selection of systems components to fulfill most efficiently the speed and capacity requirements for applications ranging from a basic job-shop system to the comprehensive public utility computing complex.

As the workload increases, this modularity also enables the addition of input/output subsystems and main storage and even additional processors to provide the full multi-processor system. Among the principal features of the 1108 System are:

- Common resource systems organization
- Equality among multiple UNIVAC 1108 Central Processors
- Multiple input/output controllers
- Large, modular, parity-checked, high speed main storage
- Overlapped and interleaved main storage access
- Redundancy among systems components
- Program address relocation
- Storage protection
- Partial-word addressability in 6, 9, 12, and 18-bit portions as well as full-word (36 bits) and double-word (72 bits) addressing.
- High speed, random access, auxiliary storage
- Privileged mode for the Executive system
- Guard mode for user programs.

## 2.2. SYSTEM COMPONENTS

The UNIVAC 1108 System is organized to allow multiple processors to perform, a number of tasks simultaneously under the direction of a common Executive control system. A multi-processor system requires a much higher degree of modularity in organization than does a unit-processor system. It must be divisible into individual logical components with the following properties:

- Each system component must have more than one access path
- Priority logic must resolve possible access conflicts
- The failure of any individual component must not prevent continued operation of the system
- System components must be logically removable for servicing without disabling the system.

The system is constructed of six types of components:

- Central Processors
- Input/Output Controllers
- Main Storage Modules
- Auxiliary Storage Subsystems
- System Interconnection Components
- Peripheral Subsystems

### 2.2.1. Central Processor

Each processor can perform all functions required for the execution of instructions including arithmetic, input/output, and Executive control. In a multi-processor configuration, all processors are equal – the test of a true multi-processor system. Included in each processor is its own set of 125-nanosecond integrated circuit control registers providing multiple accumulators, index registers, input/output access control registers, and special-use registers.

### 2.2.2. Input/Output Controller

The Input/Output Controller (Figure 2–1) is an independent processor utilized in multi-processor systems to expand the input/output capabilities of the system. It includes:

- Up to sixteen high speed input/output channels
- Independent access to main storage
- Data chaining
- Sixteen pointer registers
- Sixty-four external function and ISI data Access Control Registers
- 192 communications Access Control Registers
- An optional 256 additional communications Access Control Registers.

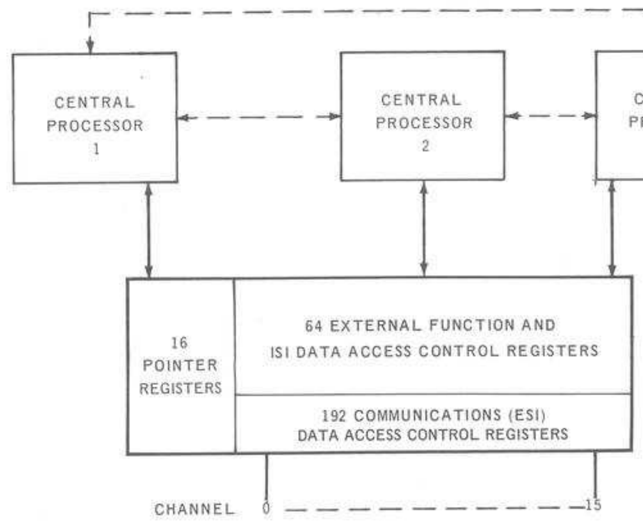

CENTRAL PROCESSOR 1
CENTRAL PROCESSOR 2
CENTRAL PROCESSOR 3
16 POINTER REGISTERS
64 EXTERNAL FUNCTION AND ISI DATA ACCESS CONTROL REGISTERS
192 COMMUNICATIONS (ESI) DATA ACCESS CONTROL REGISTERS
CHANNEL 0
15

*Figure 2–1. The Input/Output Controller*

### 2.2.3. Main Storage

The main storage of the UNIVAC 1108 System is expandable in 65,536-word increments up to a maximum of 262,144 thirty-six bit words. The main storage read/restore cycle time is 750 nanoseconds. Up to four logical banks for instruction/data reference overlapping provide the capability for an effective cycle time of 375 nanoseconds. In addition, two-way interleaving of storage modules is provided to reduce the probability of access conflicts.

### 2.2.4. Auxiliary Storage

The auxiliary magnetic drum storage subsystems are an integral part of each UNIVAC 1108 System. Up to eight FH-432 or FH-1782 magnetic drums, or any combination of the two types, may be attached to one or two control units. Both types of drum can transfer data at 1,440,000 characters per second. The average access time of the FH-432 is 4.3 milliseconds; that of the FH-1782 is 17 milliseconds.

### 2.2.5. Interconnection Components

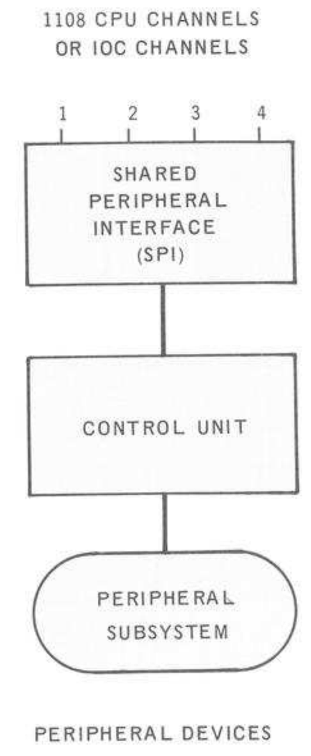
It is an essential characteristic of multi-processor systems that there must be provision for sharing of all of main storage and all I/O subsystems by all processors in the system. This sharing must be on the basis of both established priorities and reactive priorities that may change during processing. In the 1108 System, Multiple Module Access Units (MMA) provide this access to the storage modules by several processors, and the Shared Peripheral Interface (SPI) similarly enables the sharing of I/O subsystems.

#### 2.2.5.1. Shared Peripheral Interface

The Shared Peripheral Interface (SPI) controls the access of up to four input/output channels to units in a shared peripheral subsystem (see Figure 2–2). Access to shared peripheral subsystems is determined primarily by time of request. If two requests are made simultaneously, the Central Processor or Input/Output Controller on the lower numbered SPI Input/Output channel receives priority. In case of a busy condition, or a priority conflict, the Executive automatically stacks the request until the SPI channel is available. First-level queuing is handled in the SPI itself. The Executive stacks longer queues and keeps track of the number of I/O function requests outstanding.

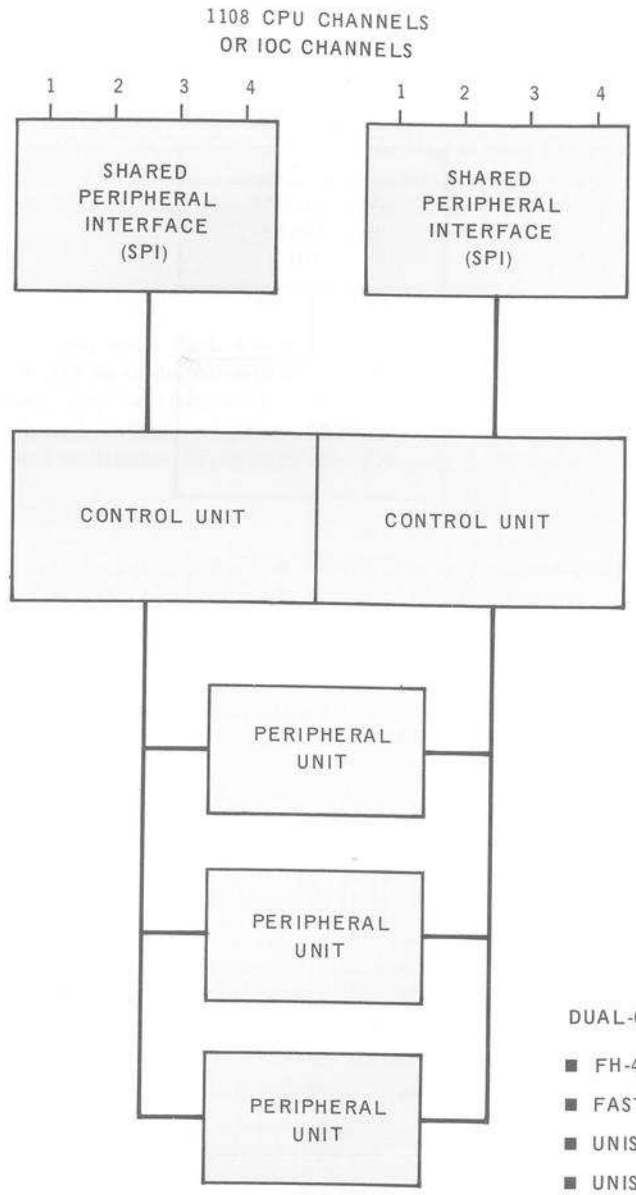
Input/output subsystems may be either single- or dual-channel, as is illustrated in Figures 2–2 and 2–3. Single-channel subsystems perform one I/O operation at a time and therefore require one control unit, one I/O channel, and, in multiprocessor systems, only one SPI. Dual channel subsystems can execute two operations simultaneously using different peripheral units in the subsystem. Both of these operations may originate in the same processor or they may come from different processors.

Different processors can be connected to such a dual-channel subsystem through two SPI Units. Two input/output channels from each Processor or Input/Output Controller take separate paths. The failure of an SPI or one of the pair of Control Units affects only one of the two paths to a peripheral subsystem. Therefore, all peripheral units are still accessible through the second SPI and Control Unit.



- FH-432/1782 Drums
- FASTRAND Mass Storage
- UNISERVO VI C Magnetic Tape Units
- UNISERVO VIII C Magnetic Tape Units
- Card Read/Punch
- Printers
- Communications Terminal Module Controller
- Word Terminal Synchronous
- 1004 Subsystem

*Figure 2–2. Shared Peripheral Interface (SPI), Single-Channel Subsystem*



DUAL-CHANNEL SUBSYSTEMS

- FH-432/1782 Drums
- FASTRAND Mass Storage
- UNISERVO VIII C Tape Units
- UNISERVO VI C Tape Units

*Figure 2–3. Shared Peripheral Interface (SPI), Dual-Channel Subsystem*

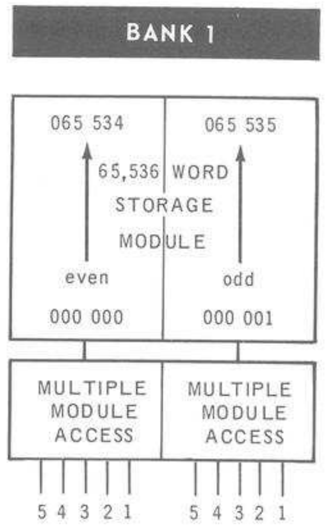


*Figure 2–4. Multiple Module Access Unit*

### 2.2.5.2. Multiple Module Access Unit

The Multiple Module Access Units (MMA) allow the sharing of individual storage modules by up to three Central Processors and two Input/Output Controllers on a fixed priority basis. (See Figure 2–4.)

The MMA recognizes the storage access requests on a priority basis with the lower channel retaining the higher priority. Input/Output Controllers which require higher priority are therefore connected to the lower numbered interfaces. Upon recognition of a storage access request the MMA connects the address lines, the CPU or IOC data lines, and the write control signals to the storage module. The MMA then sends the storage acknowledgement back to the recognized processor and provides the drive necessary to transfer the data to the processor requesting it.

### 2.2.5.3. Availability Control Unit

Because of the system availability requirement the multi-processor system must have a means for partitioning the system for specific jobs or for maintenance purposes. The Availability Control Unit (ACU) performs this and related functions as follows:

- Partitions the multiprocessor system hardware into independent systems.
- Takes units off-line for maintenance without disrupting operation of the rest of the system.
- Protects main storage in event of a power failure in the CPU or IOC.
- Automatically initiates a recovery sequence after a failure.

The ACU partitions the hardware into specific configurations by disabling and enabling the interface between units. It can set up as many as three logically independent configurations which run concurrently under control of the Executive system. The possible configurations can be pre-specified for a given site. At the same time the ACU can take units off-line for maintenance.

The ACU is an independent unit with its own power supply logically situated between the peripheral subsystems, the central processors, input/output controllers, and main storage. (See Figure 2–7.) It can interface with three Central Processors through one I/O channel of each, two IOC's, four banks of main storage, and six multiple-access peripheral subsystems. Additional peripheral subsystems, to a maximum of 24, can be added in groups of six. The ACU includes a control panel, physically located at the operator's console, that indicates all partitioning currently in effect and also shows which units are off line. It also has manual controls to switch units on or off line.

The automatic recovery sequence is based on a resettable system timer in the ACU. The period of this timer can be set to times varying from one to fifteen seconds. Unless the Executive system resets this timer within its period, the ACU assumes that a catastrophic malfunction has occurred and it initiates an automatic recovery sequence. The processor can then interrogate the ACU to determine which units are on line and available for use.

### 2.2.6. Operator's Control Console

The UNIVAC 1108 Operator's Control Console subsystem is a free-standing input/output device for directing and monitoring the operation of the CPU. A multi-processor configuration includes one console subsystem for each CPU. The various activities of the system can be apportioned among the available consoles so that the total system will be utilized to best advantage. The console is always connected to input/output channel 15.

The basic Control Console includes the following components:

- Keyboard and CRT Display

  The keyboard and CRT display enable the operator to monitor the performance of the system. The keyboard is a standard four-bank keyboard which can generate 63 Fieldata codes. A row of eight interrupt keys is located immediately above it. The CRT can display 16 lines of 64 characters each.

- UNIVAC Pagewriter

  The UNIVAC Pagewriter provides a hard copy of all messages for a permanent record of all completed transactions between the operator and the Executive system. The Pagewriter prints lines of up to 80 characters each at a rate of 25 characters per second.

- Day Clock

  The day clock on the console displays the time of day in hours, minutes, and hundredths of minutes. It furnishes the time of day to the CPU every 600 milliseconds and sends a day clock interrupt signal to the CPU every 6 seconds. The day clock may be manually disabled from the operator's console.

  On a multi-processor system, any one day clock may be selected to be active either externally or by program.

- Operator's Control and Display Panel

  The Operator's Control and Display panel includes fault, disable, and mode indicators for its CPU and associated main storage modules; displays and controls associated with selecting and releasing jumps and stops, with the Program Address Counter and Memory Select Register, and with the time display of the day clock. It also includes system controls associated with the CPU and subsystems which are logically connected to the CPU.

- Additional Features

  An auxiliary right- or left-wing console to accommodate control/display panels for Communications Terminal Module Controller subsystems can also be included.

## 2.3. CONFIGURATIONS

The introduction of the UNIVAC 1108 System with its many components provides a most flexible system. The many configurations of individual components are almost limitless. Figures 2–5, 2–6, and 2–7 illustrate typical possibilities.

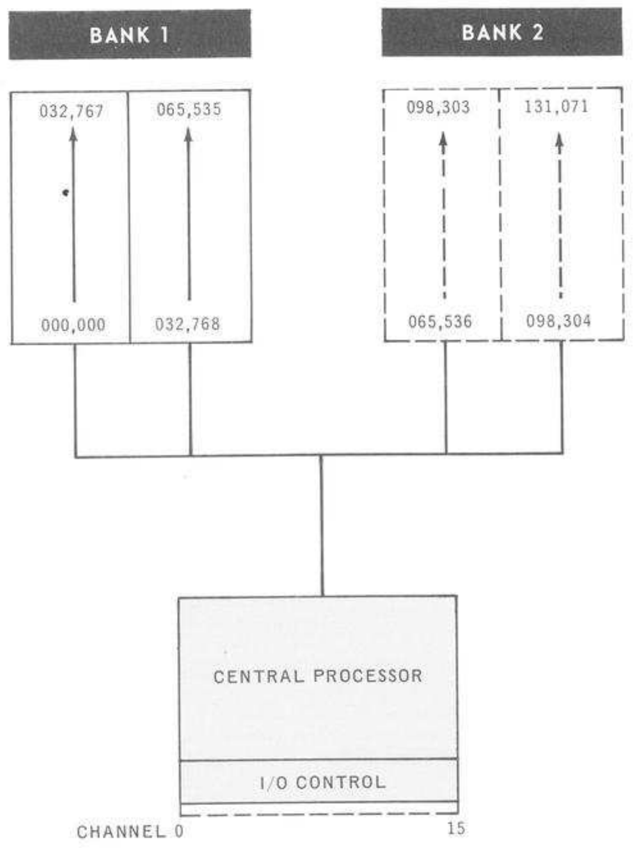


*Figure 2–5. Unit Processor System with Noninterleaved Storage*

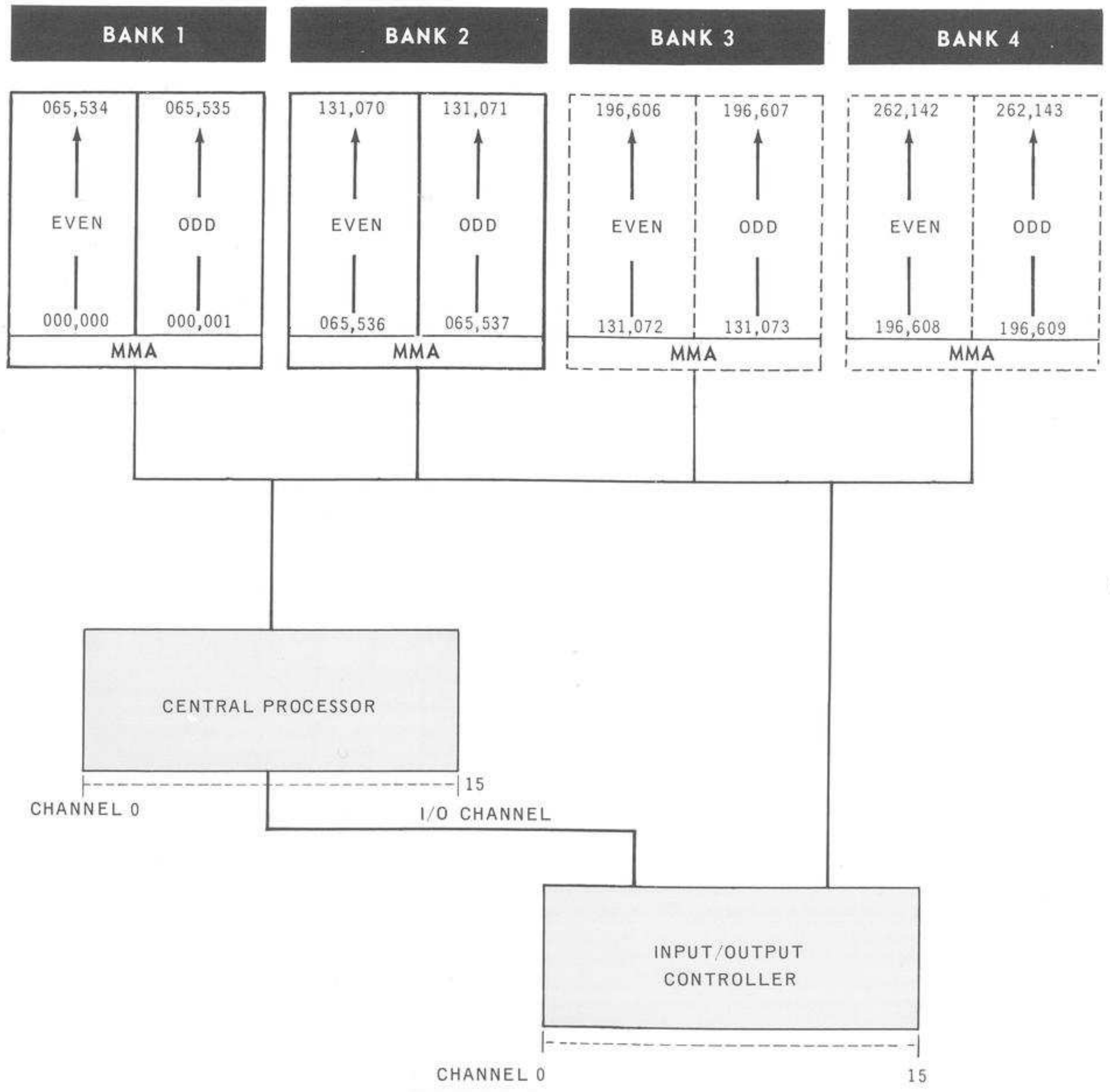


*Figure 2–6. Single Processor With I/O Controller and Interleaved Storage*

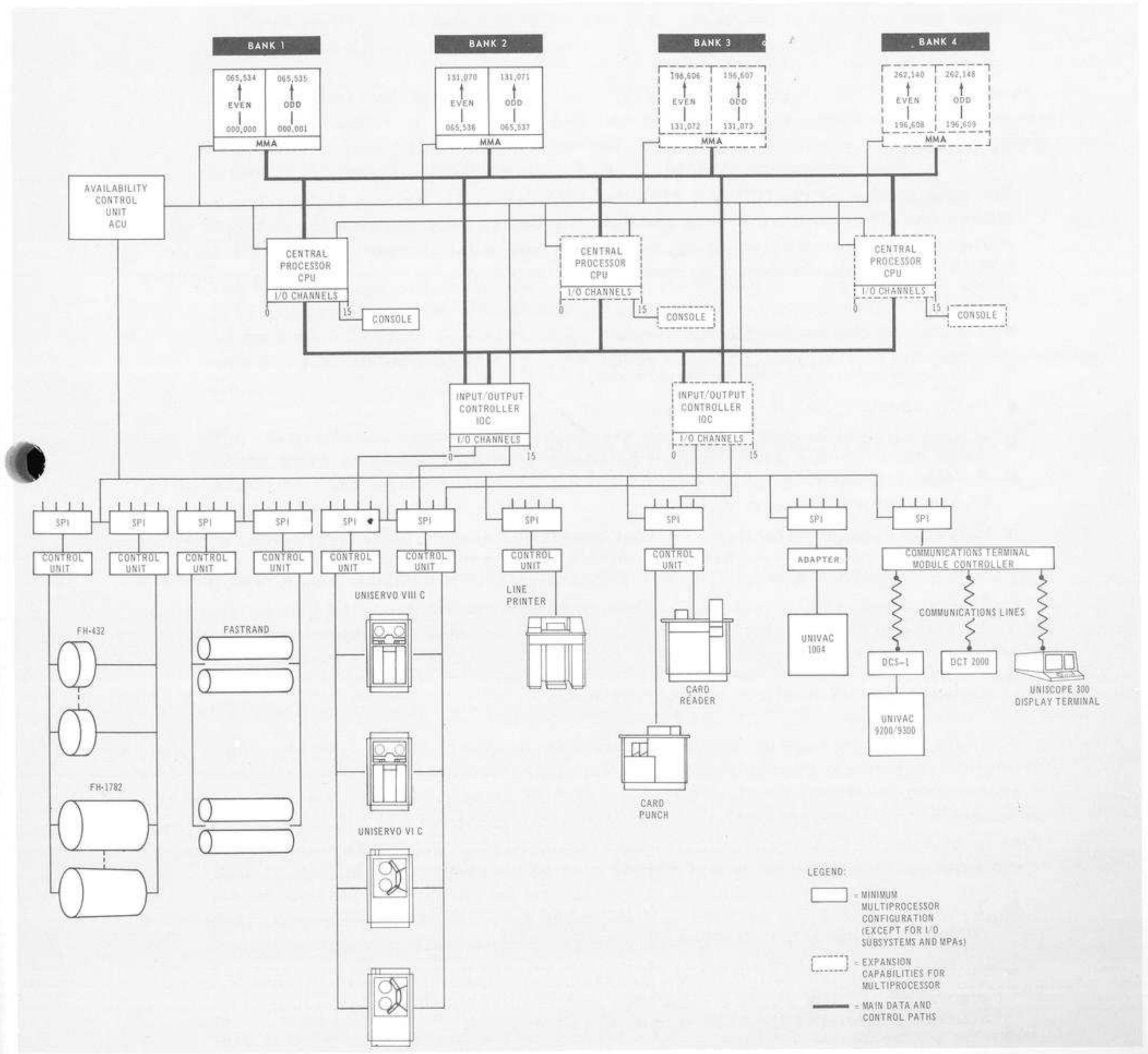


*Figure 2–7. Multi-Processor System*

# 3. MAIN STORAGE

## 3.1. GENERAL

The main storage of the UNIVAC 1108 Multi-Processor System is a high performance, fast access repository for instructions and data. Its design fully supports the concepts of multiprogramming, multiprocessing, modularity, and reliability around which the entire UNIVAC 1108 Multi-Processor System is constructed. Among its featured characteristics are:

- 750 nanosecond read/restore cycle time
- 65,536, 131,072, 196,608 or 262,144 thirty-six-bit words
- Parity checking on all storage references
- Access by up to three processors and two IOC's
- Modular expansion two, four, six or eight 32,768-word modules (one, two, three, or four 65,536-word module pairs or banks)
- Hardware storage protection – lockout boundaries establishable in 512-word increments
- Relative addressing and dynamic program relocatability through program base registers
- On-line serviceability – module pairs may be removed for servicing without stopping the entire system
- Overlapping/interleaved main storage access to boost processor performance and to minimize access conflicts among processors.

While these features are all discussed generally as storage features, many of them such as relative addressing, storage protection, overlapping/interleaving are actually functions of each processor. With proper multiprocessor system organization, the main storage then becomes a set of components of the system which are allocatable in the same manner as peripheral devices. In realizing this objective, the design departs from the traditional close integration of the processor and the storage elements in the following ways.

- The main storage is composed of independently accessible modules, yet it presents a continuous addressing structure to the processors.
- In order to service more than one processor, a method of establishing priority among processors (CPU's and IOC's) at each module is provided in case two or more processors attempt to reference the same module simultaneously.
- To ensure that a processor will wait for access to storage, the processor and the module communicate on a request/acknowledge basis.

With these considerations in mind, the storage modules (via the MMA) become passive components which perform the following functions:

- Grant storage access to a number of processors on a priority basis.
- Accept an address from any processor.
- Store or retrieve a word at that address.
- Issue an acknowledgement signifying that a storage reference has been completed
- Check parity on all data and deliver an interrupt signal to the processor requesting access should a parity error occur.

This processor module relationship has significant advantages for the immediate as well as the future needs of the system. Addition of processors or banks of storage is simplified. It becomes a simple matter to add processors or storage elements, or to replace them with improved equipment, module by module, as technology advances.

## 3.2. STORAGE MODULE

The basic storage module includes 32,768 words of ferrite core array. Each word is 36 bits long, and carries two additional parity bits in nonaddressable levels, one bit for each half word. The main components of the module are a 15-bit address register, a 36-bit read/restore register, parity checking circuits, and request/acknowledge circuits.

The 15-bit address register of each storage module provides addressing for 32,768 words. Since an 18-bit address is generated within the processor at each storage reference, three bits are available for selection of one of the eight possible storage modules.

Parity is checked on reading or calculated on writing for each storage access. If a parity error is detected, the storage bank sends a parity error interrupt signal to the processor which it is currently serving, and rewrites the word in its incorrect form to ensure subsequent data errors when the word is again referenced. Preservation of the error in this way facilitates fault location, since the Executive can determine whether the failure is transient or is associated with a marginal or complete failure of the module.

## 3.3. MULTIPLE MODULE ACCESS UNIT

In a multiprocessor system, an MMA unit is connected between each pair of main storage modules and the processors which may reference it. The MMA unit furnishes five priority-ordered processor connection paths (one for each CPU and one for each IOC) to each of the modules of the pair. Should an access conflict occur among processors, the MMA grants storage access to the processor have the highest priority, then the next, and so on. Communications between a processor and a single storage module can, therefore, be asynchronous; if the storage module is busy servicing one processor, a passive wait cycle is induced in others of lower priority that may be referencing it. Because a delay in honoring an input/output transfer can result in an undesirable "go-around" on drum, reread or rewrite on tape, or actual loss of data in the case of real-time input, I/O Controllers are ordinarily attached to the higher priority inputs of the MMA, followed by CPU's, which have built-in precedence of I/O over computational activities.

## 3.4. PACKAGING

Two 32,768-word storage modules (module pair) within a single cabinet constitute a bank. An adjacent cabinet contains DC power supplies for operation of the bank and the associated MMA.

## 3.5. STORAGE CAPACITY

Available storage capacity ranges from 65,536 words to the system maximum of 262,144 words, in steps of 65,536 words, according to the following:

65,536 words (two modules) – Minimum for unit processor system

131,072 words (four modules) – Minimum for multiprocessor system

196,608 words (six modules)

262,144 words (eight modules)

## 3.6. ADDRESSING

Two special techniques for referencing the main storage modules are used to increase processor performance and to reduce the occurrence of multiprocessor access conflicts. The first, called overlapping, enables the CPU to retrieve the current operand and the next instruction simultaneously; the second, called interleaving, enables two processors (CPU's, IOC's, or CPU and IOC) to access a pair of modules with minimum access conflicts.

### 3.6.1. Overlapping

The CPU can determine whether its current operand and next instruction lie in different storage modules, and if they do it retrieves the two words in parallel, at an effective 100% performance increase.

The overlapping feature permits the separation of the instruction and data of a program into separate physical banks. Furthermore, the base register of the CPU allows either the instruction or data area of a program to be relocated independently – a significant advantage in storage compacting to overcome program fragmentation.

### 3.6.2. Interleaving

Interleaving is a method of addressing a main storage module pair (bank) in an even/odd fashion. It significantly reduces storage access conflicts in a multiprocessor system, and thereby increases overall system performance. With interleaving, one module of a pair contains all even numbered locations and the other contains all odd numbered locations. Thus, in a fully-expanded eight module system, modules 0, 2, 4, 6 are referenced for even addresses while modules 1, 3, 5, 7 are referenced for odd. The even/odd module pairs consist of modules 0 and 1, 2 and 3, 4 and 5, and 6 and 7.

For a practical example, substitute the letters A, B, C, D for the modules contained in two banks, and assume data are being stored sequentially by a program. (The same assumption may be made for instructions being executed sequentially by the same program.) With the overlapping feature, assume processor number 1 starts executing instructions and retrieving data, with the instruction area in bank 1 and the data area in bank 2. For simplicity, assume the starting instruction and data addresses are at even numbered locations. The processor will then reference module A-B-A-B. . . for sequential instructions, and C-D-C-D. . . for sequential data locations. In any single storage interval, either modules A-C or B-D will be busy while their alternates will be idle. If another processor starts an identical process, but referencing an odd address to begin with, both processors may run concurrently without one impeding the operation of the other.

Assuming that both processors in the above example started at even addresses, the processor with lower priority passively waits one storage cycle after which the two are again in synchronization and may operate simultaneously.

## 3.7. STORAGE PROTECTION

To prevent inadvertent program reference to out-of-range storage addresses, the 1108 processor includes a hardware storage protection feature. The controlling element in this feature is the Storage Limits Register, the contents of which are as follows:

| INSTRUCTION AREA | | DATA AREA | |
|---|---|---|---|
| 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 I and D boundaries.

Before each main storage reference, the processor performs a limits check on the address, comparing against the limits of either the I or D field of the SLR. An out-of-limits address generates a guard mode interrupt, thereby allowing the Executive to regain control and take appropriate action.

### 3.7.1. Storage Protection Modes

The Executive system can establish two different modes of storage protection by means of control fields in the Processor State Register (PSR) described in Section 4. 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.

#### 3.7.1.1. 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 non-alterable (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.

#### 3.7.1.2. User Program 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.

A particular advantage of read/jump protection is that classified (confidential) programs can be confidently run together; they are fully protected from audit (inadvertent or otherwise) by other programs.

## 3.8. RELATIVE ADDRESSING

Relative addressing is a feature of great significance in multiprogramming, time-sharing, and real-time operations, for 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 storage may be brought in for operation in any available area without complicated relocation algorithms.

Relative addressing is provided for through base registers contained within the CPU. Two separate registers control the basing of the program instruction and data bank, and a third register controls the selection of the appropriate base register.

# 4. 1108 PROCESSOR

## 4.1. GENERAL

The UNIVAC 1108 Central Processor Unit (CPU) is the principal component of the UNIVAC 1108 Multi-Processor system and, generally, the one by which the entire system is identified. It can operate under Executive or user program modes of control; it performs both arithmetic and logical operations; and it accommodates and supervises up to 16 input/output channels.

## 4.2. PRINCIPAL SECTIONS

The processor is logically divided into six interacting sections each of which is identified and briefly described below.

- Control Registers – The CPU has 128 program-addressable control registers used for arithmetic operations, indexing, and input/output buffer control.
- Arithmetic Section – This section contains the adder registers, and control circuits necessary for performing fixed and floating point arithmetic, partial-word selection, shifting, logical operations, and tests.
- Control Section – This section provides the basic control and logic for instruction decoding and execution. It includes the Program Address Counter used for the sequential accession of instructions; the Program Control Register in which instructions are staticized for execution; and the Processor State Register (PSR), which determines various processor operating modes. The Control Section also services interrupts.
- Input/Output Section – This section controls and multiplexes data flow between main storage and 16 input/output channels. It includes an interrupt priority network and paths to peripheral subsystems for both function signals and data.
- Indexing Section – This section contains parallel index adders and threshold test circuitry. It is used generally for processor control functions, operand address development, program relocation, and input/output transfer control.
- Storage Class Control Section – The Storage Class Control section receives the final operand address from the index adder and establishes address and data paths to one of eight possible storage modules. Storage Class Control also determines whether a final address refers to a control register.

## 4.3. INSTRUCTION WORD FORMAT

The format of the 1108 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.

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

### 4.3.1. Function Code

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 illegal function code generates an interrupt.

### 4.3.2. Partial-Word or Immediate-Operand Designator

For function codes less than $70_8$, the j designator specifies partial-word or immediate-operand selection. (See Figure 4–2 for specific partial-word selections.)

### 4.3.3. Control Register Designator

The four-bit a field designates which control register, within a group selected by the function code, is involved in the operation. For some operations, the a field refers to an arithmetic register; for others, it refers to either an index register or some other control register. In input/output instructions, it specifies 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 128 control registers.

### 4.3.4. Index Register Designator

The x field specifies one of the 15 index registers to be used in address modification. When register 00 is designated, indexing is suppressed.

### 4.3.5. Index Modification Designator

The h field controls modification of the index value (Xm) by the increment field (Xi) after indexing (see 4.4.1). If h = 1, the right half of the index register is modified by the contents of its left half; if h = 0, modification is suppressed.

### 4.3.6. Indirect Address Designator

The i designator controls the use of indirect addressing during instruction execution. If i = 0, the instruction functions normally. If i = 1, the 22 least significant bit positions of the instruction (x, h, i and u fields) are replaced in the instruction register with the contents of the 22 least significant bit positions of (U). Indirect addressing continues as long as i = 1 with full indexing capability at each level.

### 4.3.7. Address Field

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

## 4.4. CONTROL REGISTERS

The 128 program-addressable control registers are grouped to provide multiple index registers, accumulators, input/output access control registers, and special registers (see Figure 4–1).

The control registers consist of 36-bit integrated-circuit registers, with a basic cycle time of 125 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. UNIVAC 1108 compilers, for example, perform significantly better through multiple register usage, and can produce highly efficient code.

In the following descriptions only programmable registers are discussed. The Executive, through modes established by the Processor State Register, has exclusive use of the duplicate set of control registers as well as the Access Control Registers indicated by the shaded areas in Figure 4–1.

### 4.4.1. Index Registers

Control register locations 00–15 are Index Registers and have the following format:

| $X_i$ | $X_m$ |
|---|---|
| 35 MODIFIER INCREMENT OR DECREMENT 18 | 17 INDEX MODIFIER 0 |

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.

Index Register 00, while program addressable, stores the contents of the Processor State Register (PSR) upon occurrence of an interrupt. Since its contents are overwritten at each interrupt, it is not generally useful for programming purposes.

4.4.2. Arithmetic Accumulators

Control register locations 12–27 are arithmetic accumulators, for programmed storage of arithmetic operands and results. The computation is performed in other 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, i.e., A and A + 1. In arithmetic operations, 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 memory but produce a two-word result in the specified A and A + 1.

4.4.3. Access Control Registers

Control register locations 32–63 are Input and Output Access Control Registers (ACR's). They are guard mode protected and may be referenced only by the Executive. Formats of the Access Control Words are detailed in Section 5.

The word-by-word transmission of data over an I/O channel is governed by the contents of the ACR's. Two ACR's, one for input and one for output, are assigned to each of the sixteen channels. Input ACR's (locations 32–47) control input data transfers while output ACR's (locations 48–63) govern the transmission of output data and Function Words.

When an input/output operation is initiated, the programmed ACR word is loaded into the ACR corresponding to the channel associated with the specified peripheral unit.

4.4.4. Registers

The sixteen control register locations 64–79 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.

4.4.4.1. R0 – Real Time Clock

| UNASSIGNED | CLOCK COUNT |
|---|---|
| 35 … 18 | 17 … 0 |

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 goes through zero. Thus, if the clock is initially loaded with the value 5000, an interrupt occurs in exactly one second.

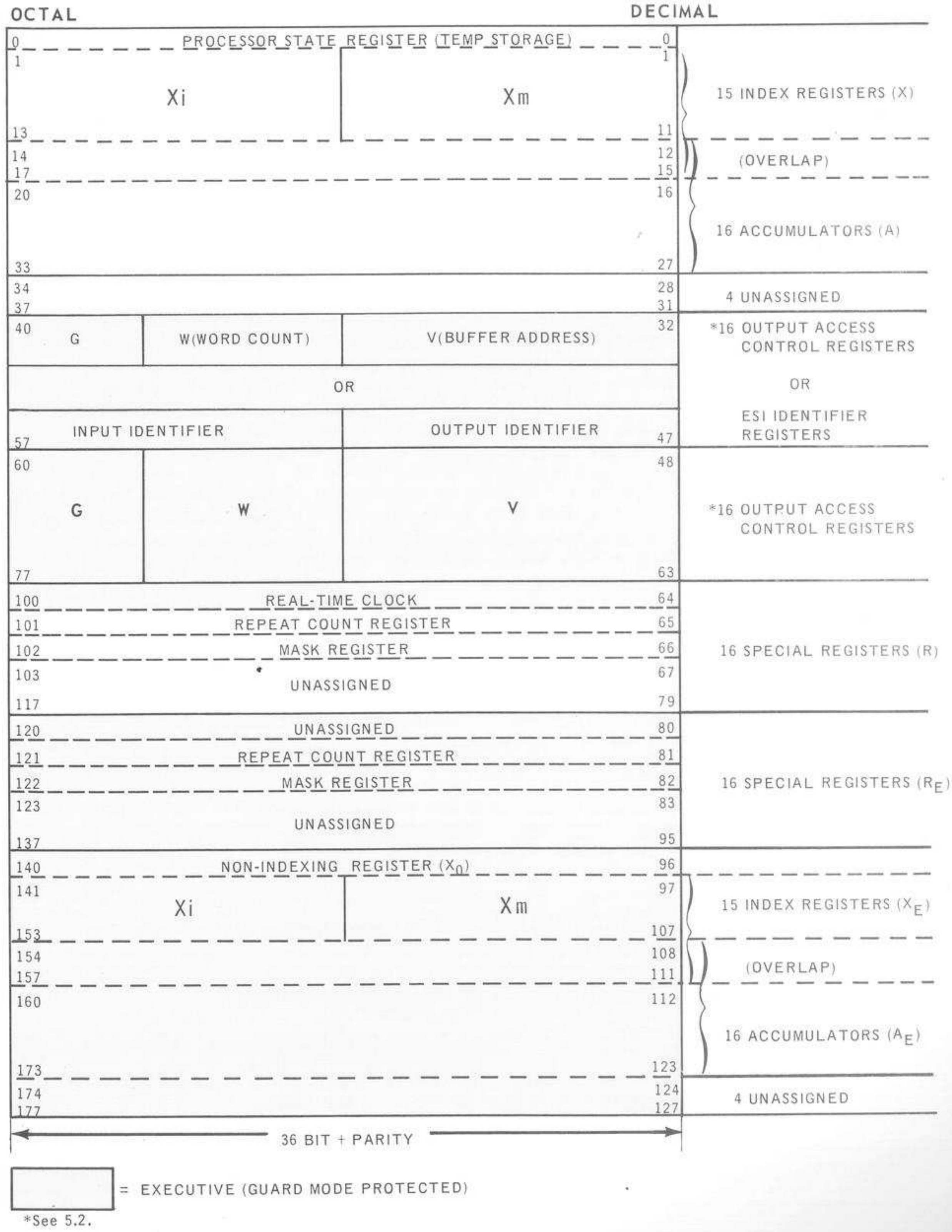


*Figure 4–1. Control Register Address Assignments*

4.4.4.2. R1 – Repeat Counter

| UNASSIGNED | REPEAT COUNT (k) |
|---|---|
| 35 ... 18 | 17 ... 0 |

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.

4.4.4.3. R2 – Mask Register

The Mask Register functions as a filter in determining which portions of words are to be tested in repeated masked search operations or in logical comparisons. (U) is compared to (A) only with respect to those positions which contain one's in the Mask Register. In repeated masked search operations, both the Mask Register and the Repeat Counter are loaded prior to executing the search command.

## 4.5. ARITHMETIC SECTION

In the UNIVAC 1108 System the manipulation of data (addition, subtraction, multiplication, division, shifting) takes place in the arithmetic section of the central processor. During the execution of an arithmetic instruction, storage registers within the arithmetic section itself are used for actual computation. The arithmetic section has the following characteristics:

- On fixed point, single precision instructions the j designator selects all or a portion of one of the operands (half, third, quarter or sixth word) for use in the arithmetic operation.
- Special split-word arithmetic instructions provide for simultaneous addition or subtraction of corresponding half or third words of the two operands.
- When a shift matrix is used, a multiposition shift requires the same time as a one place shift. Right and left shifts of single or double length operands can be specified. Left shifting is logical (zeros are filled to the right). Right shifts may be either logical or algebraic (sign bits are filled to the left).
- Sixteen arithmetic registers in the control register section, acting as sixteen accumulators, allow parallel and cumulative computation. Full double precision floating point arithmetic is provided.
- When the results of arithmetic operations are in double-length form, they are automatically stored in consecutive control registers and are available for retrieval as double-length results.
- Alphanumeric comparisons utilizing the Mask Register allow any selection of bits in one 36-bit word, to be directly compared with corresponding bits of another word.

4.5.1. The Adder

The adder in the 1108 Processor is a one's complement subtractive adder for 36-bit or 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, are the only instructions which affect these two designators.

Before the execution of one of these instructions both designators are cleared. The overflow designator is set upon generation of a significant bit in the sign position. Thus a positive result from two negative quantities or a negative result from two positive quantities sets the overflow designator. The carry designator is set whenever an end-around carry is generated. This indicates the involvement of a negative quantity as one of the operands.

After the instruction has been performed, the designators remain either set or clear until another of the designated arithmetic instructions is initiated. Both designators are set in time to be tested immediately after the specified instruction has been executed.

When an interrupt occurs, the hardware stores the settings of the carry and overflow designators in the Processor State Register (see 4.6.1) and control passes to the Executive system. This information is automatically returned to the designators when the Executive returns control to the interrupted program.

4.5.2. Arithmetic Accumulators

The sixteen arithmetic accumulators can be addressed directly by the programmer and are available for storing operands and results of arithmetic computations. These arithmetic accumulators should not be confused with the non-addressable transient registers contained within the arithmetic section itself used in actual computation.

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.

4.5.3. Partial-Word Transfers

To minimize shifting and masking and to allow computation based on selected portions of words, the 1108 System permits the transfer of partial words into and out of the arithmetic section in a varying pattern (see Figure 4–2).

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

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

S = Sign Extension, where the sign is that of the j-determined final contents of A.

*Figure 4–2. J-Determined Partial Word Operation.*

## 4.5.4. 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 same two operands are added and the result is stored in the left half of the same accumulator. There is no carry interaction between the halves. The same holds true for thirds of words. Each partial word operates as an independent arithmetic register, with its own end-around carry.

## 4.5.5. Shifting

The 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.
- Left Logical – zeros replace bits shifted out of the least significant positions.
- Right Algebraic – sign bits replace bits shifted out of the most significant positions.
- Scale-Factor Shift – a single or double accumulator left shift which positions the word and simultaneously counts the number of shifts required until $(A_{35}) \neq (A_{34})$.

## 4.5.6. Double Precision Fixed Point Arithmetic

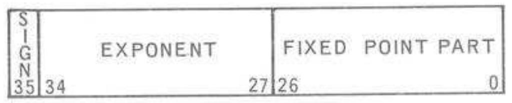
The System provides 72-bit, double precision fixed point addition and subtraction. Operands are processed as if they occupied a single 72-bit register. Bit 71, the high order bit, is the sign bit.

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. Integer and fractional division is performed upon a double-length dividend with the quotient retained in A and the remainder retained in A + 1.

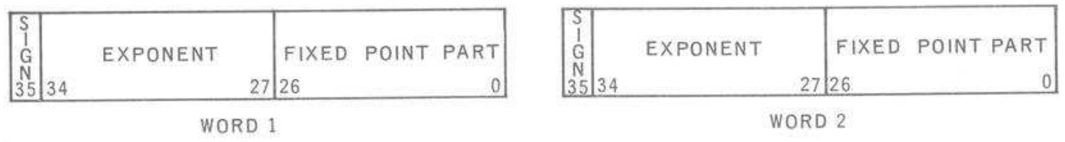
## 4.5.7. Floating Point Arithmetic

The System is equipped with an extensive hardware repertoire of floating point instructions. If the arithmetic is single precision, the range is from $10^{38}$ to $10^{-37}$ with eight-digit precision. The word formats are given below.

Source Operand Format



Result Format



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 least significant word is displaced 27 bits to the right of the binary point in the significant word. Hence, its exponent is always adjusted by −27. The two-word result of this single precision operation is developed in two contiguous Arithmetic Registers.

If the arithmetic is double precision, the range is from $10^{307}$ 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

| | | | |
|---|---|---|---|
| SIGN | EXPONENT | FIXED | POINT PART |
| 35 | 34 — 25 | 24 — 0 | 35 — 0 |

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

In any of the floating point formats the exponent can assume a range of values as follows:

| | | |
|---|---|---|
| Single precision | (8 bits): | 000–255 |
| Double precision | (11 bits): | 0000–2047 |

To express negative exponents, the hardware biases or 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 | $-128_{10}$ to $+127_{10}$ | $0 - 255_{10}$ |
| Double precision | $-1024_{10}$ to $+1023_{10}$ | $0 - 2047_{10}$ |

A positive fixed point part is normally assumed to be in range ½ to 1. Such a value places a 1 bit in the most significant bit position. When this condition exists, the floating point number is said to be normalized. A negative fixed point part causes the entire floating point word to be complemented, and a 0 appears in this position.

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

## 4.6. EXECUTIVE SYSTEM CONTROL FEATURES

To maintain the multiprogramming and multiprocessing environment the Executive must have complete control of the entire 1108 system. Special hardware features are provided to permit this control.

The multiprogramming and multiprocessing capabilities of the system are based upon Guard Mode operation. In this mode certain instructions, registers, and storage locations are available for the exclusive use of the Executive System. Under the guard mode, unrelated programs cannot interact.

### 4.6.1. Processor State Register

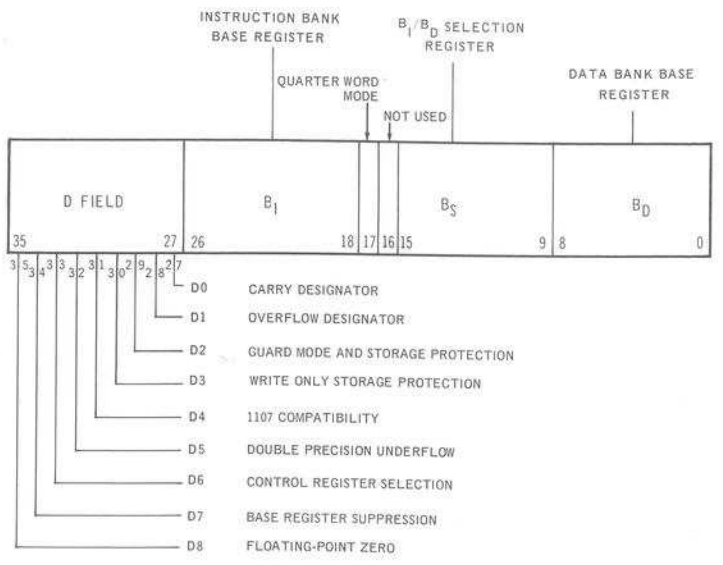
The Processor State Register (PSR) stores a 36-bit representation of various states and conditions affecting the current operations of the Processor. By means of this register the Executive sets up control modes for itself, governs the operation of worker programs, and registers status information concerning worker programs when it regains control in consequence of interrupts. Figure 4–3 explains in detail the significance of each bit of the PSR.

The Executive uses a special instruction, Load Processor State Register, for load PSR, and governing the following functions and conditions:

- Program base addresses
- Quarter word operations in the processor
- Carry and overflow status
- Guard Mode
- Storage protection mode
- 1107 compatibility mode
- Floating point underflow mode (double precision operations)
- Base register suppression
- Control register process selection
- IBM* 7090 floating point compatibility mode

The contents of the PSR are stored automatically as soon as an interrupt occurs. Program carry and overflow status are first registered in PSR and then its contents are transferred to Control Register location 00 (Index Register zero). The PSR is then force-cleared in preparation for Executive operations. The Executive saves the contents of Control Register 00 so that it can reinstate conditions as control is returned to the program that was interrupted.

*Registered trademark of International Business Machines Corporation.


INSTRUCTION BANK BASE REGISTER
$B_I/B_D$ SELECTION REGISTER
QUARTER WORD MODE
NOT USED
DATA BANK BASE REGISTER
D FIELD
$B_I$
$B_S$
$B_D$
35
27
26
18
17
16
15
9
8
0
D0 CARRY DESIGNATOR
D1 OVERFLOW DESIGNATOR
D2 GUARD MODE AND STORAGE PROTECTION
D3 WRITE ONLY STORAGE PROTECTION
D4 1107 COMPATIBILITY
D5 DOUBLE PRECISION UNDERFLOW
D6 CONTROL REGISTER SELECTION
D7 BASE REGISTER SUPPRESSION
D8 FLOATING-POINT ZERO

D FIELD

D8 7090 FLOATING-POINT COMPATIBILITY MODE

=0 Clears exponent to zero when a fixed point part equal to zero is generated
=1 Produces relative floating point 0

D7 BASE REGISTER SUPPRESSION

=0 Allows contents of Base Registers to be added to every U address.
=1 Base Register addition on storage reference is suppressed when instruction i-designator = 1.

D6 CONTROL REGISTER SECTION

=0 Selects User Program control Register Set (Locations $00–37_8$, $100–117_8$)
=1 Selects Executive Control Register Set (Locations $120–177_8$)

The Executive passes control to User Programs with bit 33 = 0, which selects the worker set. An interrupt forces this bit to 1 after (PSR) has been transferred to X0, making the upper Control Registers available to the Executive.

D5 DOUBLE PRECISION UNDERFLOW – Double Precision Floating Point Operations

=0 Interrupts on D.P. Floating Underflow
=1 Clears results to zero and continues

This is a program-requestable option which is set up for the program by the Executive.

D4 1107 COMPATIBILITY MODE

=0 1108 Mode – Full Range Addressing
=1 The upper two bits of the effective address (U) are stripped, allowing 1107 program compatibility with the 1108. In this mode, only 65,536 words of storage are available.

D3 WRITE-ONLY STORAGE PROTECTION

=0 Read, Write and Jump storage protection under Guard Mode
=1 Write Protection Only under Guard Mode

D2 GUARD MODE

=0 Guard Mode off. All instructions and storage references to Access Control Registers ($40–77_8$), the Real Time Clock ($100_8$) and Executive Control Registers ($120–177_8$) are permitted.
=1 Guard Mode on. Invalidates all instructions and control register references described above to enforce the integrity of the system. Only when Guard Mode is on are the contents of the storage limits register effective in storage protection. (If D3 = 1, there is no read or jump protection.)

D1 OVERFLOW DESIGNATOR

=1 Arithmetic Overflow at time of interrupt.

D0 CARRY DESIGNATOR

=1 Arithmetic Carry at time of interrupt

OTHER FIELDS

0–8, 9–15, 18–26 } BASE REGISTERS. These registers provide the absolute base address values on which programs "float" in storage during execution.

17 QUARTER-WORD MODE BIT

=1 Quarter-word mode effective
=0 Quarter-word mode not effective.

*Figure 4–3. Processor State Register Format*

## 4.6.2. Interrupts

The interrupt network of the UNIVAC 1108 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) or the control section of the central processor. Specific interrupt locations are assigned within the lower regions of main storage for each condition. 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 is accomplished through some of these interrupts.

Other interrupts are provided for certain error conditions within the central processor. These may result from a programming fault such as an illegal instruction, a main 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. Table 4–1 lists the fixed-address assignments. Note that they are all interrupt locations except for $200_8$ through $202_8$, which receive status words, and $216_8$, which stores the day clock count.

## 4.6.3. Guard Mode

The Guard Mode prevents user programs from executing any of the instructions listed below. These are reserved for the Executive. It also protects certain locations in main storage 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 processor interrupt.

- Load Processor State Register
- Load Storage Limits Register
- Initiate Interprocessor Interrupt
- Select Interrupt Location
- Load Channel Select Register
- Halt Instructions
- All I/O Instructions
- Disabling of I/O interrupts for more than 100 microseconds
- Attempting to write into any of the Executive control registers (32–64 or 80–127)

Guard Mode is disabled by the occurrence of any interrupt. This stores the contents of PSR in user Index Register 0, clears certain bit positions of the PSR, sets D6 =1, and establish Executive Mode operation.

| DECIMAL ADDRESS | OCTAL ADDRESS | FIXED ASSIGNMENT |
|---|---|---|
| 128 | 200 | Status Code for External Interrupt on CPU #0 |
| 129 | 201 | Status Code for External Interrupt on CPU #1 |
| 130 | 202 | Status Code for External Interrupt on CPU #2 |
| 131–135 | 203–207 | Not Used |
| 136 | 210 | Power Loss Interrupt |
| 137 | 211 | I/O ESI Access Control Register Parity Error Interrupt |
| 138 | 212 | I/O ISI Access Control Register Parity Error Interrupt |
| 139 | 213 | I/O Data Parity Error Interrupt |
| 140–141 | 214–5 | Not Used |
| 142 | 216 | Day Clock Count |
| 143 | 217 | Day Clock Interrupt |
| 144 | 220 | ISI Input Monitor Interrupt |
| 145 | 221 | ISI Output Monitor Interrupt |
| 146 | 222 | ISI Function Monitor Interrupt |
| 147 | 223 | ISI External Interrupt |
| 148 | 224 | ESI Input Monitor Interrupt |
| 149 | 225 | ESI Output Monitor Interrupt |
| 150 | 226 | Not Used |
| 151 | 227 | ESI External Interrupt |
| 152 | 230 | Status Code for External Interrupt on Unit Processor (not used by multiprocessor) |
| 153 | 231 | Real Time Clock Interrupt |
| 154 | 232 | Interprocessor Interrupt #0 |
| 155 | 233 | Interprocessor Interrupt #1 |
| 156 | 234 | Not Used |
| 157 | 235 | Main Storage Parity Error Interrupt (Bank #1) |
| 158 | 236 | Main Storage Parity Error Interrupt (Bank #2) |
| 159 | 237 | Main Storage Parity Error Interrupt (Bank #3) |
| 160 | 240 | Control Register Parity Error Interrupt |
| 161 | 241 | Illegal Instruction Interrupt |
| 162 | 242 | Executive Return Interrupt |
| 163 | 243 | Guard Mode Interrupt |
| 164 | 244 | Test and Set Interrupt |
| 165 | 245 | Floating Point Underflow Interrupt |
| 166 | 246 | Floating Point Overflow Interrupt |
| 167 | 247 | Divide Fault Interrupt |
| Last address | −1 | Main Storage Parity Error Interrupt (Bank #0) |

*Table 4–1. Fixed-Address Assignments*

## 4.7. INSTRUCTION REPERTOIRE

The UNIVAC 1108 Processor is provided with an unusually powerful and flexible instruction repertoire. Many 1108 instructions are effectively accessed and completed in the time of one storage cycle. In addition to a complete set of instructions including an extremely fast set of single and double precision floating point instructions, the repertoire includes a group which permits fast and simplified control by the Executive System operating in a multiprogramming or multiprocessing environment.

In the following discussion, the instructions in the 1108 repertoire are grouped by functional class to illustrate the power of the repertoire. Appendix C lists them numerically by octal code stating exactly what each one does. The octal codes are listed here to facilitate reference to Appendix C.

### 4.7.1. Data Transfer Instructions

To load the Arithmetic registers:

| | |
|---|---|
| Load A | 10 |
| Load Negative A | 11 |
| Load Magnitude A | 12 |
| Load Negative Magnitude A | 13 |

To load the Index and R registers:

| | |
|---|---|
| Load R | 23 |
| Load X | 27 |
| Load X Modifier | 26 |
| Load X Increment | 46 |

To load two Arithmetic registers with one instruction:

| | |
|---|---|
| Double Load A | 71,13 |
| Double Load Negative A | 71,14 |
| Double Load Magnitude A | 71,15 |

To store the Arithmetic registers:

| | |
|---|---|
| Store A | 01 |
| Store Negative A | 02 |
| Store Magnitude A | 03 |

To store other control registers

| | |
|---|---|
| Store X | 06 |
| Store R | 04 |

To store two Arithmetic registers with one instruction:

| | |
|---|---|
| Double Store A | 71, 12 |

Two special purpose transfers:

| | |
|---|---|
| Store Zero | 05 |
| Block transfer, repeat | 22 |

Any transfer instructions, except double-length transfers, move selected parts of words. That is, the partial-word feature allows any sixth, quarter, third, or half word to be loaded into the lower portion of an arithmetic register, when using Load instructions. Similarly, when using a Store instruction any sixth, quarter, third or half word can be transmitted from the lower portion of an arithmetic register, index register, or R Register to main storage.

## 4.7.2. Fixed Point Arithmetic

Single-word operations on arithmetic registers:

| | |
|---|---|
| Add to A | 14 |
| Add Negative A | 15 |
| Add Magnitude to A | 16 |
| Add Negative Magnitude to A | 17 |
| Add Upper | 20 |
| Add Negative Upper | 21 |
| Multiply Integer | 30 |
| Multiply Single Integer | 31 |
| Multiply Fractional | 32 |
| Divide Integer | 34 |
| Divide Single Fractional | 35 |
| Divide Fractional | 36 |

Double length operations on two arithmetic registers:

| | |
|---|---|
| Double Precision Fixed Point Add | 71,10 |
| Double Precision Fixed Point Add Negative | 71,11 |

Special format operations:

| | |
|---|---|
| Add Halves | 72,04 |
| Add Negative Halves | 72,05 |
| Add Thirds | 72,06 |
| Add Negative Thirds | 72,07 |
| Add to X | 24 |
| Add Negative to X | 25 |

## 4.7.3. Floating Point Arithmetic

The repertoire includes both single and double precision floating point operations, using one-word and two-word operands, respectively. One's complement arithmetic is used.

Single Precision

| | |
|---|---|
| Floating Add | 76,00 |
| Floating Add Negative | 76,01 |
| Floating Multiply | 76,02 |
| Floating Divide | 76,03 |
| Load and Unpack Floating | 76,04 |
| Load and Convert to Floating | 76,05 |

Double Precision

| | |
|---|---|
| Double Precision Floating Add | 76,10 |
| Double Precision Floating Add Negative | 76,11 |
| Double Precision Floating Multiply | 76,12 |
| Double Precision Floating Divide | 76,14 |
| Double Load and Convert to Floating | 76,15 |

Miscellaneous

| | |
|---|---|
| Magnitude of Characteristic Difference to Upper | 76,06 |
| Characteristic Difference to Upper | 76,07 |
| Floating Expand and Load | 76,16 |
| Floating Compress and Load | 76,17 |

## 4.7.4. Index Register Instructions

These instructions can be used when modifying, loading, or storing the contents of Index registers:

| | |
|---|---|
| Add to X | 24 |
| Add Negative to X | 25 |
| Load X Modifier | 26 |
| Load X | 27 |
| Store X | 06 |
| Load X Increment | 46 |
| Load Modifier and Jump | 74,13 |
| Test Less or Equal to Modifier | 47 |
| Jump Modifier Greater and Increment | 74,12 |

These instructions address the appropriate index register. Four of the index registers are overlapped with the arithmetic registers; thus all arithmetic instructions, such as multiply or shift, can operate directly on these four index registers.

## 4.7.5. Logical Instructions

The logical or Boolean operations are defined by the following truth tables.

Logical AND

| | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

Inclusive OR

| | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 1 |

Exclusive OR

| | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

The three simple logical instructions are:

| | |
|---|---|
| Logical OR | 40 |
| Logical Exclusive OR | 41 |
| Logical AND | 42 |

One special replace instruction is also logical:

| | |
|---|---|
| Masked Load Upper | 43 |

Several other instructions such as the Repeated Masked Searches employ logical operations in combination with other functions.

## 4.7.6. Shift Instructions

The twelve shift functions include circular, logical, and algebraic shifts. Circular shifts are end-around. Logical shifts fill in zeros on the end opposite the shift direction, whereas algebraic shifts fill in sign bits. The shift count (from 0 through 72 places) is taken from the u field (indexed when specified) of the shift instruction.

Right shift instructions

| | |
|---|---|
| Single Shift Circular | 73,00 |
| Double Shift Circular | 73,01 |
| Single Shift Logical | 73,02 |
| Double Shift Logical | 73,03 |
| Single Shift Algebraic | 73,04 |
| Double Shift Algebraic | 73,05 |

Left shift instructions

| | |
|---|---|
| Load Shift and Count | 73,06 |
| Double Load Shift and Count | 73,07 |
| Left Single Shift Circular | 73,10 |
| Left Double Shift Circular | 73,11 |
| Left Single Shift Logical | 73,12 |
| Left Double Shift Logical | 73,13 |

## 4.7.7. Repeated-Search Instructions

Search instructions operate as repeated comparison operations, comparing the value at u with that in A. They skip the next instruction when a specified condition is met or take the next instruction in sequence when the repeat count in R1 reaches zero.

Algebraic (Sign considered)

| | |
|---|---|
| Search for Equal | 62 |
| Search for Not Equal | 63 |
| Search for Less or Equal | 64 |
| Search for Greater | 65 |
| Search for Within Range | 66 |
| Search for Not Within Range | 67 |

Masked Algebraic (Sign Considered)

| | |
|---|---|
| Masked Search for Equal | 71,00 |
| Masked Search for Not Equal | 71,01 |
| Masked Search for Less or Equal | 71,02 |
| Masked Search for Greater | 71,03 |
| Masked Search for Within Range | 71,04 |
| Masked Search for Not Within Range | 71,05 |

Masked Alphanumeric (Unsigned)

| | |
|---|---|
| Masked Alphanumeric Search for Less or Equal | 71,06 |
| Masked Alphanumeric Search for Greater | 71,07 |

## 4.7.8. Unconditional Jump Instructions

These instructions transfer control to the location specified by the indexed u address.

| | |
|---|---|
| Store Location and Jump | 72,01 |
| Load Modifier and Jump | 74,13 |

## 4.7.9. Conditional Jump Instructions

These instructions make a comparison and if a specific condition is met, they transfer program control to the instruction location specified by u. If not, the next instruction in sequence is executed.

| | |
|---|---|
| Jump on Greater and Decrement | 70 * |
| Double Precision Zero Jump | 71,16 |
| Jump on Positive and Shift | 72,02 |
| Jump on Negative and Shift | 72,03 |
| Jump on Zero | 74,00 |
| Jump on Non Zero | 74,01 |
| Jump on Positive | 74,02 |
| Jump on Negative | 74,03 |
| Jump on Keys | 74,04 |
| Halt on Keys and Jump | 74,05 |
| Jump on No Low Bit | 74,10 |
| Jump on Low Bit | 74,11 |
| Jump Modifier Greater and Increment | 74,12 |
| Jump on Overflow | 74,14 |
| Jump on No Overflow | 74,15 |
| Jump on Carry | 74,16 |
| Jump on No Carry | 74,17 |
| Jump on Input Channel Busy | 75,02 |
| Jump on Output Channel Busy | 75,06 |
| Jump on Function in Channel | 75,12 |

* The j and a designators specify one of 128 control registers.

## 4.7.10. Test (Or Skip) Instructions

These instructions make a comparison and if the specified condition is met, the next instruction is skipped. If not, the next instruction is executed.

| | |
|---|---|
| Test Even Parity | 44 |
| Test Odd Parity | 45 |
| Test Less or Equal to Modifier | 47 |
| Test for Zero | 50 |
| Test for Non Zero | 51 |
| Test for Equal | 52 |
| Test for Not Equal | 53 |
| Test for Less or Equal | 54 |
| Test for Greater | 55 |
| Test for Within Range | 56 |
| Test for Not Within Range | 57 |
| Test for Positive | 60 |
| Test for Negative | 61 |
| Double Precision Test Equal | 71,17 |

### 4.7.11. Executive System Control Instructions

This group of instructions allows proper Executive System control of programs operating in a multiprogramming and multiprocessing environment.

| | |
|---|---|
| Executive Return | 72,11 |
| Store Channel Number | 72,14 |
| Load Processor State Register | 72,15 |
| Load Storage Limits Register | 72,16 |
| Initiate Interprocessor Interrupt | 73,14 |
| Select Interrupt Locations | 73,15 |
| Load Channel Select Register/Load Last Address Register | 73,16 |
| Prevent All I/O Interrupts and Jump | 72,13 |
| Allow All I/O Interrupts and Jump | 74,07 |

These instructions are used for establishing Processor State, storage limits boundaries, interrupt locations, identification of I/O channels, and interprocessor communication and task assignment.

### 4.7.12. Input/Output Instructions

This group of instructions allows the program (usually the Executive System) to initiate, test and control input/output operations. Monitored instructions interrupt the program when the indicated transfer is completed.

| | |
|---|---|
| Load Input Channel | 75,00 |
| Load Input Channel and Monitor | 75,01 |
| Jump on Input Channel Busy | 75,02 |
| Disconnect Input Channel | 75,03 |
| Load Output Channel | 75,04 |
| Load Output Channel and Monitor | 75,05 |
| Jump on Output Channel Busy | 75,06 |
| Disconnect Output Channel | 75,07 |
| Load Function in Channel | 75,10 |
| Load Function in Channel and Monitor | 75,11 |
| Jump on Functipn in Channel | 75,12 |
| Allow All Channel External Interrupts | 75,14 |
| Prevent All Channel External Interrupts | 75,15 |

### 4.7.13. Other Instructions

| | |
|---|---|
| Execute | 72,10 |
| No Operation | 74,06 |
| Test and Set | 73,17 |

# 5. PROCESSOR INPUT/OUTPUT CONTROL SECTION

## 5.1. GENERAL DESCRIPTION

The input/output control section of the CPU controls transmission of data between main storage and the peripheral subsystems. It communicates with a peripheral subsystem over one of 16 bidirectional input/output channels. Data is transmitted with all 36 bits of a word in parallel; thus each channel has 72 data lines (36 input and 36 output) plus control signal lines. Although most peripheral subsystems use both input and output lines, data flows in only one direction on a channel for a specific I/O instruction.

The I/O control section acts as a small processor to operate many peripheral subsystems concurrently. A programmed I/O instruction selects a specified channel and I/O device on the selected channel, and sets up the required conditions for a given activity. From that point on, the I/O control section automatically controls transmission of data to or from the subsystem at its natural speed. When a subsystem requests a word, the I/O control section refers to an access control word which specifies the location in main storage to or from which data is to be transferred.

## 5.2. PERIPHERAL CONTROL

Input and output can be performed in either of two modes: Externally Specified Index (ESI) for multiplexed data communications devices, and Internally Specified Index (ISI) for other types of peripheral equipment. All I/O channels except channel 15, which is always assigned to the control console, can operate in either mode depending on the setting of a mode switch associated with each channel.

## 5.3. INTERNALLY SPECIFIED INDEX MODE

Each 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 (see Figure 4–1, registers $40_8$ to $77_8$).

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

| G | W<br>WORD COUNT | V<br>STARTING ADDRESS |
|---|---|---|
| 35 34 | 33 18 | 17 0 |

| | |
|---|---|
| V | 18 bits, the starting address in main storage 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.<br>= 00, increment V by 1 after each word is transferred.<br>= 10, decrement V by 1 after each word is transferred.<br>= 01 or 11, do not change V. |

In initiating an input/output operation, the processor stores an access control word in the input or output Access 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 main 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.

## 5.4. EXTERNALLY SPECIFIED INDEX MODE

The Externally Specified Index (ESI), 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 basic 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. Furthermore, by means of a patch card an ESI channel can be set to operate in either halfword (18-bit) or quarterword (9-bit) mode.

Because any 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 main storage at 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 format for the ESI Access Control Word differs somewhat from that for ISI to enable control of half- and quarter-word transfers. The halfword Access Control Word is as follows:

| G | H | W<br>WORD COUNT | V<br>STARTING ADDRESS |
|---|---|---|---|
| 35 34 | 33 | 32 18 | 17 0 |

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 an H field, of one bit; this field is used to indicate which half of location V is to be used, as follows:

| SECOND HALFWORD<br>H=1 | FIRST HALFWORD<br>H=0 |
|---|---|
| 35 18 | 17 0 |

H = 0; use first halfword of location V and switch H to 1.

H = 1; use second halfword of location V, change V address as specified by G, and switch H to 0.

On input the first halfword of an incoming message causes the associated ESI access control word to be transferred from main storage to the subsystem. Since the H bit is zero the data goes to the lower half of location V. V is not altered but H is set to 1 and W is decremented. The access control is then returned to main storage until the next data from the same source arrives. At that time the access control word is again transferred from main storage. Since H now equals 1, the data goes to the upper half of location V, address V is changed as specified by G, W is decremented, H is set to 0, and the control word is stored. A similar sequence applies for output transfers.

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

| G | H | C | W<br>WORD COUNT | V<br>STARTING ADDRESS |
|---|---|---|---|---|
| 35 34 | 33 32 | 31 30 | 29 18 | 17 0 |

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 is the quarter-word designator designating the portion of V that is being addressed as follows:

| FIRST<br>QUARTER WORD | SECOND<br>QUARTER WORD | THIRD<br>QUARTER WORD | FOURTH<br>QUARTER WORD |
|---|---|---|---|
| 35 H=00 27 | 26 H=01 18 | 17 H=10 9 | 8 H=11 0 |

Notice that the data is stored in reverse of the order used in half word operation.

C is a two-bit control field 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 subsystem sends a monitor interrupt to the processor as 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.

## 5.5. BUFFER MODE DATA TRANSFERS

A buffer mode data transfer which occurs independently of main program control is used to transfer data between main storage and the communication subsystem. Before the transfer the program performs the following steps:

(1) Loads the locations specified by the ESI addresses with Access Control Words.

(2) Activates the channel to be used.

(3) Sends a function word to the communication subsystem. This step is not required to effect transfers from low or medium speed Communication Terminal Modules.

Step 2 is accomplished by one of the following four instructions. The access control word should specify a one-word dummy buffer since such a buffer is not normally used in the ESI mode.

| | |
|---|---|
| LIC | Load Input Channel |
| LICM | Load Input Channel and Monitor |
| LOC | Load Output Channel |
| LOCM | Load Output Channel and Monitor |

Step 3 is performed by a Load Function in Channel instruction. In ESI mode this instruction loads the control word for the function into the Output Access Control Register for the channel and forces one external function transfer.

Data is then transferred in quarterwords between main storage and the subsystem without main program intervention. Each time a partial word is transferred to or from storage, 1 is automatically subtracted from the W count of the access control word. When this count becomes zero, the transfer is complete. If monitor is specified, an internal I/O monitor interrupt is set.

## 5.6. INPUT/OUTPUT INFORMATION WORDS

Four types of information words are transmitted between the processor and the peripheral subsystems. Each is accompanied by a control signal which identifies it for the receiving unit.

- Data words which go in either direction.
- Function words which go from processor to subsystem.
- Identifier words which go from processor to subsystem.
- Status words which go from subsystem to processor.

### 5.6.1. Data Words

Data is transmitted all bits in parallel, the number of bits depending on the mode of operation. That is, for ISI, 36-bit parallel; for ESI, 18-bit or 9-bit parallel.

To identify the word as data, the subsystem accompanies the word with an Input Data Request signal. The I/O section acknowledges receipt by returning an input acknowledgement. Similarly, on output the system requests data by means of an output data request. As soon as data is available, the I/O control section sends it and supplies an output acknowledgement to the subsystem.

### 5.6.2. Function Words

The 36-bit function word contains operating instructions for the peripheral subsystem. This includes a function code specifying what is to be done and a unit select code if the subsystem controls more than one peripheral device.

The I/O control section identifies the information as a function word by sending an External function signal after placing the word on the data lines.

### 5.6.3. Identifier Words

The identifier word is used as a search key for any of the search functions. When such a word is sent to a subsystem that is set to perform a search operation, an external function signal accompanying it identifies it as an identifier word. The subsystem stores it in a special register and compares it with each word read until it finds an identical word. It then terminates the search and stores the location of the matching word in the status word for further use by the program. On a search/read function the subsystem starts reading as soon as the matching word is found.

### 5.6.4. Status Words

The 36-bit status word, generated by the subsystem, indicates whether an I/O instruction has been completed normally or not. Indicators within the word indicate any abnormal or error conditions. The CPU stores the word in its external interrupt status location for analysis and further action (see 4.6.2).

## 5.7. PRIORITY CONTROL

Input/output operations are arranged in sequence by a priority control network within the input/output section of the CPU. Although all sixteen I/O channels may be available for data transmissions between the processor and peripheral units at the same time, only one channel can communicate with the central processor at any given instant. Priority control circuits resolve situations in which two or more I/O channels simultaneously seek to communicate with the CPU. The following lists the priorities in descending sequence. If two or more requests have the same priority in this sequence, priority is based on the I/O channel number (lower numbered channel first).

1. Output Data Request (ODR)
2. Input Data Request (IDR)
3. Real Time Clock Decrement
4. Power Loss Interrupt
5. External Interrupt (ESI)
6. Input Monitor Interrupt (ESI)
7. Output Monitor Interrupt (ESI)
8. Real Time Clock Interrupt
9. External Interrupt (ISI)
10. Input Monitor Interrupt (ISI)
11. Output Monitor Interrupt (ISI)
12. Function Monitor Interrupt
13. Interprocessor Interrupt #0
14. Interprocessor Interrupt #1

## 5.8. INPUT/OUTPUT INSTRUCTIONS

The I/O instructions allow the program to activate, test, deactivate, and control I/O operations.

The following six instructions prepare the I/O section of the processor to perform I/O operations on the specified channel. Their use and operation has been explained above, in 5.7.

| | |
|---|---|
| Load Input Channel | 75,00 |
| Load Input Channel and Monitor | 75,01 |
| Load Output Channel | 75,04 |
| Load Output Channel and Monitor | 75,05 |
| Load Function in Channel | 75,10 |
| Load Function in Channel and Monitor | 75,11 |

The following two instructions have no effect upon the operation of the I/O channels but test the specified channel to determine if it is active in the specified mode:

| | |
|---|---|
| Jump on Input Channel Busy | 75,02 |
| Jump on Output Channel Busy | 75,06 |

Similarly, the following instruction does not have any effect upon the operation of the specified channel but tests the specified output channel to determine if the first function word has been sent to the subsystem after the channel has been activated in the function mode:

| | |
|---|---|
| Jump on Function in Channel | 75,12 |

The following two instructions enable and disable the servicing of external interrupts by the I/O Section:

| | |
|---|---|
| Allow All Channel External Interrupts | 75,14 |
| Prevent All Channel External Interrupts | 75,15 |

The following two instructions enable the program to terminate operations of an I/O channel. They are used principally to initiate operations on a channel that has not terminated properly.

| | |
|---|---|
| Disconnect Input Channel | 75,03 |
| Disconnect Output Channel | 75,07 |

### 5.8.1. Monitored Instructions

Input mode, output mode, or function mode can be activated on a channel either with or without monitor.

When ISI is in effect, at the end of a monitored instruction after the data has been transferred, the channel is deactivated and the I/O section sends a monitor interrupt to the CPU; this informs the Executive system that the transfer is complete.

This is not the case in ESI Mode. The Load Function in Channel and Monitor (LFCM) instruction never results in a Monitor Interrupt.

# 6. THE INPUT/OUTPUT CONTROLLER

## 6.1. GENERAL DESCRIPTION

The Input/Output Controller (IOC) is an independent device that controls the operation of up to 16 peripheral subsystems under the direction of as many as three different CPU's. It is functionally similar to the I/O section of the CPU discussed in Section 5. Once an I/O request has been issued to it, the controller is in complete control of the operation, transferring data between main storage and the peripherals without further attention from the CPU that originated the request.

It provides the following enhancements to the multiprocessor system:

- Independent data paths between its peripheral subsystems and main storage.
- Efficient, high speed data communication capabilities.
- Chained-buffer operation (scatter read/gather write).
- Additional I/O channels.

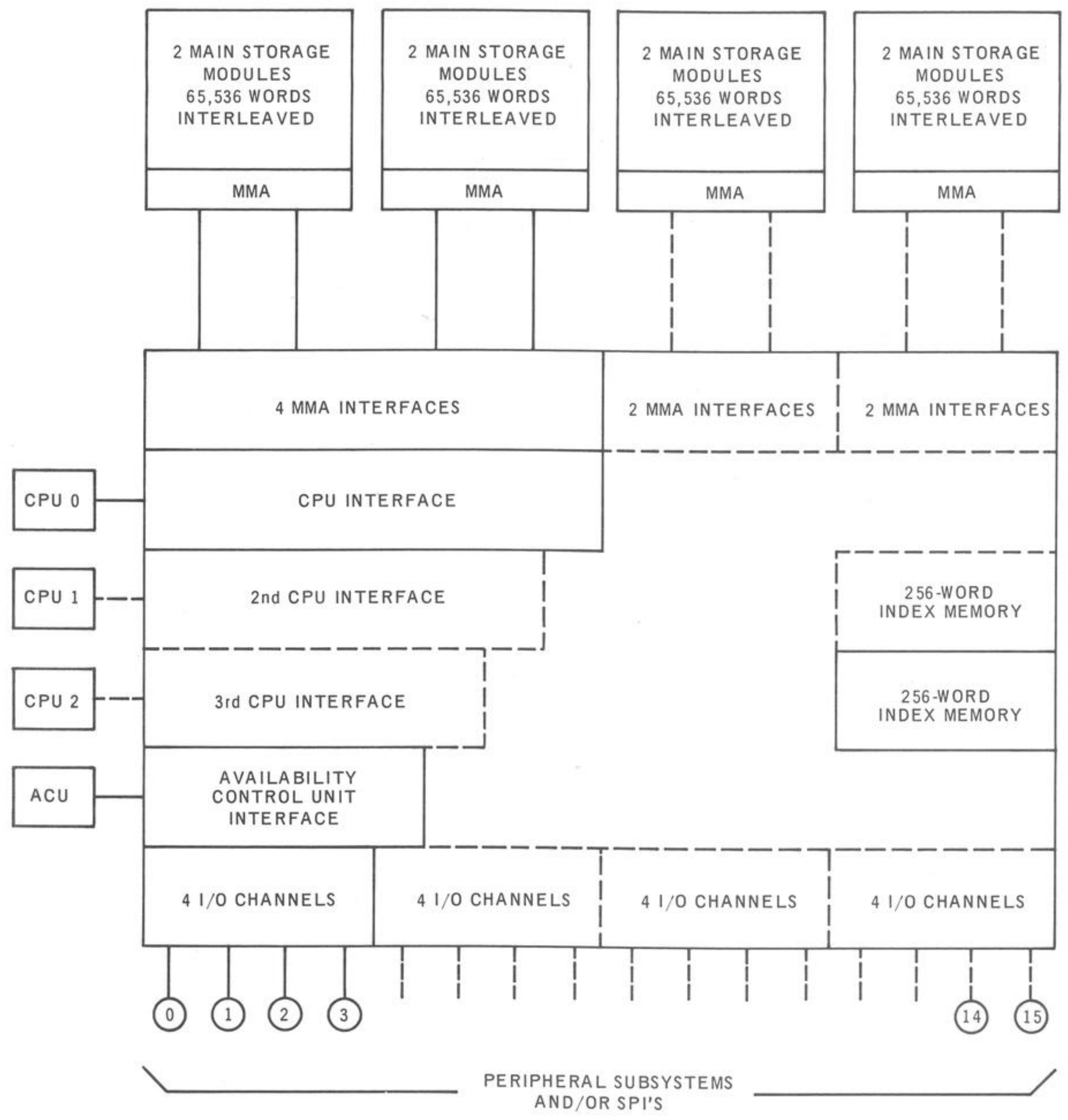
The multiprocessor system includes one or two Input/Output Controllers each of which controls its own group of peripheral subsystems (see Figure 6–1). Each provides a direct path between main storage and 4, 8, 12, or 16 high speed, bidirectional data channels. Each channel can transfer data at speeds up to 250,000 words per second. The IOC has an aggregate transfer rate of 1,333,000 words per second for all channels.

An I/O channel of the IOC performs all the functions that a processor I/O channel performs. Data transfers proceed by way of an IOC between a peripheral device and main storage, independent of the cyclic operation of the processor, thus allowing the processor more time for processing. No direct peripheral-to-peripheral (drum-to-printer) transfers are possible. All transfers must be buffered in main storage.

The IOC can interface with 1, 2, or 3 processors and has access to all of main storage by way of the Multiple Module Access units. However, rather than using the I/O access control registers of the processor, it has its own high speed index memory for buffer control.

## 6.2. POINTER REGISTERS

Access to the access control words in the index memory is controlled by pointer registers. These sixteen 6-bit pointer registers are assigned one to each I/O channel of the IOC. The program loads a given pointer register with the index memory address of an access control word. Thus the fixed relationship is between the peripheral subsystem and the pointer register, rather than between the subsystem and certain access control registers in the index storage.


2 MAIN STORAGE MODULES 65,536 WORDS INTERLEAVED
2 MAIN STORAGE MODULES 65,536 WORDS INTERLEAVED
2 MAIN STORAGE MODULES 65,536 WORDS INTERLEAVED
2 MAIN STORAGE MODULES 65,536 WORDS INTERLEAVED
MMA
MMA
MMA
MMA
4 MMA INTERFACES
2 MMA INTERFACES
2 MMA INTERFACES
CPU 0
CPU INTERFACE
CPU 1
2nd CPU INTERFACE
CPU 2
3rd CPU INTERFACE
ACU
AVAILABILITY CONTROL UNIT INTERFACE
256-WORD INDEX MEMORY
256-WORD INDEX MEMORY
4 I/O CHANNELS
4 I/O CHANNELS
4 I/O CHANNELS
4 I/O CHANNELS
0
1
2
3
14
15
PERIPHERAL SUBSYSTEMS AND/OR SPI'S

Solid lines indicate basic unit.
Dotted lines indicate optional expansion units.

*Figure 6–1. The Input/Output Controller*

## 6.3. ESI AND ISI

Like the input/output section of the CPU, the IOC operates in ESI (quarterword or halfword) or ISI modes, the mode being a field-installed option for each I/O channel.

The ISI mode is used for any I/O channels that are connected to such peripheral subsystems as magnetic tapes, magnetic drums, printers, and punched card equipment. Function word and data transfers for such subsystems make use of pointer registers (described below) which reference the appropriate ISI access control word in the index memory.

The ESI mode is used for data transfers to and from communications subsystems via the Communication Terminal Module Controller (CTMC) (see Section 8). The CTMC multiplexes data transfers to and from many communication lines through a single I/O channel on a controlled basis without disturbing the program sequence of the processor. When operating in ESI mode, the data flow for each communication line terminal is governed by its own data access control word in index memory.

The index memory stores both ESI and ISI access control words. It permits increased throughput by providing data chaining capabilities (scatter-read, gather write) for ISI subsystems. The first 64 words of index memory are reserved for ISI data access control words (both input and output) and external function access control words. These registers can serve the same purpose as the 16 Input Access Control Registers and the 16 Output Access Control Registers in the CPU.

The remaining 192 words in the basic IOC store the ESI data access control words for operation with CTMC subsystems. Expanding the index memory to 512 words provides an additional 256 ESI data access control words. These control words are the counterpart of the ESI data access control words held in main storage for use by the processor I/O section. Their use reduces the number of main storage references from three to one for each ESI data transfer.

## 6.4. COMMAND AND CONTROL WORDS

The Executive system schedules the program to be run on the system and allocates main storage, processor, IOC, peripherals, and running time for each program depending on its needs and its priority. Therefore, if a program requires a particular peripheral, the Executive system specifies the IOC and channel requirement. It also specifies a CPU to transfer a command packet from the main storage to the IOC. Once the CPU has transferred this command packet, the IOC performs the transfer independently.

The IOC accepts the following command and control words:

IOC Command Words

External Function Access [illegible]ntrol Words

Data Access Control Words

Data Chaining Control Words

Each of these is described briefly in the following paragraphs.

6.4.1. IOC Command Word

The command word specifies the type of transfer (processor to IOC, IOC to processor, or no transfer) and the function to be performed (activate input, output, or external function buffer). It also designates the IOC channel number and whether or not chaining is required.

| ZEROS | F | N | P | M | C | D | K | A |
|---|---|---|---|---|---|---|---|---|
| 35 30 | 29 27 | 26 23 | 22 | 21 | 20 | 19 18 | 17 9 | 8 0 |

F specifies one of eight functions.

N specifies one of 16 I/O channels.

P specifies whether a pointer register is required.

M specifies whether monitor interrupt is required, for ISI channels.

C specifies whether chaining of access control words is required.

D specifies direction of data transfers or no transfer.

K number of control words to be transferred.

A the address in index memory of the first access control word.

6.4.2. Data Chaining

Data chaining is the linking of a series of access control words to provide the IOC with the capabilities for scatter-read, gather-write operations. An ISI channel is placed in the chain mode when the F field of the IOC command word contains a function code of 1, 2, or 3, and the C field contains a 1. For ESI channels, since chaining is not required, the C field must always be 0. When the channel is in the chain mode, the pointer-register count is incremented by one each time the word count field of an access control word changes from one to zero. The IOC immediately reads the next index memory location specified by the pointer register. If the word count is not zero, it is the normal access control word for the next buffer area. A word count of zero indicates that this is an end-of-string word signalling the IOC that the end of that packet has been reached. This word directs the IOC either to send a terminate function code to the peripheral subsystem or to jump to a new packet of access control words in the index memory. The end-of-string word has the following format:

| ZEROS | WC | M | J | T | NOT USED | A |
|---|---|---|---|---|---|---|
| 35 | 33 18 | 17 | 16 | 15 | 14 6 | 5 0 |

WC Word Count. Always zero to indicate end-of-string.

M Monitor Interrupt.

J Initiates a jump to address A in index memory.

T Terminates output operations (for magnetic drum subsystems).

A Address in index memory of the first word of a new packet of data access control words.

6.4.3. External Function Access Control Word

This word is stored in the index memory. It specifies the location in main storage of the first function word. When the IOC receives a command packet, it transmits the function word to the peripheral subsystem on the I/O channel designated by the IOC command word. One or more function words may be transmitted to the subsystem. After completing the external function, the IOC places the channel in input or output mode as specified by the external function access control word.

| ZEROS | MON | OUT | IN | WC | A |
|---|---|---|---|---|---|
| 35 24 | 23 | 22 | 21 | 20 18 | 17 0 |

| | | |
|---|---|---|
| MON | Places channel in monitor mode | For ISI only |
| OUT | Places channel in output mode | For ISI only |
| IN | Places channel in input mode | For ISI only |
| WC | Specifies the number of external function words to be sent to the subsystem | |
| A | Address of the first function word to be sent to the subsystem | |

6.4.4. Data Access Control Word

This word specifies a data buffer in main storage. The ESI and ISI formats are the same as those used in the processor input/output section described in Section 5.

When the first data access control word is completed, the IOC starts on the second until no further chaining is specified. If no chaining is specified, the command packet consists of an IOC command word, an external function access control word, and one data access control word.

# 7. PERIPHERAL SUBSYSTEMS

## 7.1. AVAILABLE PERIPHERAL EQUIPMENT

Peripheral subsystems are attached to the 1108 processor or to the independent I/O controller through general purpose input/output channels, which have no restriction as to the manner in which peripheral subsystems may be attached. The governing factor for peripheral attachment is the transfer rate of the devices in the subsystem. Since the channels are numbered in order or priority, 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 1108 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.

The UNIVAC 1108 peripheral subsystems are:

High Performance Drums

FH-432 Magnetic Drum subsystem
FH-1782 Magnetic Drum subsystem

Mass Storage

FASTRAND II Mass Storage subsystem

Magnetic Tape

UNISERVO VI C Magnetic Tape subsystem
UNISERVO VIII C Magnetic Tape subsystem

Auxiliary Systems

Punched card subsystem (Reader/Punch)
High speed printer subsystem
UNIVAC 1004 online subsystem

Satellite System

Data communication systems

Communications Terminal Module Controller subsystem
Word Terminal Synchronous
Communications Terminal Synchronous
Data Communication Terminal – DCT 2000
Data Communication Subsystem – DCS–1
UNISCOPE Visual Communication Terminal

In addition to the standard UNIVAC 1108 subsystems, UNIVAC 1107 subsystems function with the 1108 system as well. Included in this category are:

UNISERVO IIA tape subsystem
UNISERVO IIIA tape subsystem
UNISERVO IIIC tape subsystem
UNISERVO IVC tape subsystem
FH-880 Magnetic Drum Subsystem

## 7.2 THE FLYING HEAD DRUMS

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 FH-432 (with an average access time of 4.3 milliseconds) to the large capacity (12.5 million alphanumeric characters) FH-1782 which provides extensive fast access storage that can be used for large data files that have to be referenced frequently.

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 half-revolution access time of 4.3 milliseconds (FH-432) or 17.0 milliseconds (FH-1782).

Each word in all 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 matches the contents of any drum area, up to the entire size of the subsystem, with a designated identifier word. Upon finding a match it supplies the address of the match or commences reading to main storage. This entire process is carries 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.

A new function has been incorporated in the control logic of the FH-432/1782 Subsystem to predict and reduce storage access time. This function enables the program to request the current angular position of the drum under the read/write heads on a particular drum unit.

The Input/Output handler can then select from the subsystem queue the data request that can be serviced fastest. It is possible to sequence drum requests and to make multiple accesses to a drum unit during a single revolution, instead of having to wait an average of half a revolution for each request.

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 multi-processor interfaces, a single- or dual-channel FH drum subsystem may be accessed by multiple processors. This not only provides a safeguard in case of failure but also enables all processors to access all drum units as common storage, so that all CPU's in a multiprocessor system can share a single major task, or tasks can be allocated to individual CPU's but with data storage being shared.

Any FH drum subsystem may operate with either one or two Control Units, using one or two input/output channels. Availability of two channels permits Read/Read, Read/Write, Write/Read and Write/Write operations simultaneously on any two drum units of the subsystem. If required, a Search function may be substituted for any of the Read functions. As an additional reliability measure, each control unit of a dual channel 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.

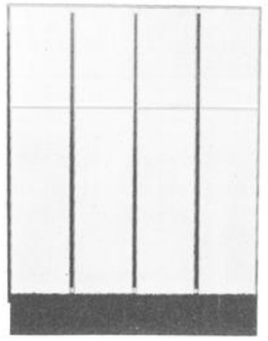
The UNIVAC 1108 Operating System is planned to use auxiliary drum storage instead of magnetic tapes as much as possible. This reduces manual handling and access and transfer time when compiling and assembling, and during basic batch input/output operations.

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 main storage utilization or operating inefficiency. The short access time of the FH-432 drum permits lesser-used control segments to be stored outside of main storage. They can then be read into a common overlay area only when required. This arrangement greatly reduces the amount of main storage required for the Executive System.

### 7.2.1. FH-432 Magnetic Drum Subsystem



The FH-432 Magnetic Drum subsystem is designed for single-channel operation, and is suited primarily to unit processor configurations. A minimum Flying Head 432 (FH-432) Magnetic Drum Subsystem includes three drums (786,432 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 262,188 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 2,048 words of data per 3 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 characters per second.

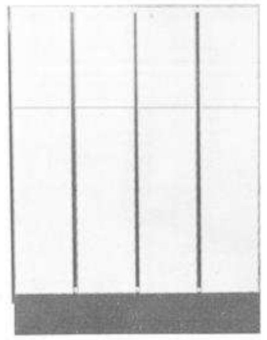
Up to nine FH-432 Magnetic Drums may be accommodated in a single subsystem, affording a maximum subsystem capacity of 2,359,296 words or 14,155,776 alphanumeric characters.

FH-432 units may be intermixed with FH-1782 units in the same subsystem to provide a powerful blend of ultrahigh speed and large capacity storage. This mixed subsystem is described in 7.2.3.

**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 | 7,200 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 per subsystem |
| NUMBER OF DRUMS PER SUBSYSTEM | 3 to 9 (14,155,776 characters maximum). |

## 7.2.2. FH-1782 Magnetic Drum Subsystem



The Flying Head 1782 (FH-1782) magnetic drum is identical to the FH-880 drum used on the UNIVAC 1107 Thin-Film Memory Computer except that the storage capacity is 2-2/3 times greater; this increase is achieved partly by an increase in the number of data tracks to 1,536 and partly by an increase in the recording density. Each track has its own read/write head, and average access time is unchanged at 17 milliseconds.

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

The increased recording density results in a character transfer rate 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 7.2.3.

CHARACTERISTICS

| | |
|---|---|
| STORAGE CAPACITY | 2,097,152 computer words of 36 data bits plus parity bits, or 12,582,912 alpha-numeric characters per drum. |
| AVERAGE ACCESS TIME | 17 milliseconds |
| DRUM SPEED | 1,800 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 |
| 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 (MAX.) | 8 (total of 100,663,296 characters.) |

## 7.2.3. FH-432/FH-1782 Magnetic Drum Subsystem

A valuable characteristic of UNIVAC 1108 drum subsystems 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 1108 storage configuration. An efficient blend can be made of high speed storage for rapidly required software, program segments, tables, and indices 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- and dual-channel versions to provide a hierarchy of auxiliary storage for both unit- and multi-processors. The dual-channel version includes two electrically 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 multi-processing.

## 7.3. FASTRAND MASS STORAGE SUBSYSTEM

The FASTRAND Mass Storage subsystem provides very large capacity random access storage. Great flexibility is provided by the availability of both a single- and a dual-channel subsystem and by the Fastband option, available on a unit basis. Each Fastband track has a permanently assigned read/write head, as distinguished from the ordinary track, which shares a head with a number of other tracks. Access to Fastband data is faster because there is no need for head positioning. FASTRAND II units include two large magnetic drums, which, like those used in the 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 over all the tracks, and 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, data layout, and programming.

An independent position control feature allows greater flexibility and decreases average access time. This is done in a multi-unit subsystem by concurrently pre-positioning the heads in each drum unit. Pre-positioning the heads saves time, because once the position instructions have been transmitted to one FASTRAND unit the Executive system can immediately initiate another operation on a different FASTRAND unit without waiting for completion of the positioning operation. Whenever the system reads from or writes on such a pre-positioned unit, the only time delay is for the latency and address circuit activation. These are the mechanical design features which contribute to the FASTRAND's operating speed.

In any effort to reduce processing time, offline search is an important advantage. In this operation the computer instructs the FASTRAND 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 computer and sends it the data. Also, all other functions of the FASTRAND permit the computer to continue its work while records are being read from or written on the drums.

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

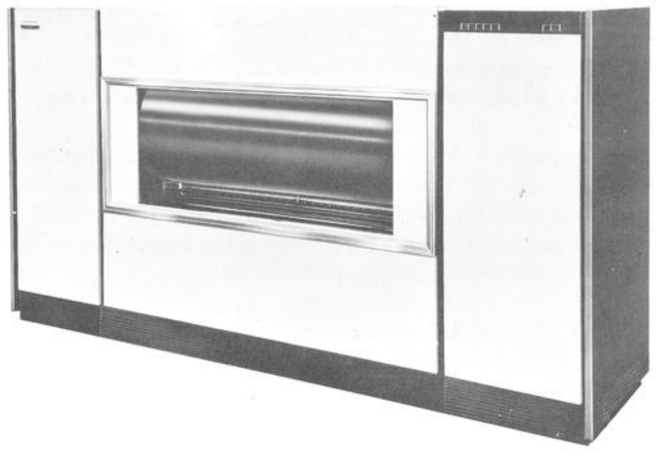
A single FASTRAND subsystem can accommodate up to eight FASTRAND II units. Any of the Input/Output channels of a UNIVAC 1108 Processor, or an Input/Output Controller, can accommodate a FASTRAND subsystem. Like the FH-432/1782 drum subsystem, either single or dual channel operation is possible, providing full-scale two-drum-unit simultaneity. An MPA enables access to the subsystem by one to four CPU's or Input/Output Controllers.

Two different dual-channel FASTRAND subsystems are available. They differ in the amount of hardware redundancy in the control units and therefore in their capability to perform simultaneous operations. The Type 5009-01 control unit provides economical dual-channel operation. Two control units which share some components are housed in one cabinet. In this system the FASTRAND units are organized into two banks and two simultaneous operations are possible so long as they call for drum units in different banks.

The dual-access FASTRAND subsystem includes two complete control units and therefore does not require the two-bank organization of FASTRAND storage units. The control units, being logically and electrically independent, provide the parallel data paths necessary to multiprocessing. Simultaneous operations can take place on any two FASTRAND units in the subsystem.

An optional feature called Fastband includes 24 additional tracks with fixed read/write heads. This provides rapid access (35 ms. average) and write lockout feature for data protection. By means of this lockout feature, the operator can manually prohibit writing on 1, 2, 4, 8, 16, 32, or all 192 tracks starting with the first track of each head.

## FASTRAND II MASS STORAGE UNIT



### CHARACTERISTICS

| | |
|---|---|
| STORAGE CAPACITY (PER UNIT) | 22,020,096 words or 132,120,576 alpha-numeric characters |
| AVERAGE ACCESS TIME | 92 milliseconds |
| RECORDING DENSITY | 1,000 bits per inch |
| TRACKS PER INCH | 106 |
| DRUM SPEED | 870 revolutions per minute |
| MOVEABLE READ/WRITE HEADS | 64 |
| CHARACTER TRANSFER RATE | 153,540 characters per second |
| WORD TRANSFER RATE | 25,590 words per second |
| **FASTBANDS (FIXED READ/WRITE HEADS) | 24 |
| FASTBAND AVERAGE ACCESS TIME | 35 milliseconds |
| FASTBAND STORAGE CAPACITY (PER UNIT) | 258,048 characters |
| **WRITE LOCKOUT PROTECTION | Yes |
| I/O CHANNELS | 1 or 2 per subsystem |
| NO. OF UNITS PER SUBSYSTEM | 8 (1,056,964,608 characters per subsystem) |

***Optional*

*NOTE: Addition of Fastband increases capacity by 258,048 characters per unit (2,064,384 if all 8 units include Fastbands).*

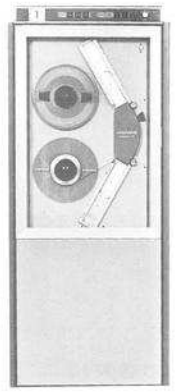
## 7.4. MAGNETIC TAPES

Magnetic tape subsystems for the 1108 system can be composed of UNISERVO VI C Magnetic Tape Units, UNISERVO VIII C Magnetic Tape Units, or any combination thereof, up to a total of 16 units. If only the UNISERVO VIII C Units are used, it is possible to double the transfer rate by using two UNISERVO VIII C Control Units, thus establishing simultaneous input and output capabilities.

Such a mixed magnetic tape subsystem affords a useful flexibility in combining high speed and economic medium speed tape units. Moreover, the seven- and nine-track unit options permits data recorded in traditional industry compatible form to be handled, and yet at the same time allows the upgrading of these records in line with the USASCII code and packed-decimal data.

The UNISERVO VIII C/VI C subsystem is available in single- and dual-channel configurations. A dual-channel configuration includes a UNISERVO VIII C control unit and a UNISERVO VIII C auxiliary control unit, each of which requires a separate I/O channel of a CPU or IOC. With this configuration, two read operations or a read and a write operation can take place simultaneously, while any of the tape units not being read or written on can be rewinding.

### 7.4.1. UNISERVO VI C Magnetic Tape Subsystem



The UNISERVO VI C Magnetic Tape Unit is a low cost unit having moderate speed and transfer rates for applications involving massive file passing, extensive sorting, or other processes for which high speed magnetic tape subsystems would be desirable.

A UNISERVO VI C subsystem can have up to 16 magnetic tape units connected to one or two input/output channels. Dual-channel operations permit a reading operation and a reading or writing operation to be performed simultaneously on any two magnetic tape units while all other tape units are rewinding.

Backward Read capability is standard on all units.

The master/slave concept is employed in the logic of the UNISERVO VI C Subsystem; circuitry has been built into one of the UNISERVO VI C units which will allow it to govern up to three other UNISERVO VI C units for certain electronic control functions. In a maximum subsystem of 16 units, there would be four master units and twelve slaves.

Data packing density is either 200, 556, or 800 characters per inch, as selected. The tape speed is 42.7 inches per second, giving maximum transfer rates of 8,540, 23,741 and 34,160 characters per second respectively.

The 800 CPU density is normally used, the 200 and 556 densities being used only for compatibility purposes. At this density more than 11,520,000 characters may be stored on a single reel in 600-character blocks.

Rewind takes place at 160 inches per second, enabling a full reel of 2,400 feet to be rewound in 180 seconds.

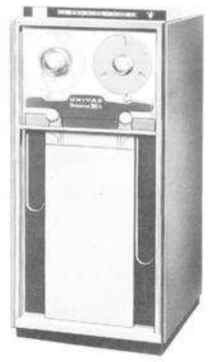
Data may be recorded in variable-length blocks under program control, with character and block (horizontal and vertical) parity. A read-after-write head allows immediate verification of all data written, and under the control of the Executive Input/Output Handler, repeated read and write operations are undertaken whenever read or write errors occur.

UNISERVO VI C Magnetic Tape units are fully compatible with IBM* 727, 729 Models I through VI, and 7330 units in seven-track mode, with IBM 2400 Series Models 1 through 3 units in nine-track mode, and with industry-compatible units produced by other manufacturers. The UNISERVO VI 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 | 8,500, 23,700 and 34,200 characters per second. |
| RECORDING DENSITY | 200, 556 and 800 characters per inch |
| TAPE SPEED | 42.7 inches per second |
| TAPE WIDTH | 0.5 inch |
| TAPE LENGTH | 2,400 feet |
| THICKNESS | 1.5 mils |
| BLOCK LENGTH | Variable |
| SPACE BETWEEN BLOCK | 0.75 inch (7 track) 0.60 inch (9 track) |
| TRACKS ON TAPE | 7 tracks: 6 data, 1 parity optional, 9 tracks: 8 data, 1 parity |
| MAXIMUM NUMBER OF UNITS IN SUBSYSTEM | 16 |
| STANDARD FEATURE | Backward read |
| PROCESSOR INPUT/OUTPUT CHANNELS | 1 or 2 |

*Trademark of IBM Corporation.

## 7.4.2. UNISERVO VIII C Magnetic Tape Subsystem



A UNISERVO VIII C subsystem can have up to 16 magnetic tape units, and can incorporate either one or two Control Units attached to one or two input/output channels for single or dual channel operator.

UNISERVO VIII C Units may be specified with seven- or nine-track mode. In seven-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 nine-track mode one parity and eight data bits are recorded in each frame across the width of the tape.

Data packing density 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 characters per second (in seven-track mode), or bytes per second (in nine-track mode).

Even higher transfer rates result if 6-bit characters are read or written in nine-track mode. This method of operation yields transfer rates of 32,000, 88,960, and 128,000 characters per second.

The rewinding rate is 240 inches per second; a full reel of 2,400 feet can be rewound in 120 seconds. The 800 frame per inch packing density will normally be used, the 200 and 556 densities being used only for 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 when the tape is moving forward only.

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 Magnetic Tape Units are fully compatible with IBM 727, 729 Models I through VI, and 7330 units in seven-track mode, and with IBM 2400 series Models 1 through 3 units in nine-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.

Two different UNISERVO VIII C tape subsystems are available: the UNISERVO VIII C/VI C subsystem and the fully simultaneous UNISERVO VIII C subsystem.

**CHARACTERISTICS**

| | |
|---|---|
| TRANSFER RATE | 24,000, 66,720 and 96,000 characters per second |
| RECORDING DENSITY | 200, 556 and 800 6-bit characters per inch |
| TAPE SPEED | 120 inches per second |
| TAPE WIDTH | 0.5 inch |
| TAPE LENGTH | 2,400 feet |
| THICKNESS | 1.5 mils. |
| BLOCK LENGTH | Variable |
| SPACE BETWEEN BLOCK | 0.75 inch (7 track)<br>0.6 inch (9 track) |
| TRACKS ON TAPE | 7 tracks: 6 data, 1 parity<br>Optional, 9 tracks: 8 data, 1 parity |
| UNITS PER CONTROL | 16 |
| STANDARD FEATURE | Backward Read |
| PROCESSOR INPUT/OUTPUT CHANNELS | 1 or 2 |

### 7.4.3. UNISERVO VIII C/VI C Subsystem

The UNISERVO VIII C/VI C Subsystem permits any combination of UNISERVO VI C and UNISERVO VIII C Magnetic Tape Units, up to 16, to be intermixed on a single or dual channel subsystem. A Shared Peripheral Interface may be attached to permit access from one to four processors.

The tape units in a mixed subsystem may have seven- and nine-track capability in nearly any combination of units. The only restriction is that a slave UNISERVO VI C unit may have the nine-track capability only if the associated master unit has that capability.

### 7.4.4. Fully Simultaneous UNISERVO VIII C Subsystem

The fully simultaneous subsystem includes two UNISERVO VIII C control units, each connected to a separate I/O channel of a CPU or IOC. With this dual channel configuration any two operations, including write/write, can be performed, in addition to rewinds. This effectively doubles the maximum subsystem transfer rate provided by a single-channel subsystem. It provides a redundant path to the tape units for multiprocessing systems.

The tape units in this subsystem require fully simultaneous features to provide the write/write capability. They may be either seven- or nine-track models. UNISERVO VI C units are not permitted.

With the addition of a Shared Peripheral Interface, up to four processors (CPU's and IOC's in any permitted combinations) can gain access to the subsystem under Executive System control.

## 7.5. UNIVAC 1004 SATELLITE SUBSYSTEM

The UNIVAC 1004 System is in itself a powerful processing unit with arithmetic, logical, and editing capabilities allied with modular core storage. Standard peripheral units are a 615 cpm Card Reader and a High Speed Printer, operating at 600 lpm with a 63 character set and a 132 character print line width.

Optional units are a second bank of 961 characters of core storage (with a character cycle time of 6.5 microseconds), a Card Punch (200 cpm), Paper Tape Reader (400 cps), Paper Tape Punch (110 cps), and a communications facility utilizing one of the Data Line Terminals.

A 1004 can be connected on-line to a single input/output channel of a UNIVAC 1108 System through a 1004 control unit to provide card reading, card punching, and printing capability. When 1004 peripheral units are used with the 1108 System, many units may be functioning on-line at the same time.

Refer to ''UNIVAC 1108 Executive Programmers Reference Manual,'' UP-4144 (current version) for details on hardware options and software.

The UNIVAC 1004 System may function as a remote data processor connected via a communications line to the UNIVAC 1108 System. This is achieved by attaching a Data Line Terminal (DLT) to the 1004 System. These terminals permit transmission at speeds of 2000, 2400 or 40,800 bits per second, dependent upon the type of DLT and communications facility employed. The interface with the UNIVAC 1108 System. is either through the 1108 Communications Terminal Module Controller (CTMC for 2000 or 2400 bits per second, or through a Communication or Word Terminal Synchronous (CTS or WTS) for up to 40,800 bits per second.

## 7.6. HIGH SPEED PRINTER SUBSYSTEM

The High Speed Printer Subsystem may use either of two high speed printers: the 700/922 LPM Printer or the 1200/1600 LPM Printer. Either printer can be attached to a printer control unit to form a subsystem, the higher speed unit requiring an adapter. The subsystem can be attached directly to an I/O channel of a CPU or an Input Output Controller, or indirectly through a Shared Peripheral Interface. The control unit includes a full line buffer (132 characters) for accumulating information before printing.

The program controls the printing operation by sending function words to the subsystem specifying the operation and number of lines to be spaced. The subsystem responds by generating program interrupts and supplying information regarding the status of the subsystem to the Executive System for analysis and action.

The speed of either printer varies from the lower rate for the full 63-character set to the higher rate for an abbreviated character set of 42 characters (A through Z, 0 through 9 and 6 special characters).

The Line Printer spaces paper at 20 inches per second.

**CHARACTERISTICS**

| | |
|---|---|
| PRINTING RATE | 700-922 LPM or 1200/1600 LPM |
| MAXIMUM CHARACTERS/LINE | 132 |
| MAXIMUM PRINTED CHARACTERS | 63 total (26 alpha, 10 numeric, 27 special characters) |
| PAPER SPACING AND SPEED | 0 to 63 spaces at 20" per sec. (or 33" per sec for higher speed unit) |
| HORIZONTAL SPACING | 10 characters/inch |
| VERTICAL SPACING | 6 to 8 lines/inch, single spaced manually selected |
| PAPER STOCK | Any standard sprocket-fed paper 4 to 22 inches wide; up to card stock in weight. |

## 7.7. PUNCHED CARD SUBSYSTEM

The Punched Card subsystem comprises a 900 cpm Card Reader and a 300 cpm Card Punch which are attached to the same Control Unit on a single input/ output channel of a CPU or Input/Output Controller.

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. Cards may be read in Fieldata code or in row or column binary.

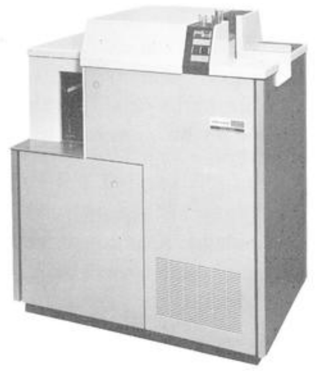
The Card Punch operates on a row-by-row basis and has an automatic check-read station. Incorrectly punched cards are recognized by the Input/Output Handler examining the status word upon the termination of the function, and such cards are passed to the error stacker. Under program control additional attempts to punch the data can be made, and if an unacceptable error rate is achieved, the job may be suspended for maintenance action. Cards may be punched in Fieldata code or in row and column binary.

### 7.7.1. Card Reader

***CHARACTERISTICS***

| | |
|---|---|
| CARD READING SPEED | 900 cards/minute |
| INPUT HOPPER CAPACITY | 3,000 cards |
| OUTPUT STACKER CAPACITY | 2,400 cards |
| REJECT STACKER CAPACITY | 100 cards |
| READ MODES | Fieldata, column binary, row binary |
| I/O CHANNELS | 1 shared with card punch |

7.7.2. Card Punch



| CHARACTERISTICS | |
|---|---|
| CARD PUNCHING SPEED | 300 card/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 |

# 8. DATA COMMUNICATIONS

## 8.1. GENERAL

There is a wide variety of methods for communicating with the UNIVAC 1108 System. Data transfer rates can vary widely, and many communication terminals can be multiplexed to one remote terminal which has direct high speed access to the processor.

## 8.2. UNIVAC 1108 COMMUNICATIONS SUBSYSTEM

The UNIVAC 1108 Communications Subsystem enables the UNIVAC 1108 System to receive and transmit data by way of any common carrier at any of the standard rates of transmission up to 4800 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 8–1, 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 processor I/O channel, multiplexing up to 16 CTM's to that channel.

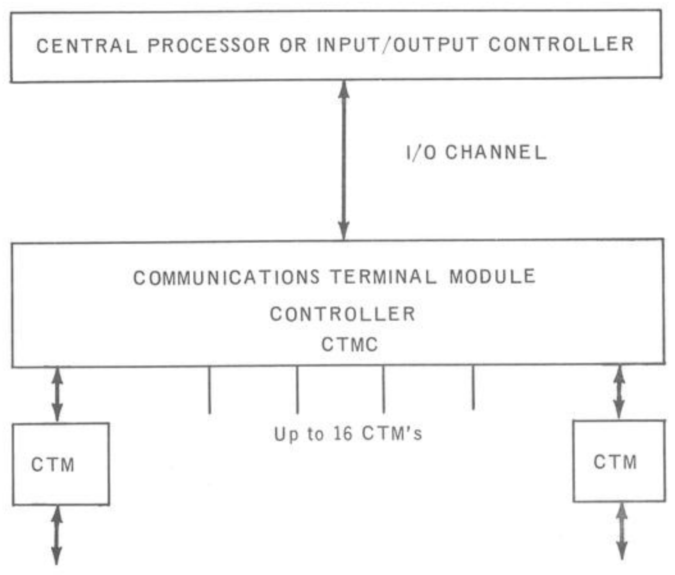

CENTRAL PROCESSOR OR INPUT/OUTPUT CONTROLLER
I/O CHANNEL
COMMUNICATIONS TERMINAL MODULE CONTROLLER CTMC
Up to 16 CTM's
CTM
CTM

*Figure 8–1. CTMC Subsystem*

There are three basic types of input and output CTM's: low speed (up to 300 bits per second), medium speed (up to 1600 bits per second), and high speed (2000 to 4800 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.

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 module enables 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 1600 BPS | | |
| High | To 4800 BPS | | |
| Dialing | Variable | Bit Parallel | 4 |
| Parallel Out | To 75 CPS | Timing Signal Bit Parallel | 8 |
| Parallel In | | | |

BPS = Bits per second; CPS = characters per second.

## 8.3. SYNCHRONOUS DATA TERMINALS

Each of the two types of synchronous data terminals discussed in this section links high speed (up to 40,800 bits per second) data circuits directly to an input/output channel without a control unit.

### 8.3.1. Word Terminal Synchronous

The Word Terminal Synchronous (WTS) is a central-site device that links a single high speed synchronous data communication line to a single I/O channel of the 1108 Processor. Though data on the line is bit serial, the WTS communicates with main storage a word at a time (six 6-bit characters per word) reducing the transfer time and the size of the buffer in main storage. However, the most significant advantage is that it reduces manipulation of data by the processor since it adds character and message parity to outgoing data and checks parity of incoming data. Upon detecting a parity error it generates an external interrupt. The WTS can handle data at speeds of 2000, 2400, and 40,800 bits per second.

The WTS allows the UNIVAC 1108 System to exchange data with a remote UNIVAC 1004 Processor. Such a configuration could be used for order entry, inventory control, and a wide variety of remote processing operations. Optional automatic dialing gives the WTS access to any station on the public network.

### 8.3.2. Communication Terminal Synchronous

Like the WTS, the UNIVAC Communication Terminal Synchronous (CTS) is a central site device that links a synchronous data communication line to a processor input/output channel, at rates of 2000, 2400, or 40,800 bits per second. (The synchronous rate is set according to the characteristics of the line with which it is used; see the table of Synchronous Data Terminal characteristics.) But unlike the WTS, the CTS communicates one character at a time, five, six, seven, or eight bits per character, instead of a word at a time.

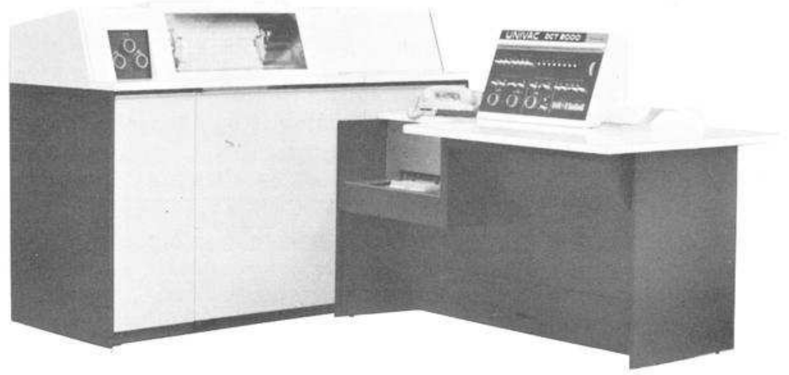
The CTS may optionally have the ability to dial remote locations automatically, and it may generate and check parity according to patchboard wiring.

A remote installation with equipment such as a UNIVAC 1004 System can, over the communication lines and through the CTS, have access to a UNIVAC 1108 System, thereby in effect affording the remote user the advantages of large-scale computer equipment. Conversely, the remote equipment can be used as an output terminal for the central facilities.

*CHARACTERISTICS*

| | WTS | CTS |
|---|---|---|
| SPEED | 2000, 2400, 40,800 bits per second | |
| COMMUNICATION FACILITIES | 2400 bits per second, or less: private voice lines. 40,800 bits per second or less: broad band transmission lines. | |
| CONTROL CODING | Field for control characters and codes denoting Synch, Start of Message, End of Message, and the like, selected by plugboard. | |
| I/O TRANSFER MODE | One word (six 6-bit characters) per transfer | One character (5, 6, 7 or 8 bits) per transfer |
| DATA CODING | 6 bits per character | 5, 6, 7 or 8 bits per character |
| I/O CHANNELS REQUIRED | One per WTS | One per CTS |

## 8.4. DATA COMMUNICATION TERMINAL (DCT) 2000



The UNIVAC Data Communication Terminal (DCT) 2000 is a combination printer and card reader/punch designed to transfer large quantities of data efficiently over voice-grade facilities. This terminal, tied into a network with computers, the UNIVAC 1004 Systems, 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 comprises 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 8–1).

| 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 8–1. DCT 2000 Field-Installable Options*

| 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 to USASCII<br>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)<br>2000 bits per second (switched telephone network) |
| TRANSMISSION CODE | USASCII<br>XS-3 (DLT compatible) |

## 8.5. UNISCOPE 300 VISUAL COMMUNICATION TERMINAL SUBSYSTEM

The UNIVAC UNISCOPE 300 Terminal is designed for applications requiring direct user interaction with the central system. These Terminals can be located at or near the site, or they may be operated as remote stations over standard data communication facilities.

The UNISCOPE Terminal itself includes a keyboard and CRT display, a core storage unit to store data as it is typed or received and displayed on the CRT, and control circuits. As the operator types a message on the keyboard, the data is accumulated in the storage unit and displayed on the CRT with a cursor marking the location of the next character to be typed. He can then make changes before releasing the message for transmission to the computer. Messages from the computer are similarly displayed and can be altered and returned to the computer by the operator.

### 8.5.1. The Keyboard

The keyboard includes alphanumeric keys, cursor controls and editing keys, and function keys. The alphanumeric section is similar to the standard electric typewriter in layout and operation. The character set has 56 symbols ordinarily, but up to 96 characters are available. The cursor controls can set the cursor to any point on the screen, so that deletions and changes can be made while composing or editing messages. Up to 40 different function keys can be added to the keyboard. The meanings of 35 of these function keys can be varied by means of overlays that fit over the group. The overlays are cards each of which has an edge formed according to an identifying code; when an overlay is in position, the coded edge causes the operation of some combination of seven switches controlling the significance of the function keys. On the face of the overlay, and appearing adjacent to the function keys when the overlay is in place, are markings to indicate the corresponding use of each key. One hundred and twenty-two different overlays can be used, enabling representation of as many as 4000 different functions. Typically, the different overlays can be used to identify stations, operators, applications, or security requirements. In addition to initiating a function, each function key displays a unique symbol on the CRT.

## 8.5.2. CRT Display

Sixteen lines of 64 characters each can be displayed on the 5 inch by 10 inch screen. The data is actually stored in a 1024 character core storage unit and is regenerated on the screen 60 times per second.

## 8.5.3. Subsystem Configurations

The subsystem can be used in single- and multi-station configurations.

The single-station terminal is completely self-contained. It interfaces directly with the data communications equipment through a standard EIA interface. Transmission rate of the line must be greater than 2000 characters per second. It can be used individually in a private line circuit or a number of them can be connected to a multipoint party line. In this latter arrangement the unit responds to a poll code from the central system.

The multi-station configuration provides a more economical arrangement where a large number of terminals is required. This configuration requires a multi-station control unit. The control unit is modular; it provides I/O message buffering, character generation, and control logic for up to 24 terminals of 1024-digit capacity or up to 48 terminals of 512-digit capacity. In this configuration each terminal is completely independent of all others. The terminals can be located as much as 1500 feet from the control unit.

Optionally the control unit can be equipped to communicate over two separate lines. This makes possible independent, concurrent communication (input and/or output) over the two lines; or, if desired; one line can be reserved as backup.

In another configuration two control units can be connected to the same set of terminals with one unit switched on and the other off to provide a backup control unit. The units can be switched on or off manually or by programming. By this means, one extra control unit can serve as a spare for as many as 48 sets of displays.

Polling techniques allow single-station and multi-station units to be mixed freely on one multipoint party line. This capability increases line utilization and significantly decreases line costs.

## 8.5.4. Reliability

In both single- and multi-station configurations the subsystem checks the character parity and message parity of incoming messages, and generates them for outgoing messages. Erroneous messages are retransmitted automatically on request. A simple message acknowledgment system ensures that no messages are lost or duplicated.

To further improve reliability, the central system can perform marginal tests on the storage unit.

## 8.5.5. Special Features

Several special features of the subsystem contribute especially to its use as an on-line device.

A pair of special function codes enable the program to insert or delete a line of copy. The insert function moves the line marked by the cursor and all lines below it down one line, the bottom line moving off the screen. It then inserts the new text. The delete function erases the line marked by the cursor and all lines move up one line, leaving the bottom line blank. By means of these functions the program can rearrange data on the screen with a minimum of new transmission.

Of particular interest is the ''roll and scroll'' technique implemented by use of the insert and delete functions. In this case text on the screen appears to roll up or down with the screen remaining filled. Typically, this is useful for scanning through large tables or other lengthy text that is too large for the screen. As soon as the required part of the table is located, the operator can stop the rolling by means of a function key.

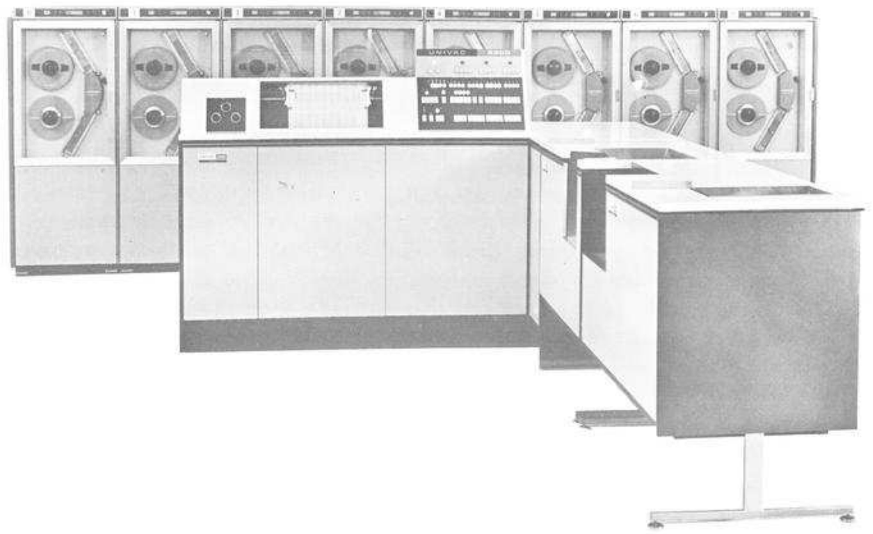
The capability of split-screen operation also increases the versatility of the terminal. This makes possible simultaneous, independent display of several messages. In this case an end-of-field symbol designates the end of each separate message. Line insert and delete functions can then be used on the messages separately and each one can be rolled and scrolled without interfering with the others.

For emphasis, specific messages or parts of messages from the system can be programmed to blink. In addition, unsolicited messages from the central system are accompanied by an audible signal. Such a message can be programmed for immediate display, overriding other outputs, or it may be withheld pending operator response.

***CHARACTERISTICS***

| | |
|---|---|
| DISPLAY | 1024 characters total (16 lines of 64 characters)<br>10 inch by 5 inch screen. |
| KEYBOARD | 56 to 96 different symbols<br>Editing Keys<br>Up to 40 function keys which, with overlays, produce more than 4000 functions. |
| ERROR CONTROL | Character and message parity.<br>All messages acknowledged. |
| DATA TRANSMISSION FACILITIES | More than 2000 characters per second.<br>Half or full duplex operation.<br>Party line via polling available.<br>Non-significant spaces suppressed. |

## 8.6. UNIVAC 9200/9300 REMOTE SUBSYSTEMS



The user of a 9200/9300 system can, without impairing to the slightest degree its capabilities as a self-contained data processor, demand the full and powerful facilities of the 1108 system. This is accomplished by use of the DCS-1 (Data Communications System) to link the 9200/9300 system to communications lines. The DCS-1 permits selection of communication speed for the most efficient use of the means of transmission; dial-switched voice line, private line, or broad-band line.

The 9200/9300 equipment has memory large enough and internal speeds fast enough to utilize communications lines to the fullest. Further more, the DCS-1 subsystem has provisions for error detection which are flexible enough to be adaptable to any reasonable requirements of the user.

When the 9200/9300 system is being used as a remote subsystem of an 1108 installation, the Executive software system will enable 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.

### 8.6.1. The UNIVAC 9200 System

The basic 9200 System is designed and priced for use in conjunction with modest sized punched card installations. But, because it uses advanced hardware innovations such as plated-wire memory and integrated circuits and because it includes such software benefits as internal programming and a full set of software packages, it compares favorably with systems two or three times larger.

The basic system includes a versatile processing unit with 8,192 to 16,384 byte memory, a 400 cpm card reader, a 75 to 200 cpm card punch and a 250 to 500 lpm printer.

Up to eight additional I/O subsystems can be connected to the system by means of an I/O Multiplexer. These subsystems include such peripheral equipment as the UNIVAC 1001 card controller, the UNISCOPE 300 Visual Communications Terminal, and the Data Communications Subsystem, DCS-1.

Thus the 9200 system is modular and can be easily expanded to the 9300 configuration. Because of programming compatibility this expansion from the smaller system to the larger requires no reprogramming.

### 8.6.2. The UNIVAC 9300 System

The 9300 System is compatible with the 9200 system; it offers faster computing, larger and faster memory, faster punched card handling and printing, and the UNISERVO VI C Magnetic Tape Subsystem. Operating speeds are summarized in the Table of characteristics below.

All card I/O operations, printing, and one read or write tape operation can be performed simultaneously with processing. With a dual-channel tape subsystem using two control units the system can perform simultaneous read, write, and compute operations.

Expanded tape systems enable an even wider range of processing capabilities. Two or three concurrent tape operations such as card-to-tape, tape-to-print, and tape sorting are possible with the 9300 tape systems.

*CHARACTERISTICS*

| | UNIVAC 9200 | UNIVAC 9300 |
|---|---|---|
| SYSTEM ORIENTATION | Card | Card/Magnetic Tape |
| BASIC MEMORY<br>MAXIMUM MEMORY | 8192 bytes<br>16,384 bytes | 8192 bytes<br>32,768 bytes |
| MEMORY CYCLE TIME | 1200 nanoseconds | 600 nanoseconds |
| ADD (DECIMAL) INSTRUCTION TIME (TWO 5-DIGIT FIELDS) | 104 microseconds | 52 microseconds |
| CARD READER - BASIC READER<br>1001 CARD CONTROLLER | 400 cpm<br>1000/2000 cpm | 600 cpm<br>1000/2000 cpm |
| CARD PUNCH - COLUMN<br>ROW | 75-200 cpm<br>Not used | 75-200 cpm<br>200 cpm |
| READ/PUNCH | Not used | optional |
| PRINTING SPEED | 250 lpm | 600 lpm |
| VARIABLE PRINTING SPEED (optional) | 250/500 lpm alphanumeric<br>500 lpm numeric | Not used |
| NUMERIC PRINTING (optional) | Not used | 1200 lpm |
| OVERLAPPED I/O UNITS | Standard | Standard |
| MULTIPLEXER CHANNEL RATE | 85,000 bytes/sec | 85,000 bytes/sec |
| REGISTERS | 8 for processor functions<br>8 for I/O functions | 8 for processor functions<br>8 for I/O functions |
| MAGNETIC TAPE RATE | Not used | 34,160 bytes/sec |
| SIMULTANEOUS TAPE READ, WRITE, AND PROCESS | Not used | optional |
| DATA TRANSMISSION RATE THROUGH DCS-1 | 2000, 2400 or 50,000 bits per second | 2000, 2400 or 50,000 bits per second |

## 8.6.3. The UNIVAC Data Communication Subsystem, DCS-1

The DCS-1 subsystem provides communication capability for the 9000 series computer. Connected to a multiplexer channel, it enables synchronous data transmission at speeds up up to 50,000 bits per second between the 9200/9300 computers and the 1108 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 subsystem is modular, permitting field modifications as demands for various options arise. As communications 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-1 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 memory size.

- Compatibility

  The DCS-1 is compatible with a number of UNIVAC systems including 1108, 494, DCT-2000, and the 1004.

*CHARACTERISTICS*

| | |
|---|---|
| SPEED AND FACILITIES | Dial voice lines - 2000 bits per second<br>Private voice lines - 2400 bits per second<br>Broad-band lines - 50,000 bits per second |
| DATA CODING | Five to eight levels, plus parity. |
| CHECKING | Odd or even message and character parity.<br>Longitudinal redundancy check is optional. |
| MULTIPLEXER CHANNELS REQUIRED | One per subsystem. |

# 9. PROGRAMMED SYSTEMS SUPPORT

## 9.1. AVAILABLE SOFTWARE

The programmed systems support (software) provided with the UNIVAC 1108 system has been designed to meet the total computing requirements of the advanced users of today. The requirements and demands of the large-scale user have grown considerably in the past, and it is in response to these changing requirements that Univac offers with the 1108 computer a software system designed to meet today's requirements and to allow the change and growth required to meet tomorrow's challenge. The degree of effective utilization of any computing system is in direct proportion to the scope and versatility of the software. With the 1108 system, Univac has combined many years of experience in multiprogramming and communication oriented systems to provide a system that is easy to operate and easy to use, yet one which ensures user program integrity in a demand-response environment.

The UNIVAC 1108 computer system includes a full, complete set of software, ranging from high level language compilers to basic mathematical functions. The major software items are:

- The Executive System
- The Assembler
- FORTRAN V
- Conversational FORTRAN
- LIFT – FORTRAN II to FORTRAN V Translator
- COBOL (Extended and Revised)
- ALGOL
- SORT/MERGE
- Application Programs

The entire 1108 software system, based upon the FH-432 and FH-1782 magnetic drums with their rapid access and high transfer rate, and backed up by mass storage, can give unsurpassed performance.

## 9.2. THE EXECUTIVE SYSTEM

To take full advantage of the speed and hardware capabilities of the UNIVAC 1108 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 1108 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 1108 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 1108 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 reduce the need for operator intervention and decisions.

#### 9.2.1.2. Demand Processing (Time-Sharing)

Complementing the batch processing capabilities of the 1108 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.

#### 9.2.1.3. Real Time

The Executive System is also designed to be applicable to programs which have real time requirements. The Communications Terminal Module Controller, 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

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

It should be noted that batch, demand, and real time programs are processed concurrently by the Executive System, whenever sufficient storage facilities are available; hence, the user of any one mode experiences little variation in his turn around time, regardless of the proportionate mix with other types of processing.

## 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 1108 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 1108 Executive System component that controls the sequencing, setup, and execution of all runs entering the 1108 System. 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 CPU dispatching. Runs entering the 1108 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 main storage is appropriate, the Dynamic Allocator initiates "storage swaps". This involves writing one program on drum storage and replacing it temporarily in main storage with another program. Such action is taken only to provide reasonable response time to remote demand-processing terminals.

The CPU Dispatching routine is a third level of scheduling; it selects among the various tasks currently occupying central storage whenever it is appropriate to switch the commitment of the CPU from one task to another. Under normal circumstances, a batch program is allowed to use the CPU 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 the CPU.

### 9.2.3.4. Storage Compacting

Certain 1108 hardware features make feasible the dynamic relocation of programs residing in central storage – a necessity for effective multiprogramming. At program termination, the assigned storage is returned to the pool of available central 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 store, 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 1108 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 1108 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 to inform 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 1108 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 operator consoles or as few as one, depending on the complexity and layout of the installation.

The 1108 hardware system has as standard equipment a display console to enhance operator-system communications. The advantages of a visual display to the operator are obvious and the possible display functions endless. 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 1108 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 input/output channels and peripheral equipment attached to the 1108, including UNISERVO magnetic tape units FH-432 and FH-1782 drums, FASTRAND Mass Storage, card readers, printers, and communication line terminals.

### 9.2.3.10. Auxiliary Processors

The System includes a set of auxiliary processors which perform functions that complement those of the source language processors such as FORTRAN, COBOL and Assembler. This set of processors includes the Collector for linking relocatable subprograms, the Procedure Definition Processor for inserting and modifying procedure definitions in a library, the ELT Processor used to insert Elements, and the DATA Processor to introduce data descriptions. A comprehensive set of library elements complements these processors.

### 9.2.3.11. Utilities

Included within the Utilities Section of the Executive System are diagnostic routines, program file manipulation routines, file utility routines, and cooperative routines which aid the user by performing such functions as reading cards, printing line images, transferring files from device to device, and carrying out housekeeping functions required for file-residence on mass storage devices.

### 9.2.3.12. Processor Interface Routines

The Processor Interface Routines provide a simple standard interface for all processors within the system. Complete facilities are provided for the input of source-language statements and the output of the resulting relocatable binary code.

### 9.2.3.13. System Setup

The System Setup section of the Executive System provides an installation with a means of generating a system tailored to its particular needs, and for subsequently maintaining this system with a minimum of effort.

## 9.3. THE ASSEMBLER

The Assembler translates a symbolic language composed of brief expressions to machine-language relocatable object coding for the UNIVAC 1108 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.

### 9.3.1. Symbolic Coding Format

In 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.

### 9.3.2. 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 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.

### 9.3.3. Additional Features

Facilities for interprogram communication permit separately assembler 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. 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 USASI 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 FORTRAN V Programmers Reference Manual, UP-4060, (current revision).

### 9.4.1. 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.

**IBM is a proprietary designation of the IBM Corporation.*

- The ABNORMAL statement permits increased optimization of object programs. Where common sub-expressions occur within a program unit, it is obviously desirable to evaluate each sub-expression only once. Where the common sub-expressions 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 sub-expressions 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 at all in a program, all function references except library functions are considered ABNORMAL and are re-evaluated 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($a_1$, ..., $a_n$) = 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 ($a_1$, $a_2$...), type ($a_1$, $a_2$, ...) 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 via 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 1108 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 (i.e. SQRT, DSQRT or CSQRT for REAL, DOUBLE PRECISION or COMPLEX arguments respectively.)

- The 1108 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 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 wth 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 '$c_1$,$c_2$,..., $c_i$' 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 & (2–8 punch) on call to a subroutine or function subprogram to indicate the transmission of a statement number.
- 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. Compiler Organization

The 1108 FORTRAN V source language processor accepts FORTRAN statements and produces a highly efficient relocatable object code element. FORTRAN, like all other UNIVAC 1108 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 intention was to create a compiler which, while quite rapid as a processor, would produce 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; a) the elimination of redundancies in loading of index and arithmetic registers, b) the recognition of common sub expressions from previous statements, and c) the removal from a loop of those 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.

## 9.5. CONVERSATIONAL FORTRAN V

An important element of the programmed systems support provided with a UNIVAC 1108 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 non-sequential 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.

### 9.5.1. 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.

### 9.5.2. 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 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.

### 9.5.3. Conversational Processor and the Executive System

The conversational 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.

### 9.5.4. 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 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 U.S.A. Standards Institute (USASI) FORTRAN. It is also defined by FORTRAN V and any programs constructed by the 1108 Conversational FORTRAN compiler may also be compiled by the batch compiler.

## 9.6. LIFT, FORTRAN II TO FORTRAN V TRANSLATOR

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.

## 9.7. COBOL

1108 COBOL is based on the Department of Defense publication ''COBOL Edition 1965''. The 1108 COBOL accepts statements written in the COBOL language as adapted for the 1108 system and produces a program ready for execution. For further information, consult the ''UNIVAC 1108 COBOL Programmers Reference Manual'', UP-4048 (current revision).

The major features implemented on the 1108 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 1108 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.

### 9.7.1. The Processor

The COBOL processor is a six-phase compiler operating within the UNIVAC 1108 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 so that it cannot easily be executed. This type of diagnostic is identified by the word ERROR* preceding the actual message.

### 9.7.2. 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 1108 processor such as FORTRAN V.

Overpunched Sign Convention – Tapes and cards prepared in the overpunched sign convention can be processed.

## 9.8. ALGOL

The ALGOL language allows the mathematician or engineer to prepare programs for the UNIVAC 1108 without the necessity of becoming familiar with the details of the internal machine operation. The 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.

The basis for the UNIVAC 1108 ALGOL is the "Revised Report on the Algorithmic Language, ALGOL 60" (Communications of the ACM, Vol. 6, January 1963, 1–17).

UNIVAC 1108 ALGOL is an extended hardware representation of ALGOL 60 designed to employ the UNIVAC 1108 processor 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.

## 9.9. SORT/MERGE

The UNIVAC 1108 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 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 central 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.

## 9.10. MATHEMATICAL FUNCTION PROGRAMS

The UNIVAC Software System includes an extensive collection of basic mathematical subroutines and functions. This collection includes all of the standard FORTRAN functions and has been expanded by over 50% to give the programmer a more complete coverage of the often used mathematical routines. Each of these mathematical routines has been carefully developed to offer the programmer maximum accuracy and range with a minimum routine size and executive time. These routines are available to each of the program languages, FORTRAN V, Assembler, COBOL (through the use of the ENTER verb option), and ALGOL. One group of routines, the series of exponentiation routines (NEXPi), are automatically referenced when the FORTRAN V Source Program indicates exponentiation with the operator **, and inline exponentiation is not feasible.

The various mathematical function programs are:

1. Library Functions
2. The FORTRAN Built-in Function
3. The Exponentiation Functions

These routines are available to the processors in different manners. For example, the routine SQRT provides a single precision square root of the argument (x). To utilize this routine the processors employ the following calling sequences:

- FORTRAN V Calling Sequence:
  ROOT = SQRT (X), when X is a FORTRAN V real variable.

- ASSEMBLY Calling Sequence:

```
a         LMJ       B11,SQRT
a + 1      +        Address of X
a + 2      +        normal return
```

  The result is left in A0.

- COBOL Calling Sequence:
  ENTER SQRT REFERENCING X

## 9.11. APPLICATION PROGRAMS

The UNIVAC 1108 System has an extensive library of application programs and subroutines. The major application programs such as Linear Programming (LP), Automatically Programmed Tools (APT III), and PERT TIME/COST are briefly described in the following paragraphs while others are simply mentioned by titles. This is by no means an exhaustive list of programs or subroutines. It is meant only to point out the many types of application programs available for UNIVAC 1108 System.

### 9.11.1. Linear Programming System

Linear programming has become one of the most useful and frequently used operations research techniques in manufacturing and transportation industries. In the production and distribution of products, LP provides a solution to minimize costs or maximize profits. The LP System developed for the UNIVAC 1108 System embodies the latest advances in computer technology with the most powerful algorithm to date. The algorithm employs the "product form of the inverse" method and is improved with an advanced path selection technique. The package is coded in FORTRAN V and Assembly Language.

The more prominent features of the 1108 LP System are as follows:

1. The System can accommodate 4094 rows.
2. The speed and random access properties of magnetic drums place the system at a distinct advantage over tape and disc handling procedures.
3. Both single precision and double precision computations are available. The selection may be manual or automatic.
4. The control language of this System is far superior to any other existing LP System. It is an interpretive control language. The sophisticated user may use macros in constructing his command string to implement the System. On the other hand, the average user may still execute his basic LP problem without detail knowledge of the control language.
5. The LP System is imbedded in the Executive System. This enhances the System as a powerful model builder. Matrix builders and output analyzers may be attached to the LP System to form corporate models. The entire model may be optimized and re-optimized in cycles in one computer run. Nonlinear programming may be accomplished by solving approximated linear functions in each cycle.

In addition to the above features, the 1108 LP System contains the flexibilities existing in other LP Systems. Vector levels can be specified and coefficients may be modified. Long LP runs can be split with restart procedures. Post-optimal parametric programming or a complete tableau can be obtained. The final output includes the objective function value, optimal basis, vector levels, and reduced costs.

### 9.11.2. APT III

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 participated in the original writing of APT III and has been a member of the APT long-range program from the beginning. Numerical control specialists are continually working to keep the UNIVAC 1108 APT program in the forefront of the art. As implemented on the UNIVAC 1108 System APT III will continue to conform to the latest APT long range program specifications.

### 9.11.3. PERT

The UNIVAC 1108 PERT/COST System is a generalized applications program based upon the framework provided by the ''DOD/NASA Guide to PERT/COST System Design''. The DOD/NASA design is based upon the concept 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 UNIVAC 1108 PERT System adheres rigorously to the DOD/NASA design and satisfies the purposes of PERT/COST concept.

Many government agencies and contractors are currently processing PERT/TIME data on existing systems. PERT/COST is relatively new and will be subject to the inevitable modifications that will result from its initial pilot tests by DOD. In order to lessen the impact of these changes upon existing PERT/TIME programs and to provide for efficient integration of time and cost data, a modular design was adopted.

The modular structure of the program permits separate processing of the time networks and of the work package costing structure while simultaneously providing for integrated time and cost reporting. The PERT/TIME Module of the UNIVAC 1108 PERT/COST System accepts as input a deck of cards describing the PERT network. The cards are processed and used to update the PERT/TIME Master File. Network computations are performed and the time reports are generated. The PERT/COST Module accepts the cost breakdown structure, actual and estimated costs for the project work packages, and a table of labor rates and applicable overhead percentages. Cost data is accumulated up through the cost breakdown tree, integrated with the time information, and the required cost reports are generated.

## 9.12. MATH-PACK

MATH-PACK provides the UNIVAC 1108 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 fourteen 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.

Appendix D lists all of the MATH-PACK subprograms.

## 9.13. STAT-PACK

STAT-PACK provides the UNIVAC 1108 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 thirteen 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.

Appendix E lists all of the STAT-PACK subprograms.

# APPENDIX A. NOTATIONAL CONVENTIONS

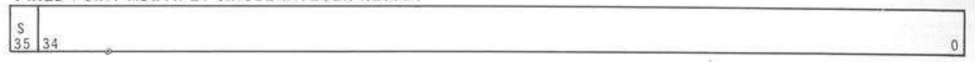
Abbreviations and symbols frequently used in the description of the instruction repertoire are given below:

| | |
|---|---|
| ( ) | Contents of register or address within parentheses. |
| $(\ )'$ | Complement of contents of register or address. |
| $\|(\ )\|$ | Absolute value or magnitude. |
| $(\ )_{17-00}$ | Subscripts indicate the bit positions involved. A full word is normally not subscripted. Subscripts are also used to designate octal or decimal notation. |
| ( )c | Floating point biased exponent. |
| ( )f | Final contents. |
| ( )i | Initial contents. |
| ( )m | Floating point fixed point part. |
| ( )j | j-designated portion. |
| f | Function code. |
| j | Partial word designator or function code extension. |
| a | Arithmetic register designator. In input/output instructions, "a" designates an I/O channel. |
| A | Arithmetic Register. |
| x | Index register designator. |
| $x_a$ | Index register designator in a-field. |
| X | Index Register. |
| $X_a$ | Index Register specified by coding $x_a$. |
| Xm | Modifier portion of an index register. |
| Xi | Increment portion of an index register. |
| r | Same as $r_a$. |
| $r_a$ | Designator specifying an R Register. It is coded in the a-designator position of an instruction word. |
| R | R Register. |
| $R_a$ | R Register specified by coding $r_a$. |

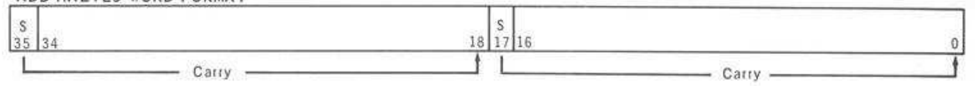
| | | |
|---|---|---|
| u | The base address of the operand (or the actual operand) as coded in u-field of an instruction. | |
| U | The effective address or value of the operand after application of indexing and indirect addressing. | |
| $U_d$ | Destination address. | |
| $U_s$ | Source address. | |
| h | h-designator of the instruction word. A value of 1 specifies incrementation of an index register. | |
| i | i-designator of the instruction word. A value of 1 specifies indirect addressing. | |
| PSR | Processor State Register. | |
| BI | I-Base Storage Block Number. | PSR Base |
| BS | Program Effective Switch point. | Relative Addressing |
| BD | D-Base Storage Block Number. | Fields |
| SLR | Storage Limits Register. | |
| CSR | Channel Select Register. | |
| P | Program Address Register. | |
| 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

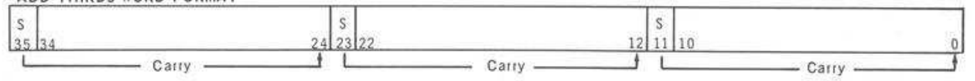
FIXED-POINT MULTIPLY SINGLE INTEGER RESULT



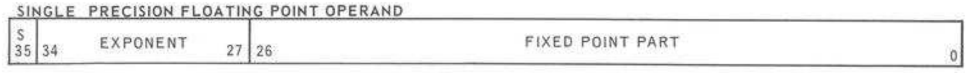
ADD-HALVES WORD FORMAT



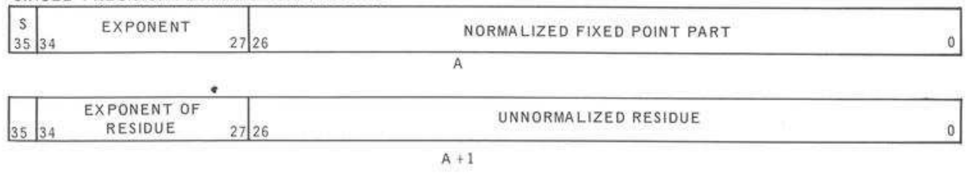
ADD-THIRDS WORD FORMAT



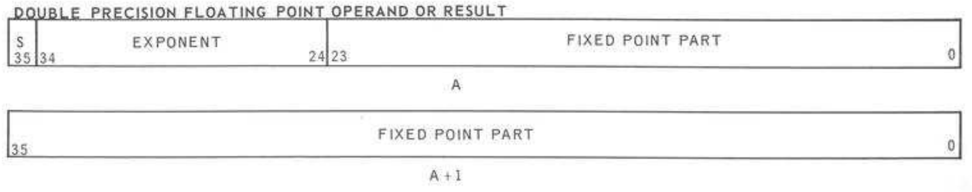
SINGLE PRECISION FLOATING POINT OPERAND



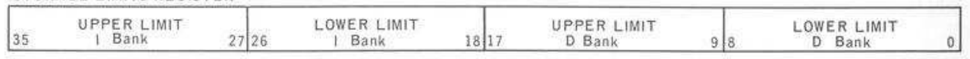
SINGLE PRECISION FLOATING POINT RESULT



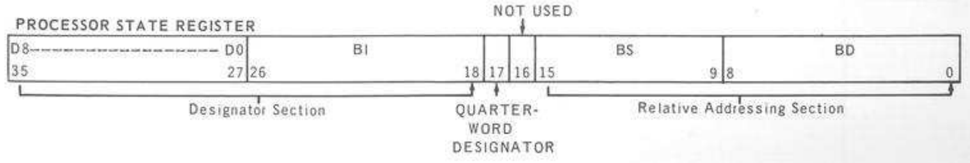
DOUBLE PRECISION FLOATING POINT OPERAND OR RESULT



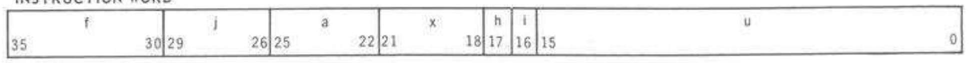
STORAGE LIMITS REGISTER



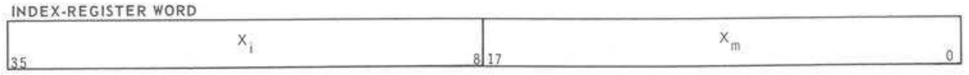
PROCESSOR STATE REGISTER



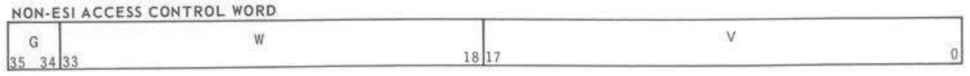
INSTRUCTION WORD



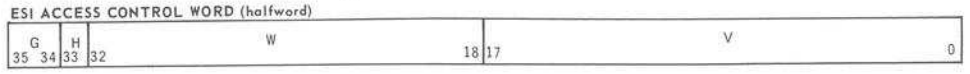
INDEX-REGISTER WORD



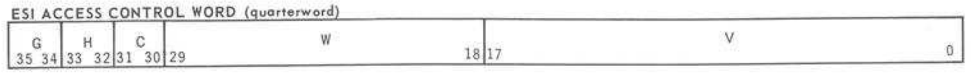
NON-ESI ACCESS CONTROL WORD



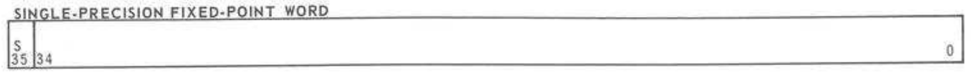
ESI ACCESS CONTROL WORD (halfword)



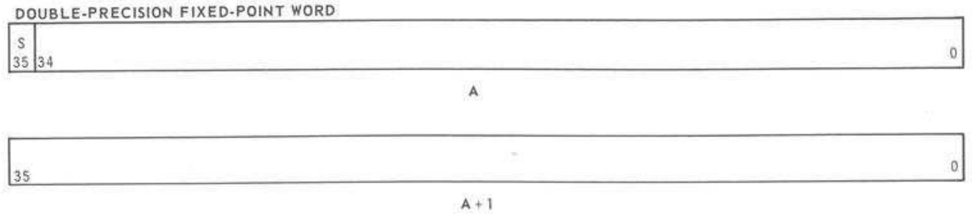
ESI ACCESS CONTROL WORD (quarterword)



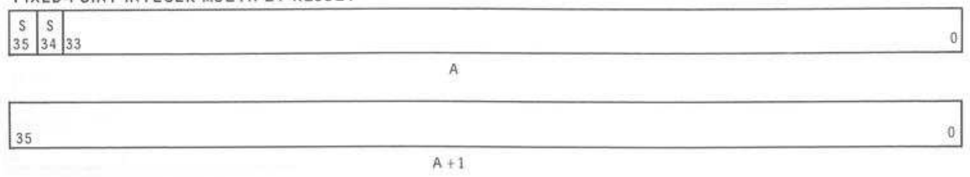
SINGLE-PRECISION FIXED-POINT WORD



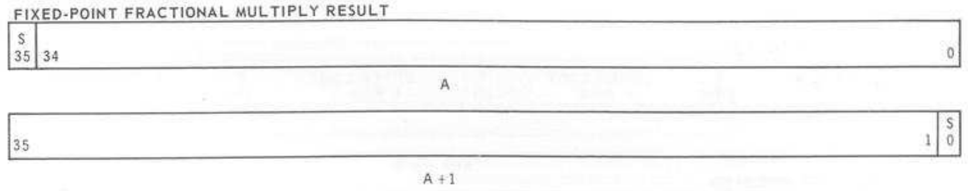
DOUBLE-PRECISION FIXED-POINT WORD



FIXED-POINT INTEGER MULTIPLY RESULT



FIXED-POINT FRACTIONAL MULTIPLY RESULT



# APPENDIX C. INSTRUCTION REPERTOIRE BY NUMERIC CODE

| Instruction Code | | Mnemonic | Instruction | Description | Execution Time * in μsec. |
|---|---|---|---|---|---|
| f | j | | | | |
| 00 | – | – | Illegal Code | Interrupt to $241_8$ | – |
| 01 | 0-15 | SA | Store A | $(A) \rightarrow U$ | .75 |
| 02 | 0-15 | SN SNA | Store Negative A | $-(A) \rightarrow U$ | .75 |
| 03 | 0-15 | SM SMA | Store Magnitude A | $\lvert(A)\rvert \rightarrow U$ | .75 |
| 04 | 0-15 | SR | Store R | $(R_a) \rightarrow U$ | .75 |
| 05 | 0-15 | SZ | Store Zero | Zeros $\rightarrow U$ | .75 |
| 06 | 0-15 | SX | Store X | $(X_a) \rightarrow U$ | .75 |
| 07 | – | – | Illegal Code | Interrupt to $241_8$ | – |
| 10 | 0-17 | LA | Load A | $(U) \rightarrow A$ | .75 |
| 11 | 0-17 | LN LNA | Load Negative A | $-(U) \rightarrow A$ | .75 |
| 12 | 0-17 | LM LMA | Load Magnitude A | $\lvert(U)\rvert \rightarrow A$ | .75 |
| 13 | 0-17 | LNMA | Load Negative Magnitude A | $-\lvert(U)\rvert \rightarrow A$ | .75 |
| 14 | 0-17 | AA | Add to A | $(A) + (U) \rightarrow A$ | .75 |
| 15 | 0-17 | ANA | Add Negative to A | $(A) - (U) \rightarrow A$ | .75 |
| 16 | 0-17 | AM AMA | Add Magnitude to A | $(A) + \lvert(U)\rvert \rightarrow A$ | .75 |
| 17 | 0-17 | ANM ANMA | Add Negative Magnitude to A | $(A) - \lvert(U)\rvert \rightarrow A$ | .75 |
| 20 | 0-17 | AU | Add Upper | $(A) + (U) \rightarrow A + 1$ | .75 |
| 21 | 0-17 | ANU | Add Negative Upper | $(A) - (U) \rightarrow A + 1$ | .75 |
| 22 | 0-15 | BT | Block Transfer, repeat | $(X_x + u) \rightarrow (X_a + u)$, repeat | 1.50 + 1.5K |
| 23 | 0-17 | LR | Load R | $(U) \rightarrow R_a$ | .75 |
| 24 | 0-17 | AX | Add to X | $(X_a) + (U) \rightarrow X_a$ | .75 |
| 25 | 0-17 | ANX | Add Negative to X | $(X_a) - (U) \rightarrow X_a$ | .75 |
| 26 | 0-17 | LXM | Load X Modifier | $(U) \rightarrow X_{a_{17-00}}$ | .875 |
| 27 | 0-17 | LX | Load X | $(U) \rightarrow X_a$ | .75 |
| 30 | 0-17 | MI | Multiply Integer | $(A) \times (U) \rightarrow A, A + 1$ | 2.375 |
| 31 | 0-17 | MSI | Multiply Single Integer | $(A) \times (U) \rightarrow A$ | 2.375 |
| 32 | 0-17 | MF | Multiply Fractional | $(A) \times (U) \rightarrow A, A + 1$ | 2.375 |
| 33 | – | – | Illegal Code | Interrupt to $241_8$ | – |

*See notes at end of table.

| Instruction Code | | Mnemonic | Instruction | Description | Execution Time * in $\mu$sec. |
|---|---|---|---|---|---|
| f | j | | | | |
| 34 | 0-17 | DI | Divide Integer | $(A,A+1) \div (U) \rightarrow A$; Remainder $\rightarrow A+1$ | 10.125 |
| 35 | 0-17 | DSF | Divide Single Fractional | $(A) \div (U) \rightarrow A+1$ | 10.125 |
| 36 | 0-17 | DF | Divide Fractional | $(A,A+1) \div (U) \rightarrow A$; Remainder $\rightarrow A+1$ | 10.125 |
| 37 | – | – | Illegal Code | Interrupt to $241_8$ | – |
| 40 | 0-17 | OR | Logical OR | (A) OR (U) $\rightarrow$ A+1 | .75 |
| 41 | 0-17 | XOR | Logical Exclusive OR | (A) XOR (U) $\rightarrow$ A+1 | .75 |
| 42 | 0-17 | AND | Logical AND | (A) AND (U) $\rightarrow$ A+1 | .75 |
| 43 | 0-17 | MLU | Masked Load Upper | (U) AND (M) OR (A) AND (M) $\rightarrow$ A+1 | .75 |
| 44 | 0-17 | TEP | Test Even Parity | Skip if (A) AND (U) is even parity | 2.00/1.25 |
| 45 | 0-17 | TOP | Test Odd Parity | Skip if (A) AND (U) is odd parity | 2.00/1.25 |
| 46 | 0-17 | LXI | Load X Increment | $(U) \rightarrow {X_a}_{35-18}$ | 1.00 |
| 47 | 0-17 | TLEM<br>TNGM | Test Less or Equal to Modifier<br>Test Not Greater than Modifier | Skip if $(X_a)_{17-0} \geq (U)$; always $(X_a)_{17-00} + (X_a)_{35-18} \rightarrow (X_a)_{17-00}$ | 1.75 /1.00 |
| 50 | 0-17 | TZ | Test for Zero | Skip if $(U) = \pm 0$ | 1.625/ .875 |
| 51 | 0-17 | TNZ | Test for Non Zero | Skip if $(U) \neq \pm 0$ | 1.625/ .875 |
| 52 | 0-17 | TE | Test for Equal | Skip if $(A) = (U)$ | 1.625/ .875 |
| 53 | 0-17 | TNE | Test for Not Equal | Skip if $(A) \neq (U)$ | 1.625/ .875 |
| 54 | 0-17 | TLE<br>TNG | Test for Less or Equal<br>Test for Not Greater | Skip if $(U) \leq (A)$ | 1.625/ .875 |
| 55 | 0-17 | TG | Test for Greater | Skip if $(U) > (A)$ | 1.625/ .875 |
| 56 | 0-17 | TW | Test for Within Range | Skip if $(A) < (U) \leq (A+1)$ | 1.75 /1.00 |
| 57 | 0-17 | TNW | Test for Not Within Range | Skip if $(U) \leq (A)$ or $(U) > (A+1)$ | 1.75 /1.00 |
| 60 | 0-17 | TP | Test for Positive | Skip if $(U)_{35} = 0$ | 1.50 / .75 |
| 61 | 0-17 | TN | Test for Negative | Skip if $(U)_{35} = 1$ | 1.50 / .75 |
| 62 | 0-17 | SE | Search for Equal | Skip if $(U) = (A)$, repeat | 2.25 + .75K |
| 63 | 0-17 | SNE | Search for Not Equal | Skip if $(U) \neq (A)$, repeat | 2.25 + .75K |
| 64 | 0-17 | SLE<br>SNG | Search for Less or Equal<br>Search for Not Greater | Skip if $(U) \leq (A)$, repeat | 2.25 + .75K |
| 65 | 0-17 | SG | Search for Greater | Skip if $(U) > (A)$, repeat | 2.25 + .75K |

*See notes at end of table.

| Instruction Code | | Mnemonic | Instruction | Description | Execution Time * in $\mu$sec. |
|---|---|---|---|---|---|
| f | j | | | | |
| 66 | 0-17 | SW | Search for Within Range | Skip if $(A) < (U) \leq (A+1)$, repeat | 2.25 + 75K |
| 67 | 0-17 | SNW | Search for Not Within Range | Skip if $(U) \leq (A)$ or $(U) > (A+1)$, repeat | 2.25 + .75K |
| 70 | † | JGD | Jump on Greater and Decrement | Jump to U if $(ja) > 0$, then $(ja) - 1 \rightarrow ja$ | 1.50/ .75 always |
| 71 | 00 | MSE | Mask Search for Equal | Skip if (U) AND (M) = (A) AND (M), repeat | 2.25 + .75K |
| 71 | 01 | MSNE | Mask Search for Not Equal | Skip if (U) AND (M) $\neq$ (A) AND (M), repeat | 2.25 + .75K |
| 71 | 02 | MSLE<br>MSNG | Mask Search for Less or Equal<br>Mask Search for Not Greater | Skip if (U) AND (M) $\leq$ (A) AND (M), repeat | 2.25 + .75K |
| 71 | 03 | MSG | Mask Search for Greater | Skip if (U) AND (M) > (A) AND (M), repeat | 2.25 + .75K |
| 71 | 04 | MSW | Masked Search for Within Range | Skip if (A) AND (M) < (U) AND (M) $\leq$ (A+1) AND (M), repeat | 2.25 + .75K |
| 71 | 05 | MSNW | Masked Search for Not Within Range | Skip if (U) AND (M) $\leq$ (A) AND (M) or (U) AND (M) $\geq$ (A+1) AND (M), repeat | 2.25 + .75K |
| 71 | 06 | MASL. | Masked Alphanumeric Search for Less or Equal | Skip if (U) AND (M) $\leq$ (A) AND (M), repeat | 2.25 + .75K |
| 71 | 07 | MASG | Masked Alphanumeric Search for Greater | Skip if (U) AND (M) > (A) AND (M), repeat | 2.25 + .75K |
| 71 | 10 | DA | Double Precision Fixed Point Add | $(A,A+1) + (U,U+1) \rightarrow A,A+1$ | 1.625 |
| 71 | 11 | DAN | Double Precision Fixed Point Add Negative | $(A,A+1) - (U,U+1) \rightarrow A,A+1$ | 1.625 |
| 71 | 12 | DS | Double Store A | $(A,A+1) \rightarrow U,U+1$ | 1.50 |
| 71 | 13 | DL | Double Load A | $(U,U+1) \rightarrow A,A+1$ | 1.50 |
| 71 | 14 | DLN | Double Load Negative A | $-(U,U+1) \rightarrow A,A+1$ | 1.50 |
| 71 | 15 | DLM | Double Load Magnitude A | $\|(U,U+1)\| \rightarrow A,A+1$ | 1.50 |
| 71 | 16 | DJZ | Double Precision Zero Jump | Jump to U if $(A,A+1) = \pm 0$ | 1.625/ .875 always |
| 71 | 17 | DTE | Double Precision Test Equal | Skip if $(U,U+1) = (A,A+1)$ | 2.375/1.625 |
| 72 | 00 | – | Illegal Code | Interrupt to $241_8$ | – |
| 72 | 01 | SLJ | Store Location and Jump | (P) – Base address modifier [BI or BD] $\rightarrow U_{17-00}$; jump to U+1 | 2.125 always |

† The j and a designators together serve to specify any of the 128 control registers.

* See notes at end of table.

| Instruction Code | | Mnemonic | Instruction | Description | Execution Time * in μsec. |
|---|---|---|---|---|---|
| f | j | | | | |
| 72 | 02 | JPS | Jump on Positive and Shift | If $(A)_{35} = 0$, jump to U; always shift (A) left circularly one position | 1.50 / .75 always |
| 72 | 03 | JNS | Jump on Negative and Shift | If $(A)_{35} = 1$, jump to U; always shift (A) left circularly one position | 1.50 / .75 always |
| 72 | 04 | AH | Add Halves | $(A)_{17-00} + (U)_{17-00} \rightarrow (A)_{17-00}$; $(A)_{35-18} + (U)_{35-18} \rightarrow A_{35-18}$ | .75 |
| 72 | 05 | ANH | Add Negative Halves | $(A)_{17-00} - (U)_{17-00} \rightarrow A_{17-00}$; $(A)_{35-18} - (U)_{35-18} \rightarrow A_{35-18}$ | .75 |
| 72 | 06 | AT | Add Thirds | $(A)_{35-24} + (U)_{35-24} \rightarrow A_{35-24}$; $(A)_{23-12} + (U)_{23-12} \rightarrow A_{23-12}$; $(A)_{11-00} + (U)_{11-00} \rightarrow A_{11-00}$ | .75 |
| 72 | 07 | ANT | Add Negative Thirds | $(A)_{35-24} - (U)_{35-24} \rightarrow A_{35-24}$; $(A)_{23-12} - (U)_{23-12} \rightarrow A_{23-12}$; $(A)_{11-00} - (U)_{11-00} \rightarrow A_{11-00}$ | .75 |
| 72 | 10 | EX | Execute | Execute the instruction at U | .75 |
| 72 | 11 | ER | Executive Return | (PSR) → 000; Ones to D7 and D6 and zeros to D8 and D5–D0 of PSR; interrupt to $242_8$ | 1.375 always |
| 72 | 12 | – | Illegal Code | Interrupt to $241_8$ | – |
| 72 | 13 | PAIJ | Prevent All I/O interrupts and Jump | Disable all I/O interrupts and Jump to U | .75 always |
| 72 | 14 | SCN | Store Channel Number | If a = 0, channel number → $U_{3-0}$ If a = 1, channel number → $U_{3-0}$ and Processor Number → $U_{5-4}$ | .75 |
| 72 | 15 | LPS | Load Processor State Register | (U) → PSR | .75 |
| 72 | 16 | LSL | Load Storage Limits Register | (U) → Storage Limits Register | .75 |
| 72 | 17 | – | Illegal Code | Interrupt to $241_8$ | – |
| 73 | 00 | SSC | Single Shift Circular | Shift (A) right U places circularly | .75 always |
| 73 | 01 | DSC | Double Shift Circular | Shift (A,A + 1) right U places circularly | .75 always |
| 73 | 02 | SSL | Single Shift Logical | Shift (A) right U places; zero fill | .75 always |
| 73 | 03 | DSL | Double Shift Logical | Shift (A,A + 1) right U places; zero fill | .75 always |
| 73 | 04 | SSA | Single Shift Algebraic | Shift (A) right U places; sign fill | .75 always |

* See notes at end of table.

| Instruction Code | | Mnemonic | Instruction | Description | Execution Time * in μsec. |
|---|---|---|---|---|---|
| f | j | | | | |
| 73 | 05 | DSA | Double Shift Algebraic | Shift (A,A + 1) right U places; sign fill | .75 always |
| 73 | 06 | LSC | Load Shift and Count | (U) → A; Shift (A) left circularly until $(A)_{35} \neq (A)_{34}$ or until (A) has been shifted 35 times. Store the result in A and the number of shifts in A + 1 | 1.125 |
| 73 | 07 | DLSC | Double Load Shift and Count | (U,U + 1) – (A,A + 1); shift (A,A + 1) left circularly until $(A)_{71} = (A)_{70}$ or until (A) has been shifted 71 times. Store the result in A,A + 1 and the number of shifts in A + 2 | 2.125 |
| 73 | 10 | LSSC | Left Single Shift Circular | Shift (A) left U places circularly | .75 always |
| 73 | 11 | LDSC | Left Double Shift Circular | Shift (A,A + 1) left U places circularly | .75 always |
| 73 | 12 | LSSL | Left Single Shift Logical | Shift (A) left U places; zero fill | .75 always |
| 73 | 13 | LDSL | Left Double Shift Logical | Shift (A,A + 1) left U places; zero fill | .75 always |
| 73 | 14 | III | Initiate Inter-Processor Interrupt | Initiate Inter-Processor Interrupt on channel a | .75 always |
| 73 | 15 | SIL | Select Interrupt Location | $(A)_{2-0}$ → MSR | .75 always |
| 73 | 16 | LCR | Load Channel Select Register | If a = 0, $(U)_{3-0}$ → CSR; | .875 |
| | | LLA | Load Last Address | If a = 1, $(U)_{2-0}$ → LAR | |
| 73 | 17 | TS | Test and Set | If $(U)_{30} = 1$, interrupt to $244_8$ If $(U)_{30} = 0$, take next instruction. Always set $(U)_{30} = 1$; $(U)_{35-31} = 0$; $(U)_{29-00}$ undisturbed. | 1.125 always |
| 74 | 00 | JZ | Jump on Zero | If (A) = ±0, jump to U | 1.50 / .75 always |
| 74 | 01 | JNZ | Jump on Non Zero | If (A) ≠ 0, jump to U | 1.50 / .75 always |
| 74 | 02 | JP | Jump on Positive | If $(A)_{35} = 0$, jump to U | 1.50 / .75 always |
| 74 | 03 | JN | Jump on Negative | If $(A)_{35} = 1$, jump to U | 1.50 / .75 always |
| 74 | 04 | JK | Jump on Keys | If a = key which is set, or if a = 0 jump to U | .75 always |

* See notes at end of table.

| Instruction Code | | Mnemonic | Instruction | Description | Execution Time * in $\mu$sec. |
|---|---|---|---|---|---|
| f | j | | | | |
| 74 | 05 | HKJ<br>HJ | Halt on Keys and Jump<br>Halt and Jump | Stop if a = 0, or if a AND key setting ≠ 0 | .75 always |
| 74 | 06 | NOP | No Operation | Proceed to next instruction | .75 if i = 0 |
| 74 | 07 | AAIJ | Allow all I/O interrupts and Jump | Enable all interrupts and jump to U | .75 always |
| 74 | 10 | JNB | Jump on No Low Bit | If $(A)_{00} = 0$, jump to U | 1.50 / .75 always |
| 74 | 11 | JB | Jump on Low Bit | If $(A)_{00} = 1$, jump to U | 1.50 / .75 always |
| 74 | 12 | JMGI | Jump Modifier Greater and Increment | If $(X_a)_{17-00} > 0$, jump to U; always increment $X_a$ | 1.625/ .75 always |
| 74 | 13 | LMJ | Load Modifier and Jump | $(P) \rightarrow X_{a_{17-00}}$; jump to U | .75 always |
| 74 | 14 | JO | Jump on Overflow | Jump to U if overflow designator set | 1.50 / .75 always |
| 74 | 15 | JNO | Jump on No Overflow | Jump to U if overflow designator not set | 1.50 / .75 always |
| 74 | 16 | JC | Jump on Carry | Jump to U if carry designator set | 1.50 / .75 always |
| 74 | 17 | JNC | Jump on No Carry | Jump to U if carry designator not set | 1.50 / .75 always |
| 75 | 00 | LIC | Load Input Channel | (U) → input control register; initiate input mode on channel a OR CSR | .75 |
| 75 | 01 | LICM | Load Input Channel and Monitor | (U) → input control register and initiate input mode on channel a OR CSR with monitor | .75 |
| 75 | 02 | JIC | Jump on Input Channel Busy | If channel a OR CSR is in input mode, jump to U | .75 always |
| 75 | 03 | DIC | Disconnect Input Channel | Terminate input mode on channel a OR CSR | .75 always |
| 75 | 04 | LOC | Load Output Channel | (U) → output control registor, initiate output mode on channel a OR CSR | .75 |
| 75 | 05 | LOCM | Load Output Channel and Monitor | (U) → output control register; initiate output mode on channel a OR CSR with monitor | .75 |
| 75 | 06 | JOC | Jump on Output Channel Busy | If channel a OR CSR is in output mode, jump to U | .75 always |
| 75 | 07 | DOC | Disconnect Output Channel | Terminate output mode on channel a OR CSR | .75 always |

* See notes at end of table.

| Instruction Code | | Mnemonic | Instruction | Description | Execution Time * in $\mu$sec. |
|---|---|---|---|---|---|
| f | j | | | | |
| 75 | 10 | LFC | Load Function in Channel | (U) → output control register, and initiate function mode on channel a OR CSR | .75 |
| 75 | 11 | LFCM | Load Function in Channel and Monitor | (U) → output control register and initiate function mode on channel a OR CSR with monitor | .75 |
| 75 | 12 | JFC | Jump on Function in Channel | If channel a OR CSR is in function mode, jump to U | .75 always |
| 75 | 13 | – | Illegal Code | If in Guard mode, interrupt to address $243_8$; if not, go to next instruction | .75 |
| 75 | 14 | AACI | Allow All Channel External Interrupts | All external interrupts are allowed | .75 always |
| 75 | 15 | PACI | Prevent All Channel External Interrupts | All external interrupts are disabled | .75 always |
| 75 | 16 | – | Illegal Code } | If in Guard Mode, interrupt to address $243_8$; if not go to next instruction | – |
| 75 | 17 | – | Illegal Code } | | |
| 76 | 00 | FA | Floating Add | (A) + (U) → A,A + 1 | 1.875 |
| 76 | 01 | FAN | Floating Add Negative | (A) – (U) → A,A + 1 | 1.875 |
| 76 | 02 | FM | Floating Multiply | (A) x (U) → A,A + 1 | 2.625 |
| 76 | 03 | FD | Floating Divide | (A) ÷ (U) → A ; Remainder → A + 1 | 8.25** |
| 76 | 04 | LUF | Load and Unpack Floating | Unpack (U); store fixed-point part in A + 1 and store biased exponent in $A_{01-10}$ | .75 |
| 76 | 05 | LCF | Load and Convert to Floating | Normalize (U); pack with biased exponent from (A) and store at A + 1 | 1.125 |
| 76 | 06 | MCDU | Magnitude of Characteristic Difference to Upper | $\|\,\|(A)_{35-27}\| - \|(U)_{35-27}\|\,\| \rightarrow$ A + 1 bits 08 to 00 | .75 |
| 76 | 07 | CDU | Characteristic Difference to Upper | $\|(A)_{35-27}\| - \|(U)_{35-27}\| \rightarrow$ A + 1 bits 08 to 00 | .75 |
| 76 | 10 | DFA | Double Precision Floating Add | (A,A + 1) + (U,U + 1) → A,A + 1 | 2.625 |
| 76 | 11 | DFAN | Double Precision Floating Add Negative | (A,A + 1) – (U,U + 1) → A,A + 1 | 2.625 |
| 76 | 12 | DFM | Double Precision Floating Multiply | (A,A + 1) × (U,U + 1) → A,A + 1 | 4.250 |
| 76 | 13 | DFD | Double Precision Floating Divide | (A,A + 1) ÷ (U,U + 1) → A,A + 1 | 17.25*** |

* See notes at end of table.

| Instruction Code | | Mnemonic | Instruction | Description | Execution Time * in $\mu$sec. |
|---|---|---|---|---|---|
| f | j | | | | |
| 76 | 14 | DFU | Double Load & Unpack Floating | Unpack (U,U+1); fixed-point part → A+1, A+2; exponent → $A_{10-00}$ | 1.50 |
| 76 | 15 | DFP | Double Load & Convert to Floating | Normalize and pack from fixed-point part in (U,U+1), exponent in $(A)_{10-00}$ and store in A+1, A+2 | 2.125 |
| 76 | 16 | FEL | Floating Expand and Load | $(U)_{01,08,27} \rightarrow A, A+1_{01,11,60}$ | 1.00 |
| 76 | 17 | FCL | Floating Compress and Load | $(U,U+1)_{01,11,60} \rightarrow A_{01,08,27}$ | 1.625 |
| 77 | 0-17 | – | Illegal Code | Interrupt to $241_8$ | – |

NOTES

* Times given are for alternate-bank case only except where noted "always". In cases where instruction and operand are procured from the same bank, add .75 microseconds to execution time.

For all comparison instructions, the first number represents the skip or jump condition, the second number is for no skip or no jump condition.

For function codes 01–67, add .375 microseconds to execution times for 6-bit and 12-bit writes.

Execution time for the Block Transfer and the Search instructions depends on the number of repetitions of the instruction required. The variance in all cases is .75K micro-seconds where K equals the number of repetitions; that is, K equals the number of words in the block being transferred or the number of words searched before a match is found.

** If 28 instead of 27 subtractions are performed, add .25 microseconds.

*** If 61 instead of 60 subtractions are performed; add .25 microseconds.

# APPENDIX D. MATH-PACK ROUTINES

The following is a complete listing of MATH-PACK routines. Numbers refer to section numbers in revision 1 of "UNIVAC 1108 MATH-PACK Program Abstracts", UP-4051 and in the "UNIVAC 1108 MATH-PACK Programmers Reference Manual", UP-7542.

2. INTERPOLATION

2.1. GNINT – Gregory-Newton Interpolation

2.2. GNEXT – Gregory-Newton Extrapolation

2.3. GNPOL – Gregory-Newton Polynomial Evaluation

2.4. BESINT – Bessel Interpolation

2.5. STINT – Stirling Interpolation

2.6. CDINT – Gauss Central-Difference Interpolation

2.7. AITINT – Aitken Interpolation

2.8. YLGINT – Lagrange Interpolation

2.9. SPLN1, SPLN2 – Spline Interpolation

3. NUMERICAL INTEGRATION

3.1. TRAPNI – Trapezoidal Rule

3.2. SIM1NI – Simpson 1/3 Rule

3.3. SIM3NI – Simpson 3/8 Rule

3.4. STEPNI – Variable Step Integration

3.5. GENNI – Generalized Numerical Quadrature

3.6. DOUBNI – Double Integration

3.7. LGAUSS – Gauss Quadrature Abscissas and Weights

3.8. SIMPTS – Simpson 1/3 Rule Abscissas and Weights

12. SYSTEMS OF EQUATIONS

12.1. JACMX – Jacobi Iteration to Determine Eigenvalues and Eigenvectors of Symmetric Matrix

12.2. HJACMX – Jacobi Iteration to Determine Eigenvalues and Eigenvectors of Hermitian Matrix

12.3. LSIMEQ – Solution to a Set of Linear Simultaneous Equations

12.4. NSIMEQ – Functional Iteration to Determine Solution to Set of Non-Linear Equations

13. CURVE FITTING

13.1. CFSRIE – Coefficients of Fourier Series on a Continuous Range

13.2. FTRANS – Fourier Transform

13.3. DFSRIE – Coefficients of Fourier Series on Discrete Range

13.4. FITD – Fitted Value and Derivative Values for a Least-Squares Polynomial

13.5. ORTHLS – Orthogonal Polynomial Least-Squares Curve-Fitting

13.6. FITY – Fitted Values for a Least-Squares Polynomial

13.7. COEFS – Coefficients of a Least-Squares Polynomial

14. PSUEDO-RANDOM NUMBER GENERATORS

14.1. NRAND – Interval $(0,2^{27})$ Generator

14.2. RANDU – Uniform Distribution

14.3. RANDN – Normal Distribution

14.4. RANDEX – Exponential Distribution

15. SPECIFIC FUNCTIONS

15.1. BSSL – Zero- and First-Order Bessel Functions

15.2. BESJ – Regular Bessel Functions of Real Argument

15.3. BESY – Irregular Bessel Functions of Real Argument

15.4. BESI – Regular Bessel Functions of Imaginary Argument

15.5. BESK – Irregular Bessel Functions of Imaginary Argument

15.6. GAMMA – Gamma Function Evaluation

15.7. LEGEN – Legendre Polynomial Evaluation

15.8. ARCTNQ – Arctangent of a Quotient

# APPENDIX E. STAT-PACK ROUTINES

The following is a complete listing of STAT-PACK routines. Numbers refer to section numbers in revision 1 of "UNIVAC 1108 STAT-PACK Program Abstracts", UP-4041 and in the "UNIVAC 1108 STAT-PACK Programmers Reference Manual", UP-7502.

2. DESCRIPTIVE STATISTICS

2.1. FREQP – Frequency Polygon

2.2. HIST – Histogram

2.3. MHIST – Multivariate Histogram

2.4. GROUP – Grouping of Data

3. ELEMENTARY POPULATION STATISTICS

3.1. AMEAN – Arithmetic Mean

3.2. GMEAN – Geometric Mean

3.3. HMEAN – Harmonic Mean

3.4. MEDIAN – Median

3.5. MODE – Mode

3.6. QUANT – Quantiles

3.7. OGIVE – Distribution Curve

3.8. IQRNG – Interpercentile Range

3.9. RANGE – Range

3.10. MNDEV – Mean Deviation

3.11. STDEV – Standard Deviation

3.12. CVAR – Coefficient of Variation

3.13. ORDER – Order and Rank Statistics

3.14. CMONT – Central Moments

3.15. AMONT – Absolute Moments

3.16. CUMLT – Cumulants

3.17. SHPCOR – Sheppard's Corrections

3.18. KURSK – Skewness and Kurtosis