SPERRY RAND

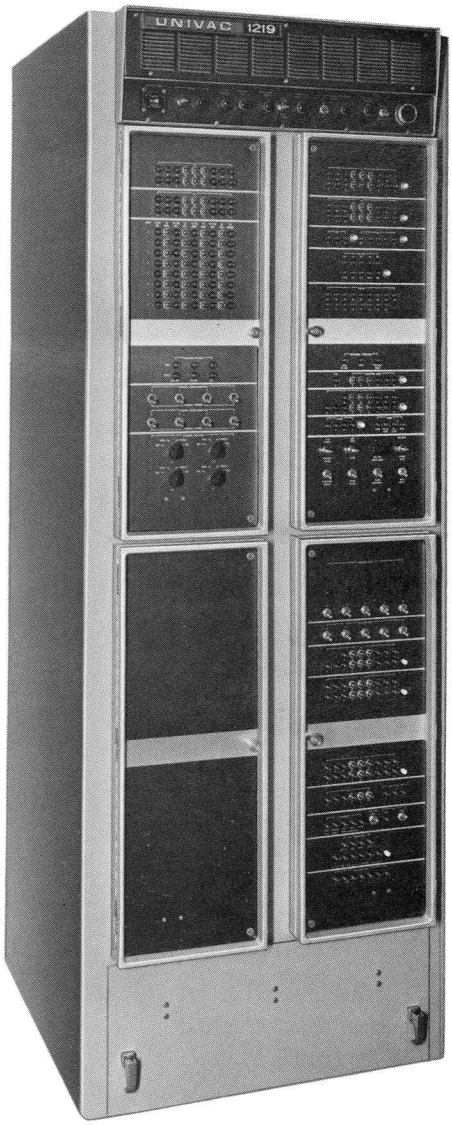
UNIVAC®

1219B

COMPUTER

TECHNICAL DESCRIPTION

SPERRY RAND



UNIVAC®

1219B

COMPUTER

TECHNICAL DESCRIPTION

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont.)

TABLE OF CONTENTS (Cont.)

LIST OF ILLUSTRATIONS

DESCRIPTION OF COMPUTER

INTRODUCTION

The UNIVAC 1219B Military Computer is a medium-scale, general-purpose, digital computer, specifically designed to comply with the environmental specifications of MIL-E-16400. It is a more versatile version of the 1219 computer and is functionally compatible with it and the widely used UNIVAC 1218 Computer.

To meet the extreme requirements of real time and concurrent batch processing operations, the UNIVAC 1219B is equipped with a 2-microsecond internal random-access core memory in sizes of 8192, 16,384, 32,768 or 65,536 18-bit words with a "read" access time of 0.9 microsecond and a fast 500-nanosecond control memory. Other random-access storage devices connected to input/output channels provide expandable memory capacities. A portion (32 word locations) of core memory has a characteristic nondestructive feature which stores constants and instructions for automatic recovery from fault situations and for an initial load of routines.

With its high internal operating speed, core memory, and 500-nanosecond control memory, the UNIVAC 1219B Computer is capable of transferring 500,000 words per second. Arithmetic and input/output operations can be performed on the basis of a single-length 18-bit word, or a double-length, 36-bit word if greater precision is required for compatibility with other computers. The repertoire of 102 instructions allows complete programming freedom in mathematical and logical computations, as well as full control of input/output buffer transfers and of real-time, on-line operations. The computer features parallel transfers, one's complement binary arithmetic, direct addressing, and program controlled automatic address or operand modification via eight control-memory-contained index registers.

GENERAL CHARACTERISTICS

The accompanying specifications were used as the basis for the design and construction of the 1219B computer.

General Electronic Equipment (Reliability, Simplicity, Material, Workmanship, Production and Central Inspection, Ease of Operation and Maintenance)	MIL-E-16400E (4)
Enclosure	MIL-STD-108D
Technical Manuals	MIL-M-16616(1)
Drawings	MIL-D-70327(2)
Preparation for Delivery	MIL-E-17555F(2)
Radio Interference	MIL-I-16910C(2)
Vibration	MIL-STD-167
Quality Control	MIL-Q-9858B, Amendment 2
Shock Tests	MIL-S-901C

Physical

The computer is housed in a single cabinet that contains the power supply, logic circuits, core memory, maintenance, and control panel, and a cooling system. Logic modules are encapsulated printed circuit cards which plug into the wired chassis of easily accessible pull-out drawers. The front of each drawer is the associated portion of the computer control panel. The logic and memory drawers are mounted in a vertical position. Figure 1 shows the logic drawer extended; Figure 2 shows a main memory module. The power supply drawer (see Figure 3) is mounted horizontally at the bottom of the cabinet.

Size and Weight

2 Module, 8 I/O, 32K Memory

Height: 71.75 inches Depth: 30.5 inches
Width: 26.25 Inches Weight: 1000+ lbs.

3 Module, 12 or 16 I/O, 65K Memory

Height: 71.75 In. Depth: 30.03 inches
Width: 38.00 in. Weight: 1400-1500 lbs.

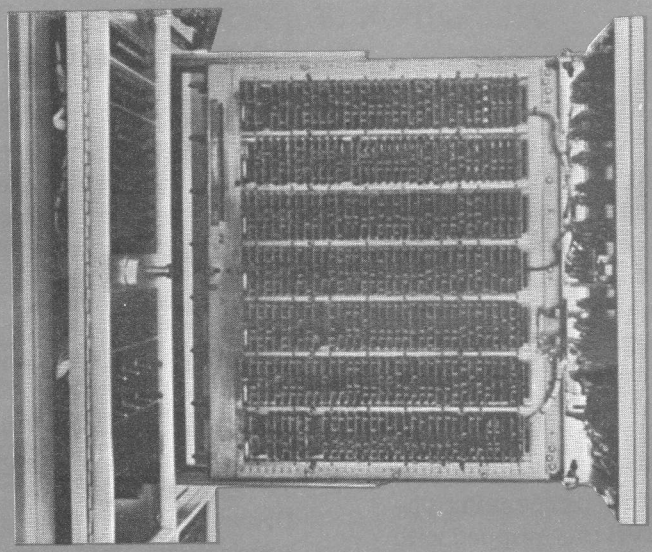


Figure 1. Logic Drawer Extended

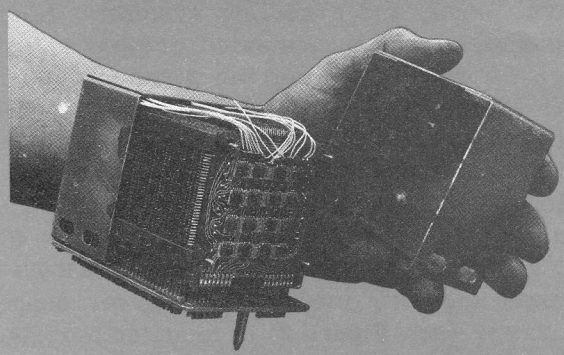


Figure 2. Main Memory Module

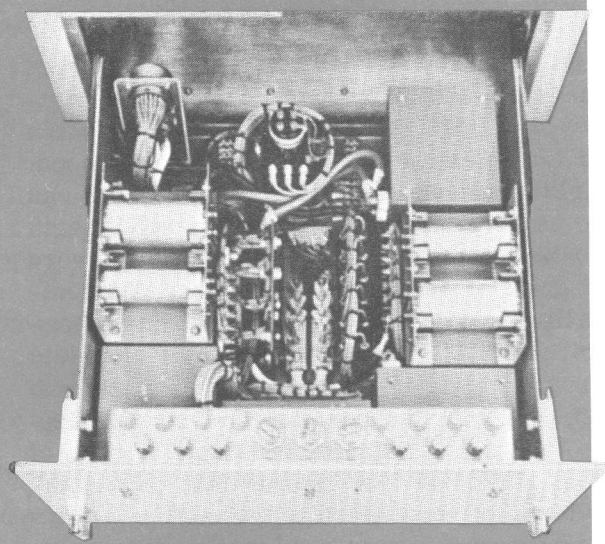


Figure 3. Power Supply Drawer

Environment

Operating temperatures 0°C to 50°C
Storage temperatures -62°C to +75°C
Humidity — Relative humidity to 95 percent
Cooling — Blower forced ambient air.

Power Requirements

115-volt, ± 5 percent, 3-phase, 400-cps, 2000 watts maximum, air cooled (for 16 I/O channels and 32K memory).

TECHNICAL CHARACTERISTICS

Memory

Control Memory

Cycle Time:	500 nanoseconds
Capacity:	128 or 256 18-bit words
Type:	Word organized, magnetic core
Purpose:	Index registers, clock cells, I/O buffer control registers; operates in the "shadow of the Main Memory at a 4:1 ratio.

Main Memory

Cycle Time:	2 microseconds
Capacity:	8192, 16,384, 32,768 or 65,536 18-bit words (standard options)
Type:	Coincident current, magnetic core
Purpose:	I/O interrupt registers, program and data strg.

Nondestructive Readout Memory

Cycle Time:	2 microseconds
Capacity:	32 18-bit words
Type:	Word organized, transformer core, unalterable
Purpose:	Bootstrap (initial load) program storage. Programs available for paper tape and magnetic tape load; others on request.

Input/Output

Channels

Type:	Simplex, 18-bit parallel
Number:	32 maximum; 16 input plus 16 output
Transfer Rate:	One channel — 166,000 18-bit words per second Multi-channel —500,000 18-bit words per second (maximum)
Operation:	Each channel fully buffered and once activated operates without program attention, asynchronously, at the rate of the peripheral unit.

Information Transfers

Input Channels:	Input data, interrupt data
Output Chan.	Output data, external command data
Processing Time Required:	2 microseconds/word transferred 0 microsecond during extended sequence instructions
Delay due to Program:	2 microseconds (max.)

Operating Modes (Standard)

Normal Single Channel:
18-bit parallel transfers

Normal Dual Channel:
Consecutive (even/odd numbered) channels may be "paired" to form a single 36-bit parallel channel.

Externally Specified Index (Dual Chan.):
18-bit parallel data transfers with storage address indirectly specified by external device; useful for multiplexing/decommutating data to/from computer

Externally Spec. Address (Dual Channel):
18-bit parallel data transfer address directly specified by external device.

Continuous Data Mode:
Program controlled automatic reinitiation of previously established buffers. Program controlled termination of CDM. 18-bit parallel or 36-bit parallel input/output word transfers.

Intercomputer Single Channel:
Direct 18-bit parallel data transfers with other UNIVAC computers. No interface adapters required for intercomputer communication.

Intercomputer Dual Channel:
Direct 36-bit parallel data transfer with other UNIVAC computers. No interface adapters required for intercomputer communication.

Interrupts

Input Channels:
16 external interrupts plus 16 internal interrupts (programmer option)

Output Channels:
16 internal interrupts (programmer option)

Control

Instructions

	Single Address
Address Modification:	8 Control-Memory contained index registers
Repertoire:	102 instructions

Clock

Type:	Automatic, additive, under program control
Location:	Control Memory
Duration:	Established under program control
Granularity:	Least significant bit represents 1/1024 second; others on request
Interrupt:	Interrupt occurs when program preset value is reached.

Synchronizer

Interrupt:	Interrupt occurs whenever the non-I/O synchronizing control line is set to logical one by an external device
Purpose:	To allow a variable-granularity clock function or to provide a high priority alarm recognition capability.

Arithmetic

Organization:	18-bit parallel, one's complement, integer
Execution Time:	Typical execution times, including instruction and data fetch plus indexing.
Add, Subtract (single length)	4 μsec
Multiply/Divide	14 μsec
Add, Subtract (double length)	6 μsec
Compare/Masked Compare and Branch	6 μsec
Register Shifts: right, left, single, double	2+.5n μsec (n = shift count)

INSTALLATION REQUIREMENTS

A motor-alternator rated at 3.0 kva furnishes regulated, transient-free, 115-volt, 400-cycle, 3-phase power to the computer. The motor-alternator operates on 220/440 volt, +10%, 60-cycle, 3-phase (3-wire) current. The computer has been designed to be cooled by either water or air. The standard configuration is an air cooled cabinet using approximately 400 cubic feet of air per minute at 32°F to 122°F. The air-cooled machine is available for use in installations such as trailers, helihuts, and land-based shelters.

Signal interconnection of the units comprising a 1219B system is accomplished through interconnecting cables terminated in connectors which are plugged into mating receptacles at each unit.

The type of installation (mobile or stationary) determines the method of installing the computer. When used in a stationary installation, the computer must be bolted to the floor to prevent the cabinet from tipping forward when the pull-out drawers are extended. When used in a mobile installation, the computer must be bolted to the floor and to the wall at the upper rear of the cabinet. The stabilizing brackets for the wall attachment are provided at the rear of the cabinet. Figures 4 and 5 are outline and dimensional drawings showing mounting holes, brackets, and space requirements.

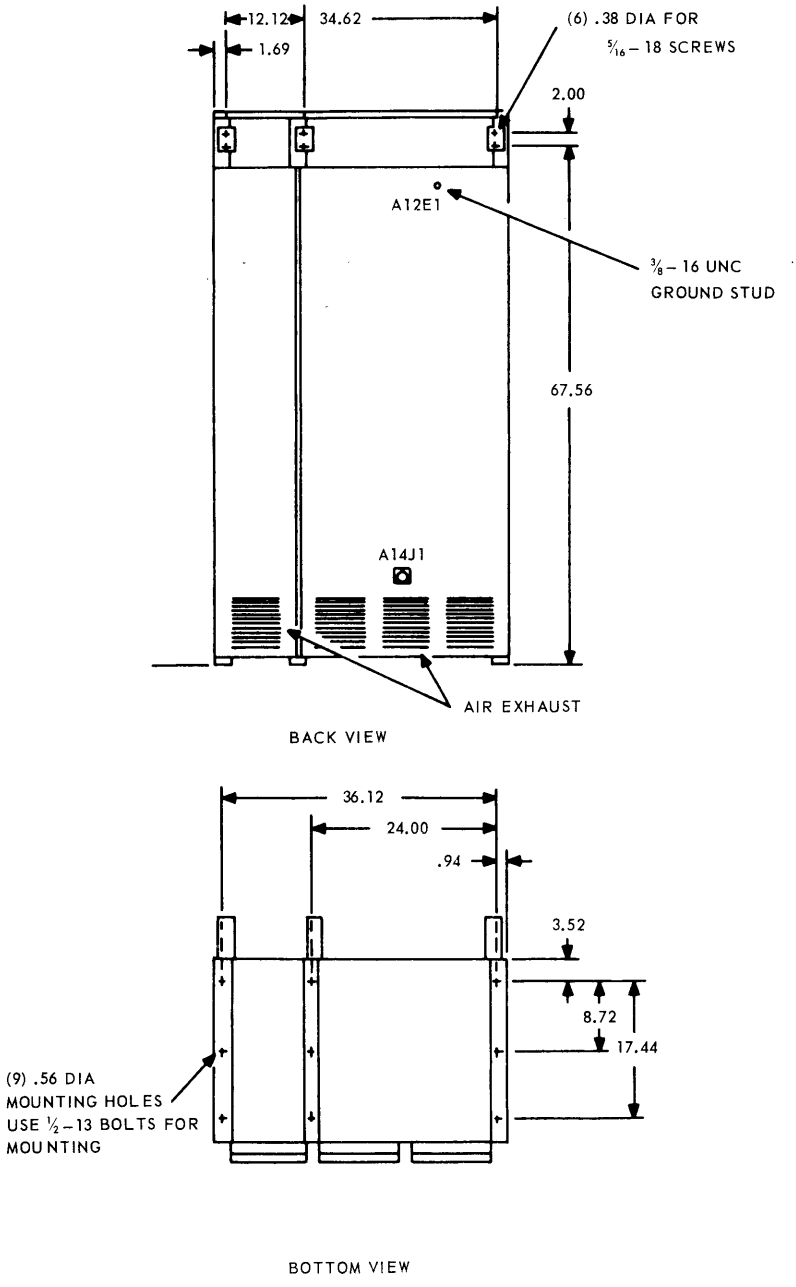


Figure 4. **Location of Cabinet Mounting Holes**

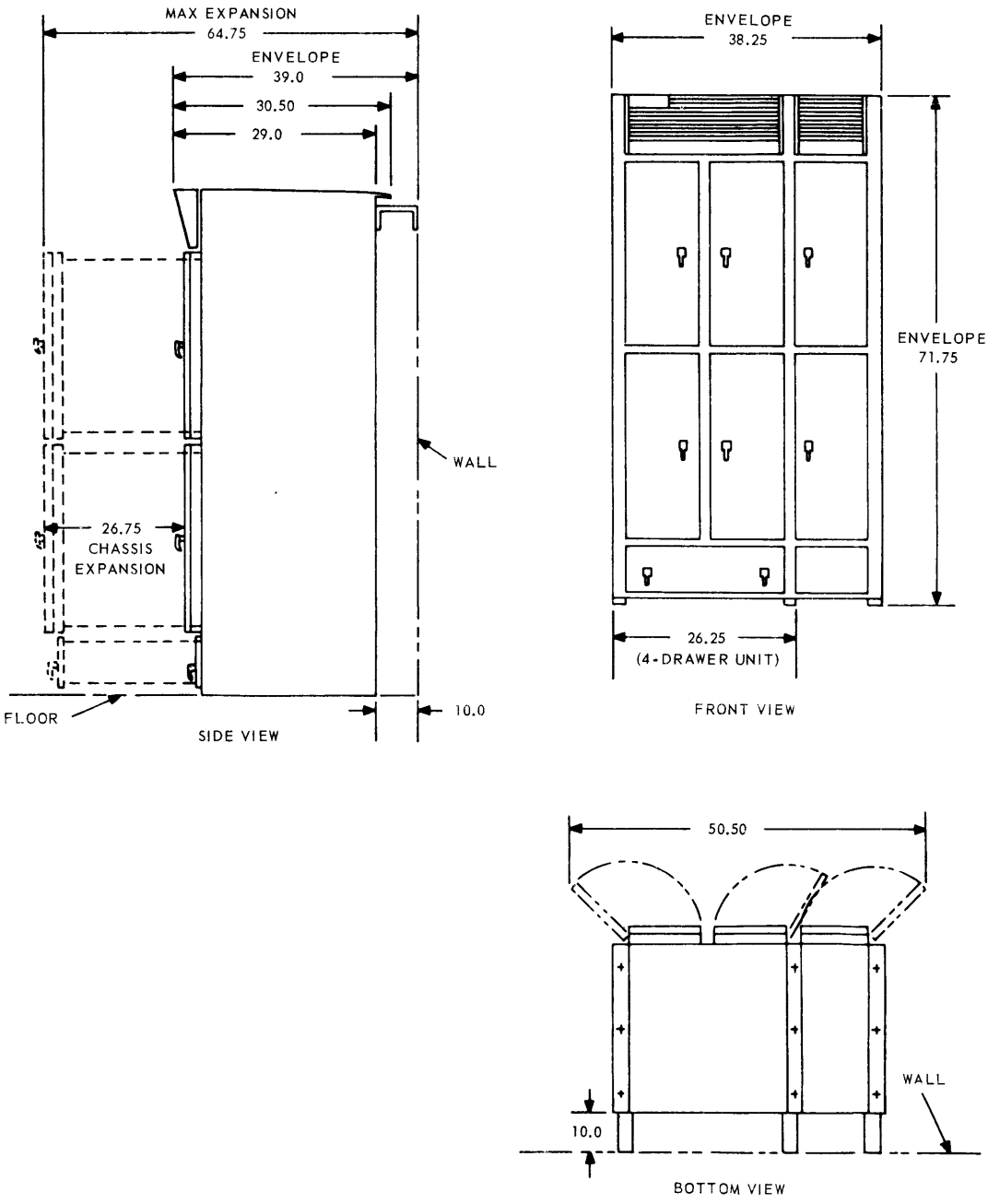


Figure 5. **Computer Clearance Requirements**

MAINTAINABILITY

Maintainability considerations are an integral part of the 1219B equipment design. Reliability of a 1219B is a function of the components selected and manufacturing techniques applied. Processes and controls applied during fabrication reflect quality, which, in turn, supports reliability. Therefore, quality, reliability, and maintainability all contribute to the productive service time of the 1219B by decreasing the time to repair when a malfunction occurs and by increasing the productive time between malfunctions.

Univac applies three important maintainability philosophies in the design of the 1219B:

- Ease of diagnosis
- Ease of parts replacement
- Commonality of replaceable parts.

Maintenance Diagnosis

The 1219B is designed to permit front access to all printed circuit card assemblies for direct testing and physical replacement. Each of the logic roll-out drawers contains two wire-wrapped chassis of printed circuit cards and other related components and connectors. When drawn out from the cabinet, an "open book" construction feature permits access to all test points and back chassis wiring for easy maintenance. The computer system is logically and electrically designed to facilitate location of malfunctioning cards or modules through diagnostic maintenance programs in conjunction with the operating console. Use of standard test equipment to support self-testing procedures is sometimes required. However, with diagnostic routines and related documentation, repair by module replacement and subsequent checkout can be accomplished normally within 15 minutes.

Ease of Parts Replacement

When a malfunction is isolated to a particular circuit card or module, the card is replaced. This card replacement concept of troubleshooting is the lowest level of repair to be performed and simplifies repair procedures.

Commonality of Replacement Parts

The computer system is designed to contain a minimum number of printed circuit cards and component types in order to keep spare parts complement and stock as low as possible. Spare parts lists are compiled with commonality of parts and established failure rates as governing factors.

COMPUTER CONTROL PANELS

The maintenance and control panels, located on the front of the computer, include indicator lamps which display a detailed report of the internal status of the computer and controls to permit manual initiation of various operations. It is not necessary during normal operations, however, to monitor the maintenance panel or console.

Each register is represented on the maintenance panel by pushbutton display lamps, each of which can be used to enter a "one" manually into the corresponding bit position, and a clear button which can be used to enter "zeros" manually into all-bit positions of the register. Many of the registers are involved only in the mechanics of executing instructions and are not directly accessible to the program.

One input/output panel is provided for each eight channels. Status and control indicators and switches for I/O activity are assembled on this portion of the computer control panel.

During the debugging or operational run of a program, different modes of computer utilization may be selected at the control panel. Operating instructions for representative modes of computer operations follow.

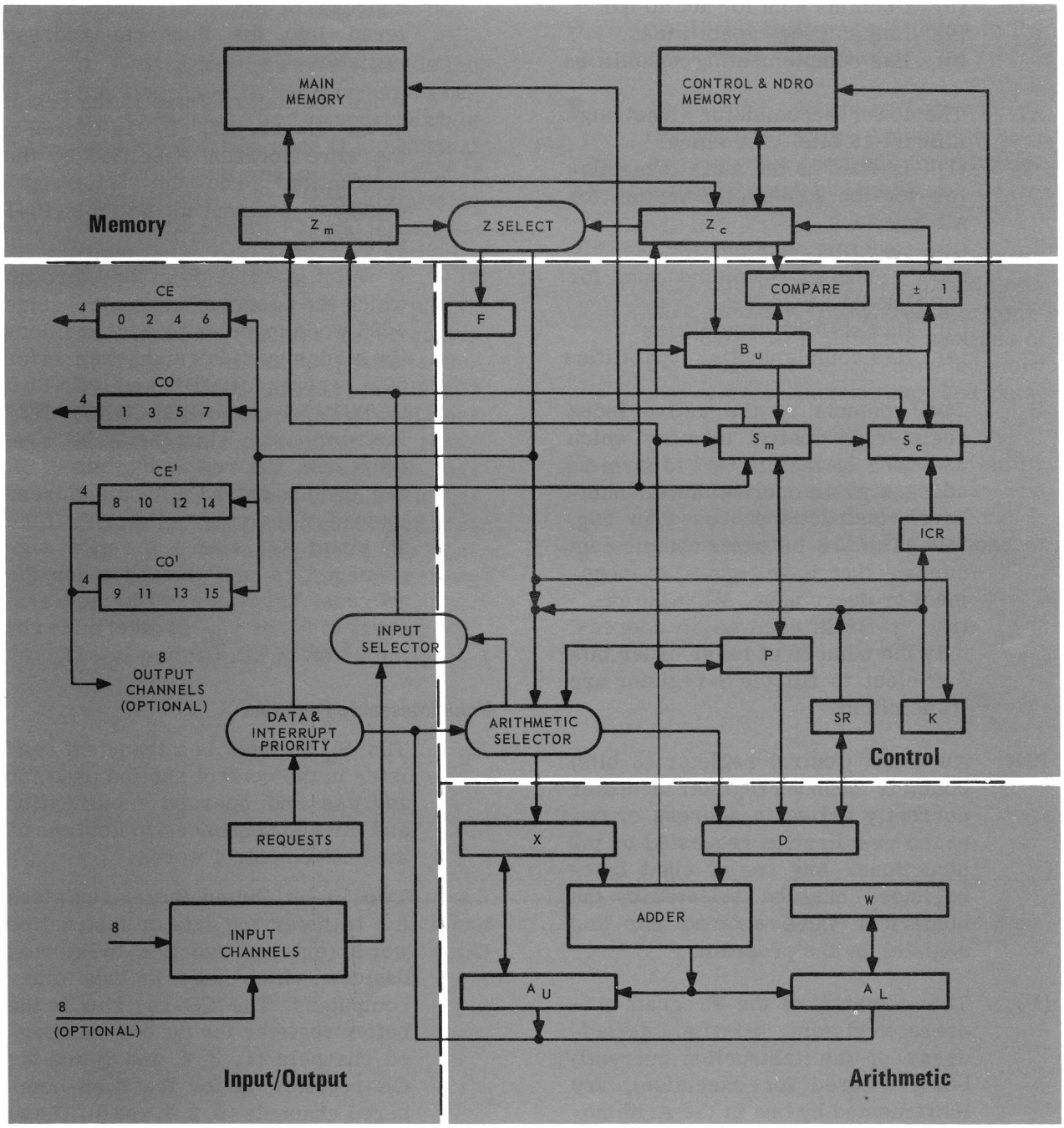


Figure 6. **UNIVAC 1219B Computer Block Diagram**

AU — The upper accumulator (most significant 18 bits) of A which:
(1) Contains a mask for logical instructions
(2) Captures the remainder for the "divide" process
(3) Contains sum for add to AU
(4) Has shifting capabilities
(5) Has complementing capabilities

AL — The lower accumulator (least significant 18 bits) of A which:
(1) Is used as the main accumulator for the Arithmetic Section for all functions
(2) Contains quotient for the "divide" process, contains sum for "add" to AL
(3) Has shifting capabilities
(4) Has complementing capabilities

B — The contents of the active index register in control memory which are used to modify "y" to form an address or an operand in odd-numbered instructions less than 50_8. "B" is an 18-bit one's complement number that may be used to increment or decrement. When the quantity "y+B" is used as an address, only the number of lower order bits sufficient to fill the S-register are transmitted.

ICR — An Index Control register (3 bits) contains the Index register identifier currently active in address or operand modification requested by instructions. Any one of eight index registers may be selected by the numerical value entered into this register by the program.

(P) — The contents of the Program Address register, "P" (i.e., the address of the instruction currently being entered for execution), are incremented by one in the Arithmetic Section as soon as the instruction is transferred from memory. If the computer is stopped, the P-register exhibits the address of the next instruction, "(P)+1". This is incremented by one again if the condition stated by a SKIP instruction is satisfied. When the current instruction is a Return Jump, (P)+1 is stored in the core location specified by the instruction, and the entrance address of the new routine is entered into the Program Address register.

When the Return Jump is the result of an "Interrupt", (P) is stored in the core location specified by the instruction since the "Interrupt" condition does not initiate the (P)+1 sequence.

SR — A 5-bit Special Register through which the program has control of the 4096-word modules in core memory (in all instructions numbered under 50, except "JUMP" and "ENTER CONSTANT or ADD CONSTANT" instructions). When the 2^3 bit contains one, the remaining bits of SR are used to extend u for an address instead of the upper bits of P. If the 2^3 bit of SR is zero, the most significant bits of P extend u for the address. Therefore, y (the address) equal to u_P or u_{SR} is determined by the 2^3 bit of SR. (Active if = 1).

Nonaddressable Registers

B_u — The B_u register is a transient register in the control section used for address and operand modification and for I/O sequences to hold one of the buffer control words.

CE and CO — Two 18-bit Output Buffer registers for transferring data or instruction words (external function) to external devices which may include other computers. The CO-register is the buffer register for the odd-numbered channels (1, 3, 5, and 7) and the CE-register is for the even-numbered channels (0, 2, 4, and 6). These two output registers may be linked in consecutive "even-odd" pairs to permit 36-bit parallel output transfers when words larger than 18 bits are desired.

CE' and CO' — Two 18-bit Output Buffer registers for channels 10_8 through 17_8 (optional).

D — An 18-bit Arithmetic Exchange register holds an operand for the adder during arithmetic operation.

F — A 7-bit Function register holds the function code of the instruction being executed. The low order six bits hold the function code (f for Format I instructions and m for Format II instructions). The most significant bit will be set for Format II instructions only. Computer control is directed from this register.

S — An Address register receives the address of a memory location at the beginning of a memory cycle and holds it to control the translators and circuitry throughout the read/write cycle. The S-register may receive its address from the Input/Output Section (which generates certain assigned addresses), the Control Section, the Arithmetic Section, or from an input channel connected to a device capable of specifying an address. Sm (16 bits) is associated with Main Memory and Sc (8 bits) is associated with Control Memory and NDRO Memory.

W — An 18-bit Shifting register is in the Arithmetic Section.

X — An 18-bit Exchange or Communication register in the Arithmetic Section receives operands for arithmetic and logical instructions.

Zm — An 18-bit Main Memory buffer register for all transfers to and from core memory. The Z-register communicates with all other sections of the computer since core memory may contain instructions and data.

Zc — An 18-bit Control Memory buffer register for all transfers to and from Control Memory locations. It communicates with all sections of the computer.

FUNCTIONAL INFORMATION

REAL-TIME PROCESSING

The ability of the UNIVAC 1219B computer to process various applications concurrently is implemented by a program intervention system called "Interrupts". These Interrupts may originate at some remote external device (External Interrupts) or they may originate within the computer (Internal Interrupts) as a result of program requests. Since more than one may occur at the same time, the computer possesses a priority scheme with decision-making qualities so that it can select the branch of operation for solving the problem requiring the most urgent attention. Under program control, the other interrupts may be honored in turn according to the next highest priority or they may be ignored. With this "interrupt" feature, real-time problem solution and maximum processing potential of the system are realized since less important routines can occupy the computer's surplus time.

MAIN MEMORY AND CONTROL MEMORY CONCURRENT OPERATION

The master clock in the UNIVAC 1219B computer controls and synchronizes all operations performed by the various sections through the electronic timing chains allotted to them. The read/restore cycle time of main memory is two microseconds. All control and timing sequences for the various functions the computer performs are based on this two-microsecond cycle. Four 500-nanosecond control memory read-write cycles occur during one main memory read-write cycle. An instruction from main memory storage can be transferred to the control section for execution in approximately 0.9 microsecond. Any modification to this instruction and complete translation are completed before the end of that main memory cycle since the modifiers are extracted from control memory in less than 250 nanoseconds. The Input/Output Section has independent access to control memory for its control words, clocks, etc., during instruction sequences.

INPUT/OUTPUT

The Input/Output Section includes those data paths and control circuits used by the computer for communicating with external equipment.

All communication between the computer and the external equipment is accomplished via 16 input and 16 output channels and their associated control circuits. The channels, both input and output, are numbered from 0 through 17_8, with each channel consisting of 18 information lines plus control lines; channel priority ranks from 17 through 0, with the high numbered channels given preference over the lower numbered channels. Input and output communication alternates if both types of requests exist simultaneously.

The Input/Output Section uses assigned control memory addresses which dictate the memory area affected by this input or output operation (buffer control words) and unique core memory addresses which contain instructions executed when certain input/output conditions occur (interrupts).

Input Channels

The input channels are used to receive two types of information from external equipment: input data and external interrupt information. External interrupt information

originates at the external equipment and usually informs the computer of an abnormal condition such as tape breakage or incorrect parity. Input data information transfers are controlled by buffer control words in control memory.

Output Channels

The output channels are used to transmit two types of information from the computer to external equipment: data and external function information. External function, transmitted through the data lines, is used for external equipment control such as turn on reader, rewind tape, or turn off typewriter. Both data and external function information transfers are controlled by buffer control words in control memory.

Buffer Control Word Locations

An Input Buffer, an Output Buffer, and an External Function Buffer may be active on a channel simultaneously. Each channel is assigned two addresses in control memory for the buffer control words for each type of buffer (see control memory address assignment). External Function Buffer control words and Continuous Data Mode (CDM) reload words occupy the same addresses for a channel.

Continuous Data Mode is a feature which provides program controlled reinitiation of previously established buffers that have terminated. If the programmer is using CDM, there is no need to send new External Function (commands). Therefore, no conflict exists in using common control memory locations.

1218-1219B Buffer Modes — Any one of three buffer modes can be manually selected for all channels by the location of a plug-in printed wiring assembly. The three modes are as follows:

a. 1218 (normal) mode
   1. Disables Continuous Data Mode.
   2. Terminates Output Data and External Function buffers only upon receiving the next output data request or external function request from the peripheral device after the last word of the buffer has been sent.

b. 1218 (NTDS compatible) mode
   1. Disables Continuous Data Mode..
   2. Terminates Output Data and External Function buffers upon sending the last word of the buffer.

c. 1219B mode
   1. Enables Continuous Data Mode.
   2. Terminates Output Data and External Function buffers upon sending the last word of the buffer.

Priority

The higher numbered channel operation is given highest priority. Then the I/O function priority circuits provide automatic selection of the higher priority operation when two or more operations are requested by peripheral equipment or by the computer at the same time. Some real-time events as well as certain information transfers require special handling or main program intervention. These operations or interrupts are processed by the Input/Output Section according to a prearranged priority scheme.

Interrupts and Assigned Interrupt Addresses

Interrupts in the computer system cause main program intervention. An instruction located in a core memory address, designated by the condition causing the interrupt, will be executed.

An external interrupt results from an external device placing a signal on an External Interrupt line. Appropriate action is generally taken by the interrupt program. Internal I/O interrupts are generated by the Input/Output Section of the computer whenever a buffer, which has been initiated with a monitor imposed, terminates.

The instruction located in the designated memory location is chosen by the programmer and is usually an indirect return jump. A Return Jump instruction stores the address of the next sequential instruction of the main program in the interrupt routine so that computer control can return. External interrupts may accompany an Interrupt Code (on the input lines) which is stored at the address following the Interrupt Entrance register at the time the computer honors the interrupt. When interrupts are honored, the computer will generate the addresses required to call out the instructions from the assigned locations as well as the addresses for storage of the Interrupt or Fault codes. These generated addresses for an 8 I/O channel computer (channels 0-7) are as follows:

(1) A Synchronizing Interrupt (not associated with any input or output channel) is provided on the computer via a single line. Whenever certain events occur at some external device, which requests the computer to perform a given routine, this synchronizing input will be used to alert the computer. When this Interrupt occurs, control is transferred to the instruction located in memory address 00016.

(2) External Interrupt Entrance Register is 100_8* plus twice the channel number for instruction location; 101_8* plus twice the channel number for code word storage.

(3) Output or External Function Monitor Interrupt Register is 140* plus twice the channel number (even addresses only).

(4) Input Monitor Interrupt Register is 160* plus twice the channel number (even addresses only).

For External Interrupt operation on a single channel (18-bit word transfers), control is transferred to the even-numbered address for that channel and the Interrupt code is stored in the odd-numbered address. When dual channel operation (36-bit word transfers) is used, control is transferred to the even-numbered address of the odd-numbered channel of the pair; the Interrupt code is stored in both of the odd-numbered addresses (18 MSB ➡ even channel).

A diagram of External Interrupt Entrance registers and their uses is shown below.

Conventionally, all interrupt entrance locations are filled with one of two types of instructions:

- To ignore the interrupt, a Remove Interrupt Lockout instruction is used, and the program will continue with the normal execution of instructions since the P-register is not affected by the interrupt itself.

FOR CHANNEL NUMBER	MEMORY ADDRESS	CONTENTS	SOURCE OF CONTENTS
0	00100	Return Jump Instr.	Programmed
	00101	Interrupt Code	Peripheral Device
1	00102	Return Jump Instr.	Programmed
	00103	Interrupt Code	Peripheral Device

*For computers with 16 I/O channels (channels 10_8-17_8) add 200_8 to these figures.

- A response to the interrupt requires a Return Jump (usually an Indirect Return Jump) to the interrupt routine. The Return Jump instruction, under Interrupt Mode, saves the address of the next instruction that would have been executed in the normal sequence if no interrupt had occurred, rather than the address of the Return Jump instruction+1. This provides a return to the program that was interrupted.

RTC Overflow Interrupt

When the RTC register overflows (777777 to 000000), the computer program is interrupted and the next instruction is taken from the RTC Overflow Entrance Register (location 13 in control memory). This interrupt may be locked out with the RTC disconnect switch on the console.

RTC Monitor Interrupt

A Real-Time Clock incrementing register is assigned location 15 in control memory. It is used for timing three specific internal interrupting capabilities that are provided by hardware design and for any other program controlled timing activities.

The Real-Time Clock Monitor Interrupt may be initiated by storing a desired time count in the Real-Time Clock Monitor Word register (location 14 in control memory) and enabling the Real-Time Clock Monitor via the "Enable Real-Time Clock Monitor" instruction (Code 50 14). When the Real-Time Clock incrementing register equals the count stored in location 14, the computer program is interrupted, the next instruction is taken from the RTC Monitor Interrupt Entrance register (location 12 in control memory), and the RTC Monitor is disabled.

Intercomputer Time Out Interrupt

The Intercomputer Time Out Interrupt is available during intercomputer operation. Any single bit of the RTC incrementing register may be wired to monitor the Resume circuitry. When the RTC count reaches the specified bit, a designator is set. If no Resume is received by the computer before the next time the count reaches that bit, the Intercomputer Time Out Interrupt is activated, and the next program instruction is taken for the IC Interrupt Entrance register (location 11).

Fault Interrupt

If a meaningless function code (i.e., 00, 01, 77, 50 00, or 50 77) is executed, a special case of an internal interrupt will be generated. This fault interrupt will cause the next instruction to be executed from address 00000.

Externally Specified Index (ESI)

This outstanding feature provides peripheral devices with a means of specifying core storage areas in the computer's memory for any input or output transfers they may request. The Externally Specified Index (ESI) mode of operation is useful as a multiplexing device for a number of slow transfer peripheral units occupying one dual channel. The buffer control words governing the transfers are located at the index address. If input is desired, an Input Request is presented with the Index on one channel of the pair and the data on the other channel. If output is desired, an Output Request is presented with the Index address.

Externally Specified Addressing (ESA)

The ESA feature provides peripheral devices with a means of specifying an absolute core memory location for storage or retrieval of data. An active dual-channel mode of operation is required for computer response to this function. The address is presented on one channel and the data transmission path on the other. If input is desired, the external device presents an Input Request with the address and data. If output is desired, an Output Request is presented with the address.

Continuous Data Mode (CDM)

The Continuous Data Mode, requested when initiating a buffer on a channel, is a feature which provides an automatic reinitiation of the buffer upon completion. A new pair of buffer control words is transferred to the control memory buffer control addresses from the control memory CDM addresses for that channel. The Monitor Interrupt can be incorporated with the CDM, and, if so, the interrupt will occur each time the buffer is terminated and reinitiated. The CDM is especially useful when a continuous, high rate, stream of data must be transferred in or out of the computer.

ARITHMETIC

The Arithmetic Section, which contains a subtractive type ADDER, performs all the arithmetic and logical operations for the computer under direction of the Function Code Translator. The Arithmetic Section is used by Control for any address or operand modification requested by an instruction and for OVERFLOW detection if overflow exists at the completion of any arithmetic instruction except multiply.

CONTROL

The Control Section contains circuitry necessary to procure, modify, and execute the single address instructions of a program stored in the core memory of the computer. It controls parallel transfers of instructions and data. Direct or indirect addressing capabilities and automatic address and operand modification are directed by the Control Section translators and the timing of the synchronous electronic master clock. This section controls all arithmetic, logical, and sequential operations of the computer except those assigned to the Input/Output Section. It has facilities to permit an interruption of the running program when certain real time events require such interventions.

*When not assigned for these functions, the locations may be used for data storage.

MEMORY

The computer memory consists of up to 65,536 18-bit words of addressable storage locations divided into three distinct sections in a continuous addressing structure.

Control Memory

An independent high-speed core memory, consisting of 128 18-bit words, is used for index registers, buffer control words, real-time clock cells, real-time clock interrupts, and the fault interrupt address. The fixed addresses for these functions are:

Address	Assignment
00000	Fault Interrupt Entrance Register
00001-00010	8 Index Registers
00011	Intercomputer Time-Out Interrupt Register
00012	Real-Time Clock Interrupt Register
00013	Clock Overflow Interrupt Register
00014	Real-Time Clock Monitor Word Register
00015	Real-Time Clock Incrementing Register
00016	Synchronizing Interrupt Register
00017	Scale Factor Shift Count
00020-00037	Continuous Data Mode (Channels 0-7) or External Function Buffer Control Registers (0-7)
00040-00057	Output Buffer Control Registers (Channels 0-7)
00060-00077	Input Buffer Control Registers (Channels 0-7)

For UNIVAC 1219B Computers equipped with 16 I/O Channels:

00200-00217	Unassigned
00220-00237*	Continuous Data Mode (Channels 8-15) or External Function Buffer Control Registers (8-15)
00240-00257*	Output Buffer Control Registers (Channels 8-15)
00260-00277*	Input Buffer Control Registers (Channels8-15)

32-Word NDRO Memory (Bootstrap)

The computer is provided with 32_{10} nondestructive readout memory locations (00500_8-00537_8) which contain computer instructions and constants for an initial load program (Bootstrap). This provides the ability to enter an initial package of utility routines that may be used to load and/or debug more sophisticated programs. These memory locations have unique characteristics in that they are transformer cores which operate in a special type of nondestructive readout mode. They are not accessible to the programmer for store-type instructions.

NDRO (Bootstrap) Memory may be locked out by activating a switch on the console. If the Bootstrap Memory is locked out, addresses 00500-00537 in Main Memory are then available for other programming requirements.

Main Memory

The main memory consists of a 2-microsecond core storage that is used for program, constants, and data storage. All locations are accessible to the programmer at random and to all sections of the computer on a time-shared basis. Some locations are given special assignments which the programmer must respect and provide for their contents. The fixed addresses assigned to main memory are as follows:

Address	Assignment
00100-00117	External Interrupt Registers (Channels 0-7)
00120-00137	Unassigned
00140-00157	Output Monitor Registers (Channels 0-7)
00160-00177	Input Monitor Registers (Channels 0-7)
00300-00317	External Interrupt Registers (Channels 8-15)
00320-00337	Unassigned
00340-00357	Output Monitor Registers (Channels 8-15)
00360-00377	Input Monitor Registers (Channels 8-15)
00400-00477*	Unassigned
00600-00677*	Unassigned
00540-177777	Unassigned

*These addresses are in control memory when 256 words of control memory are used.

INPUT/OUTPUT CHARACTERISTICS

GENERAL OPERATION

All references to input or output in Table 1 and Figure 7 are made from the standpoint of the computer; that is, "input" is always input to the computer and "output" is always output from the computer. (For additional information, refer to UNIVAC Defense Computer Input/Output Design Characteristics.)

Communication with the UNIVAC 1219B Military Computer is carried on in an 18-bit parallel mode. The computer is provided with up to sixteen input and sixteen output channels, each logically independent of the others and each brought to its own cable connector (32 connectors in all). Each channel contains 18 information lines plus control signals. If it is desired to communicate with an external device requiring more than 18 bits of parallel data, a dual channel option may be selected by one of four (eight for 16 channel computer) switches on the control panel. The selected option logically combines a pair of sequentially numbered even and odd channels to form a single channel having 36 bits of parallel data plus one set of control lines. All signals on information lines and control lines are at two d-c levels which may be changed upon interchange of information. These may be held stable for microseconds or days, depending on the nature of the particular task. Information is transferred in parallel with external devices in a "Request-Acknowledge" basis. The computer program initiates the process by directing the device to perform an activity. A command is transferred via the output information lines on the output cable and identified as a command by the External Function line of the same output cable. When the device responds to the command and is ready to perform, it sets a Request line (Input Request line for an input activity and Output Request line for an output activity).

The computer responds to the input/output request at its convenience by storing the information presented on the Input cable, if input, and answering with an Input Acknowledge on the same cable or, if output, by transferring a word from storage to the output cable and answering with an Output Acknowledge on that cable.

Commands may be a single word command or a multi-word type depending on the requirements of the external device. If multi-word commands are required, the device presents an Output Request to the computer upon accepting the first word. The computer responds at its convenience and transfers each succeeding command word in a buffer mode. The External Function line is set each time a command word is placed on the output lines. In another method, the computer can "Force" command words to external devices; in which case, the Output Request line need not be set. (Refer to Instruction Codes 50 26 and 50 27.) As soon as the output register is available, the computer transfers the word to the device.

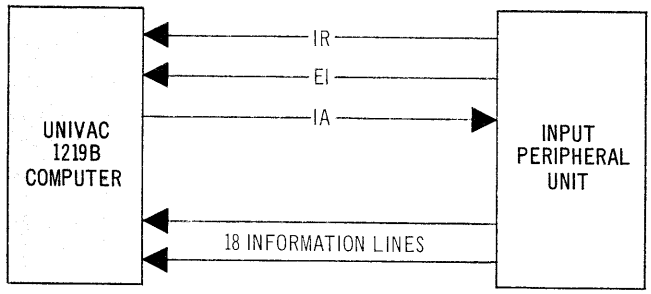
Input/Output Priority

The computer is scanning for input/output or interrupts during the time it is transferring input/output data or performing an instruction. If it finds an input or output request, it will not look for interrupt since I/O scan is performed ahead of Interrupt scan. If both Input and Output Requests are present, scan will alternate service. If both are not present, it will honor either.

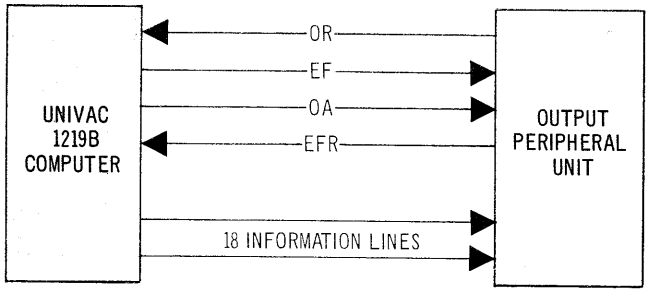
In the priority scheme, the higher numbered channel activity is honored first. The scanning of functions for priority determination is based on the computer's 2-microsecond cycle time. During any major sequence (i.e., one that requires a memory reference), one

TABLE 1 **DESCRIPTION OF INPUT/OUTPUT SIGNALS**

	SIGNAL NAME	ORIGIN	MEANING
Input Channel	Input Request (IR)	Peripheral Equipment	"I have a data word on the input lines ready for you to accept."
	Input Acknowledge (IA)	Computer	"I have sampled the word on the input lines."
	External Interrupt (EI)	Peripheral Equipment	"I have an Interrupt Code word on the input lines ready for you to accept."
Output Channel	Output Request (OR)	Peripheral Equipment	"I am in a condition to accept a word of data from you."
	Output Acknowledge (OA)	Computer	"I have put a data word for you on the output lines; sample them now."
	External Function (EF)	Computer	"I have put an External Function message for you on the output lines; sample them now."
	External Function Request (EFR)	Peripheral Equipment	"I am in a condition to accept an external function message from you."



ONE 1219B INPUT CHANNEL



ONE 1219B OUTPUT CHANNEL

Figure 7. **Input and Output Connections**

priority A data scan cycle occurs during the first microsecond and one priority B interrupt scan follows in the next microsecond if some inhibiting condition does not exist.

The A scan cycle selects top function priority in order as follows:

Order of Priority	Function
1	Real-time clock update
2	External Interrupt status word storage
3, 4	Output Request (or External Function) if preceding A scan sequence honored input
4, 3	Input Request if preceding A scan honored output or External Function Output or input if only one request exists.

If none of the above functions is detected during the A scan sequence, the interrupt B scan is initiated to operate during the next microsecond of the computer cycle. Otherwise, the A sequence is initiated to read control words from control memory for the transfer being honored and restarts the A scan. Priority A functions are again checked before returning to the program. The Priority B Scan for interrupts is inhibited during extended sequences because interrupts cannot be honored during the execution of an instruction. The scanning sequence is not effective when an interrupt lock-out exists.

The B scan cycle selects top interrupt priority as follows if no lock-out exists.

Order of Priority	Function
1	Program Fault
2	Intercomputer Time Out Interrupt
3	Real-Time Clock Monitor Interrupt
4	Synchronizing Interrupt
5	Real-Time Clock Overflow Interrupt
6	External Interrupt Monitor
7	Internal Interrupt from Output or External Function transfer with Monitor
8	Internal Interrupt from Input Transfer with Monitor

If no A event is detected during the A scan, the Priority B Interrupt Scan cycle is initiated to scan the B functions. Priority order 6, 7, and 8 functions are "channel dependent" and will be honored in highest channel number order. If one or more B events are detected, the highest priority function will be processed, and the A scan cycle will be restarted. If no B function is detected, the A scan cycle will be restarted at the end of the B scan.

I/O Word Transfer Timing (In Terms of Memory Cycle Time)

Scan time (to search for an I/O request) is concurrent with instruction time if channels are not busy and concurrent with word transfer time if channels are busy.

Order of I/O Sequence	I/O Sequence Name	Event Executed
1	$I/O\text{-}1_i$	Read current address word from controlmemory location specified by channel from I/O translator or externally specified index.
2	$I/O\text{-}1_i$	Read terminal address word from control memory next sequential address. Compare with current address word. Begin scanning for new request.
3	$I/O\text{-}1_f$	Modify and restore current address word to control memory.
4	$I/O\text{-}1_f$	Transfer data or function word (single or dual channel). If ESI, input from even channel, output on both.

Order of I/O Sequence	I/O Sequence Name	Event Executed (cont.)
5	I/O-2 if dual, or ESI, and ESA Terminate	Transfer second data or function word if dual channel operation. Even channel = most significant 18 bits.
6	I_f, R_f, W_f, or I/O-1_i	Complete the sequence of the instruction that was in process when the I/O request was detected or handle new I/O request.

External Interrupt — The EI from an external device may be considered as a request. It requests the computer to accept the 18-bit status word on the input data lines and store it in control memory (address 101 + 2k, k = channel No.). Accepting the EI signal sets the EI monitor for the particular channel and generates the EI monitor interrupt signal, which is handled as a separate request for I/O services.

Order of I/O Sequence	I/O Sequence Name	Event Executed
2	I/O-2 if dual, or ESI,ESA Terminate	Store the even channel interrupt code at 101 + 2k.
3	I_f, R_f, W_f, or I/O-1_i	Complete the instruction sequence that was in process when the request was detected, or handle new I/O request.

Special and Monitor Interrupts — If no I/O requests are detected during the first half of scan (Scan A), the computer checks for special and monitor interrupt signals during the second half (Scan B). This interrupts the normal sequence of instructions and transfers control to a special Interrupt Entrance Address.

After an interrupt is honored, the all interrupt lockout must be program cleared to honor another interrupt. This flip-flop, as well as the external lockout flip-flop, can be program set and cleared to allow control of interrupts. Also, the program can cause itself to wait for an interrupt.

Order of I/O Sequence	I/O Sequence Name	Event Executed
1	I_f, R_f, W_f, or I/O_f	Completion of any instruction in process or completion of a prevous I/O request.
2	Interrupt i sequence and I_i	Transfer control to interrupt entrance address determined by I/O translator if monitor interrupt or special interrupt translator is special instruction.
3	Interrupt f sequence and I_f	P will hold original program address. If the interrupt entrance address contains a RIL instruction, the computer will return to the program.
4	Int_f and I_f	If there is an indirect return jump stored at the interrupt entrance address, (P) will be stored and the computer will jump to a subroutine to handle the interrupt.

Control Signals

Control signals exchanged between the computer and peripheral equipment are summarized in Table 1. All control signals will be resynchronized to ensure that the control line has been returned to a logical zero state between successive recognitions of control signals (being reset to the one state). This requirement guarantees that only a single recognition pulse will be gen-

erated each time the control signal is set to a one state, and also is a safeguard against false gating of data lines caused by noise or other spurious signals appearing on the control lines. There is no such restriction on the information lines. These need not be cleared to zeros between successive words.

The Input Acknowledge is the computer response to an Input Request or to an Interrupt. To eliminate misinterpretation of the Input Acknowledge signal, peripheral equipment must not interrupt until its last Input Request has been acknowledged by the computer. A Request signal (or an Interrupt), once set, must remain set until it is acknowledged. This is necessary to maintain synchronism in the passing of data words back and forth between units. Under emergency conditions, when data loss is of secondary importance, the IR may be dropped but data lines must remain stable for not less than four microseconds. If during this four-microsecond interval an IA is received, the peripheral equipment may assume successful transfer of last data word. At any time, after the four-microsecond interval, the peripheral equipment may change the data lines and send an Interrupt. When these conditions prevail, an Input Acknowledge signal that occurs after the Interrupt is raised will be in answer to the Interrupt.

The Output Acknowledge or External Function signal is the computer response to an Output Request. Data or Command Codes are placed on the information lines and identified by the Output Acknowledge or External Function respectively.

The Override instruction to an intercomputer channel will not be executed until a Resume (acknowledge) is received from the receiving computer (until the Resume flip-flop is set) or by a Set Resume instruction in the program which would have to precede the override. Any delay, therefore, in an acknowledge from the receiving computer will hold up the program since the program will not proceed until the Override instruction has been executed, which will not occur until the Resume flip-flop has been set.

Information lines must be stable at the time they are gated into storage elements. This self-evident axiom dictates the timing relationships between the information lines and the control signals. In the case of computer output, this rule requires the computer to provide a suitable delay between gating information into output registers and raising the Output Acknowledge or the External Function signals, and similarly requires the computer to drop the Output Acknowledge or the External Function a suitable interval before changing the information on the lines. Thus, the peripheral equipment is guaranteed that Output lines will be stable for sampling any time the Output Acknowledge or External Function is present. In the case of input, the computer detects the Input Request or the Interrupt before sampling the data lines. The peripheral equipment cannot change the information lines after raising either its Input Request or Interrupt until an Acknowledge has been received, indicating that the data lines have been sampled (except as stated above).

SPECIAL MODES OF OPERATION

Dual Channel Operation

As previously mentioned, users of the computer have the option of communicating over the 18-bit parallel single channels or of logically combining sequential even and odd numbered channels into a 36-bit parallel dual channel. This option is selected by one of eight switches on the control panel. Selection of this dual channel option affects only that pair of channels selected, but both input and output channels of a given channel number are combined by a single switch setting. For example, if the first switch is activated, input and output channels 0 and 1 are combined to form a 36-bit input and a 36-bit output channel, but input and output channels 2 through 17_8 remain 18-bit logically independent channels. If this dual option is selected, the set of control lines belonging to the odd-numbered channel will have control of the information transfers over all 36 lines.

With this dual option selected, energizing of the Request lines or the Interrupt line on the even-numbered channel will cause the computer to interpret this as a desire to communicate in a single channel mode and the computer will reply over the even-numbered set of control lines. Eighteen bits of information will be accepted from the peripheral equipment on the even-numbered set of lines, and as in any single channel operation, identical information will appear to both sets of output lines.

Externally Specified Indexing (ESI)

ESI is usable only in a dual channel mode. The corresponding ESI switch (one of eight, for each pair of channels) must also be selected on the computer control panel. The index address word is always placed on the set of input lines for the odd-numbered channel. If the peripheral equipment desires to send an input word, it will place the input data word on the 18 lines of the even channel and raise the odd Input Request line. The computer will reply on the odd Input Acknowledge line. If the peripheral equipment should raise the even IR line instead of the odd, the computer will interpret this as normal single channel communication and ignore the index address, as explained above. The program must provide an active channel for the ESI operation (instruction 5001, 5002, 5011, or 5012).

If the peripheral equipment wants a word of output data, it will place the index address on the odd-numbered set of input lines and raise the odd Output Request. The computer will reply with 18 bits of data, duplicated on both sets of output lines, with the odd channel Output Acknowledge. Activation of the even-numbered Output Request will similarly cause the computer to ignore the ESI feature.

A sequence of events for output from a computer using ESI is as follows:

a. The computer program provides an active output channel to the equipment.
b. The external device places an even numbered 16-bit control memory index address, n, on the odd input channel.
c. The external device sets the Output Request control line on the odd output channel.
d. The computer detects the output request and at its convenience reads and compares the addresses stored in n and n+1.
e. The computer transfers the word from the address located in n+1 to both output channels.
f. The computer sets the Output Acknowledge line on the odd channel.

The sequence of events for input is the same as for output except that a data word is placed on the even input channel with the index address on the odd channel, and an Input Request is raised. The data word is stored at the absolute address stored in n+1. The computer responds with an Input Acknowledge.

Addresses n and n+1 are treated as normal buffer control words with the same buffer control options available. When the monitor interrupt occurs during the ESI input/output sequence, the index address n is stored in the associated monitor interrupt status word location and the channel is deactivated.

Externally Specified Addressing (ESA)

ESA is a switch selectable dual channel mode of operation that allows external devices random access to core memory for retrieval and storage of data as opposed to the contiguous addressing scheme associated with normal buffers. Individual and/or random entries in a data pool lend themselves to an effective use of ESA. The external unit must be capable of presenting the absolute address on the odd input channel of the pair.

A sequence of events for output from the computer using ESA is as follows:

a. The computer program provides an active output channel to the equipment. (The 5002 and 5012 instruc-

tions are used to activate the channel.)

b. The external device places a 16-bit absolute address, n, on the odd channel input lines.
c. The external device sets the Output Request Control line on the odd output channel.
d. The computer detects the Output Request and at its convenience transfers the address, n, to the S-register.
e. The computer places the contents of address, n, on both output channels of the pair.
f. The computer sets the Output Acknowledge on the odd channel.

The sequence of events for input is similar to output except that a data word is placed on the even input channel with the storage address on the odd channel and the Input Request line is raised. The computer responds with an Input Acknowledge. The computer channel is activated by using instruction codes 5001 and 5011.

I/O Transfer Under CDM

Assume a buffer has been initiated via a 5011 03 (Initiate Input on Channel 3) instruction, and the Input transfer has been completed. If the CDM bit were set (Bit 17 of Terminal Address Buffer Control Word), the contents of control memory addresses 26 and 27 would be transferred to addresses 66 and 67, and the input buffer for Channel 3 will have been reinitiated without program attention. Before the buffer, defined by the new BCWs, has been completed, the program has the option of storing another set of BCWs in address 26 and 27 with or without the CDM bit set. If set, this cycle continues until the program clears the CDM bit in location 26. Similar action occurs for output and external function buffers.

Intercomputer Communication Mode

Eight switches on the control panel provide the option of intercomputer communication to any or all input/output channels. The selection of a given I/O channel as an intercomputer channel has no effect upon the modes of the unselected channels. An intercomputer channel can function in either the dual or ESI mode in addition to the "normal" 18-bit mode.

The selection of a given channel as an intercomputer channel affects only the logic concerned with the Output and External Function Buffers. A channel which is sending data or external function messages to a given peripheral device holds the data in the output registers for a fixed minimum time period, after which any Output or External Function Request on any other channel that is part of this 4-channel group can cause the data to be changed. However, a channel sending data or External Function messages to another computer must hold the information in the output register/s until the receiving computer acknowledges receipt of those words. This acknowledge signal is received on what is known as the Output Request line when not on intercomputer mode. This line, in the intercomputer mode, is known as the Resume line. In the case of UNIVAC 1219B-to-1219B communication (see Figure 8), this Resume line is connected to the Input Acknowledge line of the receiving computer. Activation of the Resume signal on the transmitting computer channel causes the setting of the Resume flip-flop for that even or odd group of four channels. It is this flip-flop which, when set, allows the transmitting computer to proceed to the next highest priority output function (the next Output Data Word or External Function message). If an output channel is holding data for another computer and no Resume is received from that computer, the output registers will be tied up, until the intercomputer time-out interrupt branches to a remedial routine. During the interim, no Output Buffers or External Function Buffers to other equipment on that channel group can proceed. To limit the possibility of this hang-up occurring, two instructions and the intercomputer time-out interrupt are provided by which the computer program can monitor the status of the Resume flip-flop. These instructions are: Skip On No Resume (50 57 k) and Set Resume (50 20 k). The former allows examination of the Resume flip-flop, and the latter allows the program to correct the situation in which the "hang-up" exists.

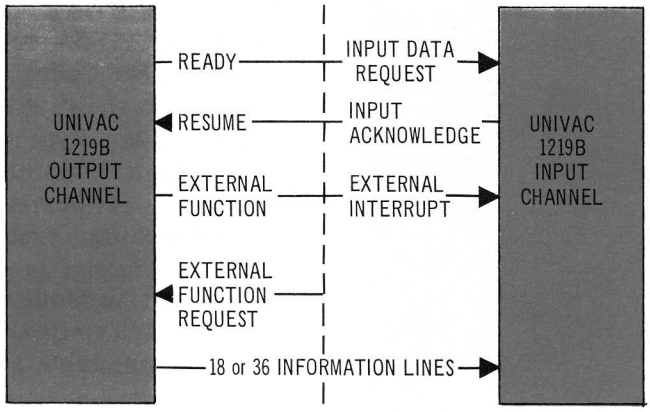
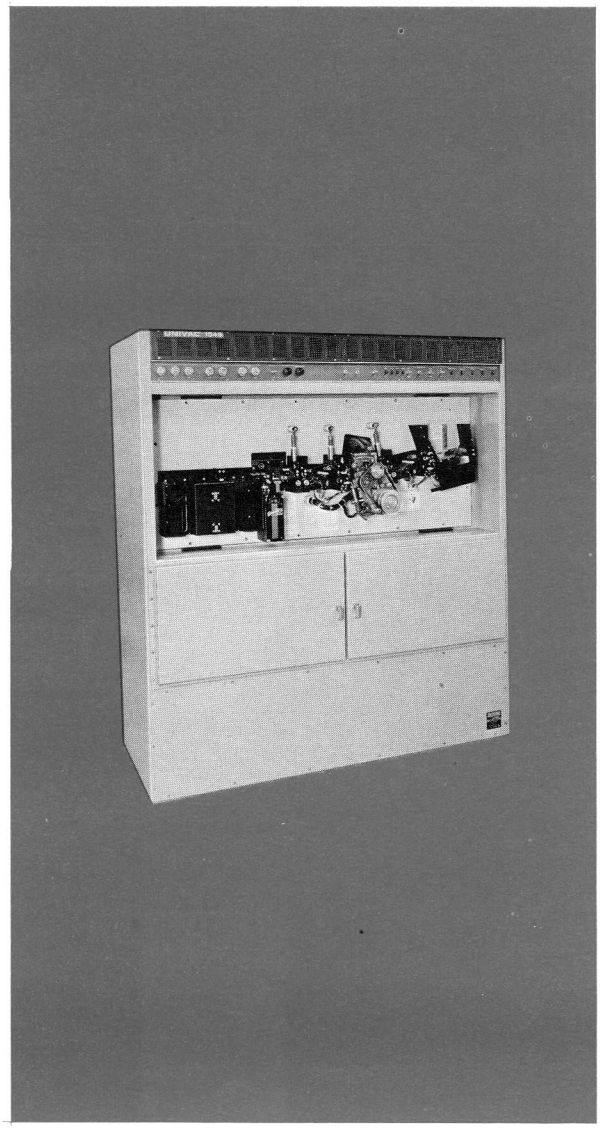


Figure 8. **1219B-to-1219B Communication**

PERIPHERAL EQUIPMENT

The following paragraphs contain brief descriptions of selected peripheral equipment available for use with the UNIVAC 1219B

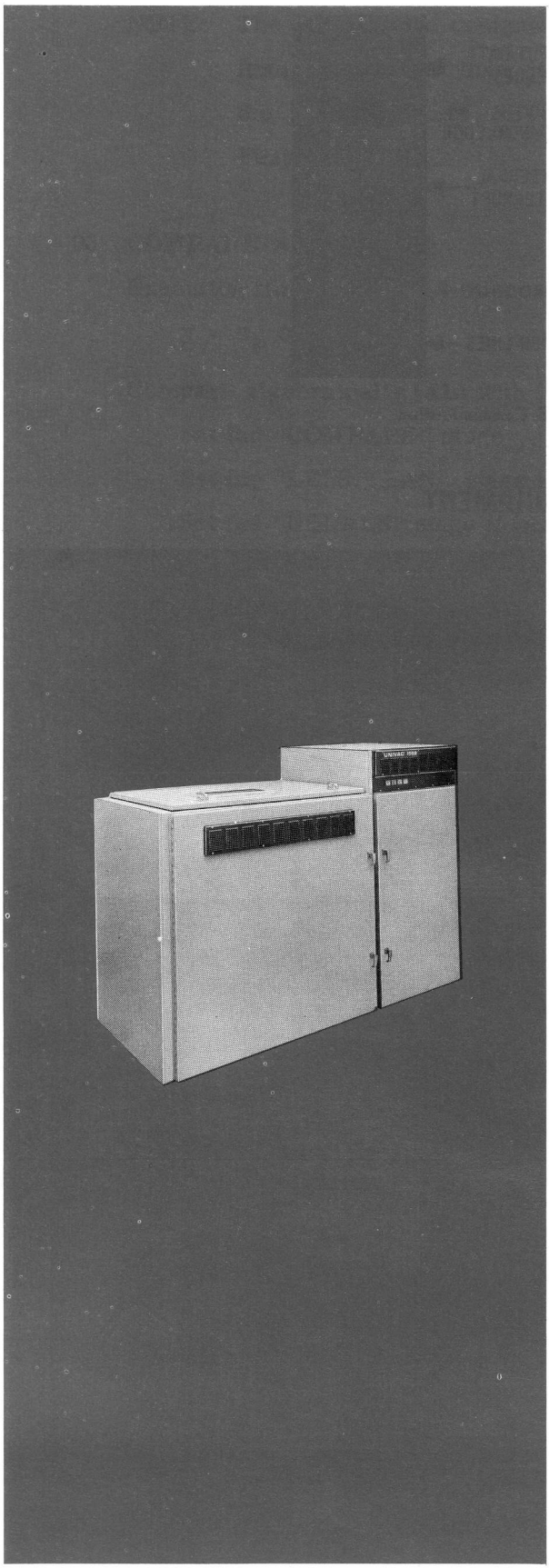


CARD READER-PUNCH-INTERPRETER

Inches: Height 65, Width 62, Depth 24.5
Centimeters: Height 165.1, Width 157.5, Depth 62.2

The UNIVAC 1549 Card Reader-Punch-Interpreter (CRPI) provides the required card reading, punching, and interpreting for the computer. In addition to the punched card unit itself, the CRPI cabinet houses a "common" control unit for other peripheral devices such as the UNIVAC 1533 Input/Output Keyboard-Printer and the UNIVAC 1569 High-Speed Printer. Included among the CRPI functional characteristics are the following:

Card Punching Speed: 200 cards per minute, nominal
Card Reading Speed: 330 cards per minute, nominal
Card Printing Speed: 32 cards per minute minimum (faster when less characters are printed)
Input Hopper Capacity: 500 cards minimum
Output Stacker Capacity: 500 cards minimum
Card Size: 3.250 inch by 7.375 inch — 80 column per RS-292
Hole Size and Shape: EIA Standard Rectangular per RS-292
Reading Method: Photoelectric



Punching: Paper or plastic cards
Number of Stackers: 2
Card Printer Code: XS3, others optional at extra cost
Card Printer Type Font Size: 0.090 inch high x 0.062 inch wide, 60 characters per line on each of two lines
Card Punching Verification: Post punch read
Card Printing Method: Print hammer strikes against drum containing 63 printable symbols and characters

HIGH-SPEED PRINTER

Inches: Height 44.8, Width 64.9, Depth 27.3
Centimeters: Height 113.8, Width 164.8, Depth 69.3

The UNIVAC 1569 High-Speed Printer is a means of data output for the computer. It accepts the results of the data processing in the form of digital data from the computer and prints it in alphanumeric form on multicopy paper. Physically, the high-speed printer has a common control unit with and in the UNIVAC 1549 Card Reader-Punch-Interpreter cabinet. In contrast, the 1569 printer cabinet contains the print head assembly, the associated drive electronics, and storage space for paper. Included among the high-speed printer functional characteristics are the following:

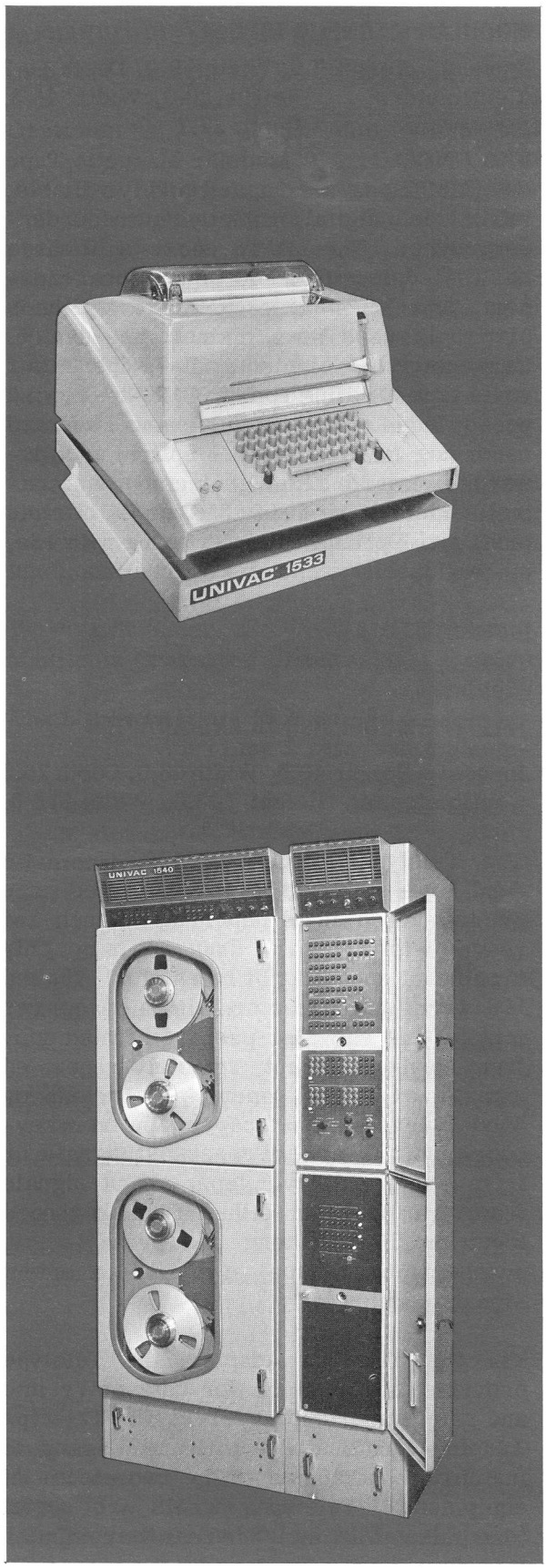
Printing Speed: 600 lines per minute max. 450 lines per minute average
Print Format: 120 or 132 characters per line; 10 characters per horizontal inch; 6 lines per vertical inch
Printed Characters: 63 plus space
Paper Width: 17-25/32 inches to approximately 5 inches
Print Type Font: Open Gothic characters
Printing Method: Print hammer impact against drum containing 64 characters in each character position of a line
Paper Form Length: 22 inches maximum
Paper Slew Rate: 8 inches per second
Vertical Format Control: punched tape loop
Paper Feed: pin feed type tractors (four)
Printer Code: XS-3

INPUT/OUTPUT KEYBOARD PRINTER

Inches: Height 13.3, Width 19.7, Depth 24.3
Centimeters: Height 33.8, Width 50.0 Depth 61.7

The UNIVAC 1533 Input/Output Keyboard-Printer consists of a Model 35 Teletype page printer and alphanumeric keyboard assembled into a compact unit which operates on a single input/output channel. The Keyboard-Printer operates under the control of the computer program. It accepts instructions from the computer via the UNIVAC 1549 Card Reader-Punch-Interpreter, which houses the keyboard-printer control circuitry. Included among the functional characteristics are the following:

Printing Speed: 10 characters per second
Keyboard Input: printed for all operations
Keyboard Generated Codes: ASCII
Printer Response Codes: ASCII
Printed Format (10 characters per second printer): 72 characters per line
10 characters per inch, horizontal
6 lines per inch, vertical



MAGNETIC TAPE UNIT

Inches: Height 72.0, Width 38.0 (2 transports and control), Depth 29.0 (air cooled)
Centimeters: Height 182.8, Width 96.5, Depth 73.6

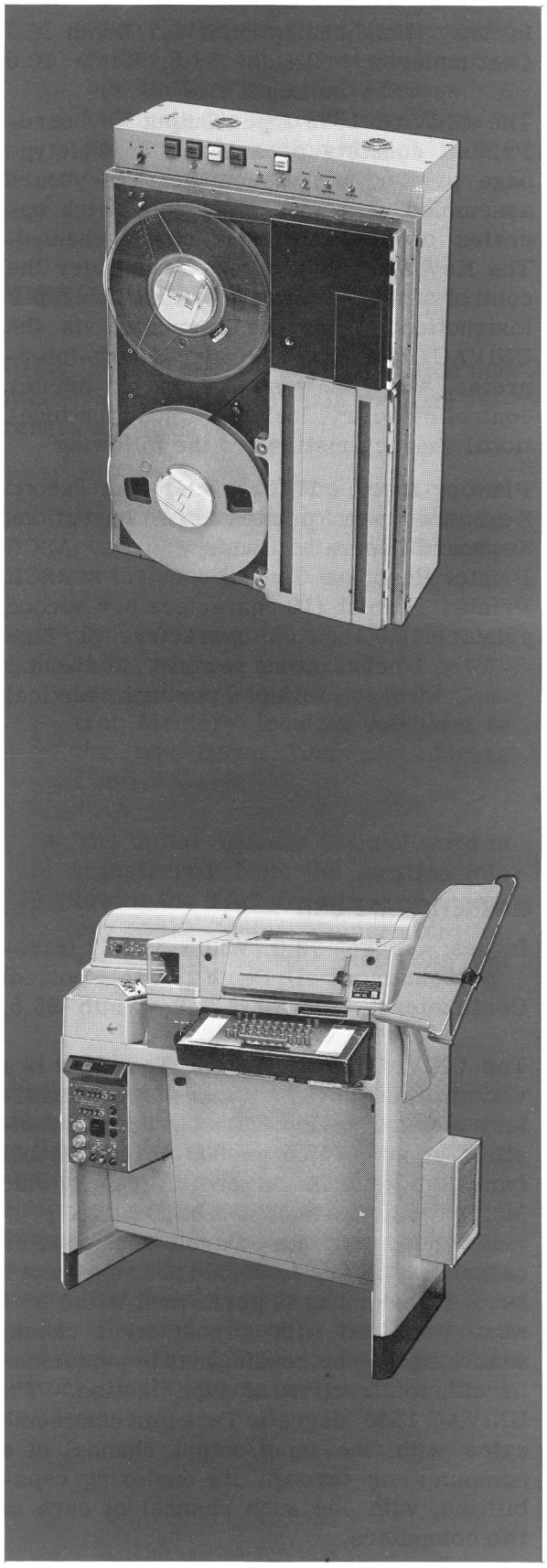
The UNIVAC 1540 Magnetic Tape Unit is a versatile, large capacity, militarized auxiliary storage device. Data recording densities of 200, 555.5 or 800 seven-channel frames per inch, on one-half inch wide Mylar* tape up to 3600 feet long, are provided under program control. It is capable of transferring up to 96,000 characters per second. Recording is performed in the forward direction with a post write check; searching and/or reading may be performed in either direction of tape motion. The UNIVAC 1540 Magnetic Tape Unit communicates with one input/output channel of a computer or through its duplexing capabilities, with one such channel of each of two computers.

*Mylar is a trademark of the DuPont Co.

MODULAR MAGNETIC TAPE SET

Inches: Height 7.9, Width 19.0, Depth 10.9
Centimeters: Height 20.0, Width 48.3, Depth 27.7

The UNIVAC 1840 Modular Magnetic Tape Set (MMTS) is a compact, highly reliable, militarized, digital, magnetic tape recorder-reproducer. The MMTS consists of three basic modules: Control Unit, Tape Transport, and Maintenance Console which is primarily used as a maintenance device. Characteristics include: A 75-ips read/write tape handling speed; A 150-ips rewind speed; Industry-accepted 7-channel forward recording (with 9-channel option) and forward/backward reading at computer-controlled densities of 200, 556, or 800 bits per inch; and Options for interfacing with 18-, 30-, or 32-bit computers.



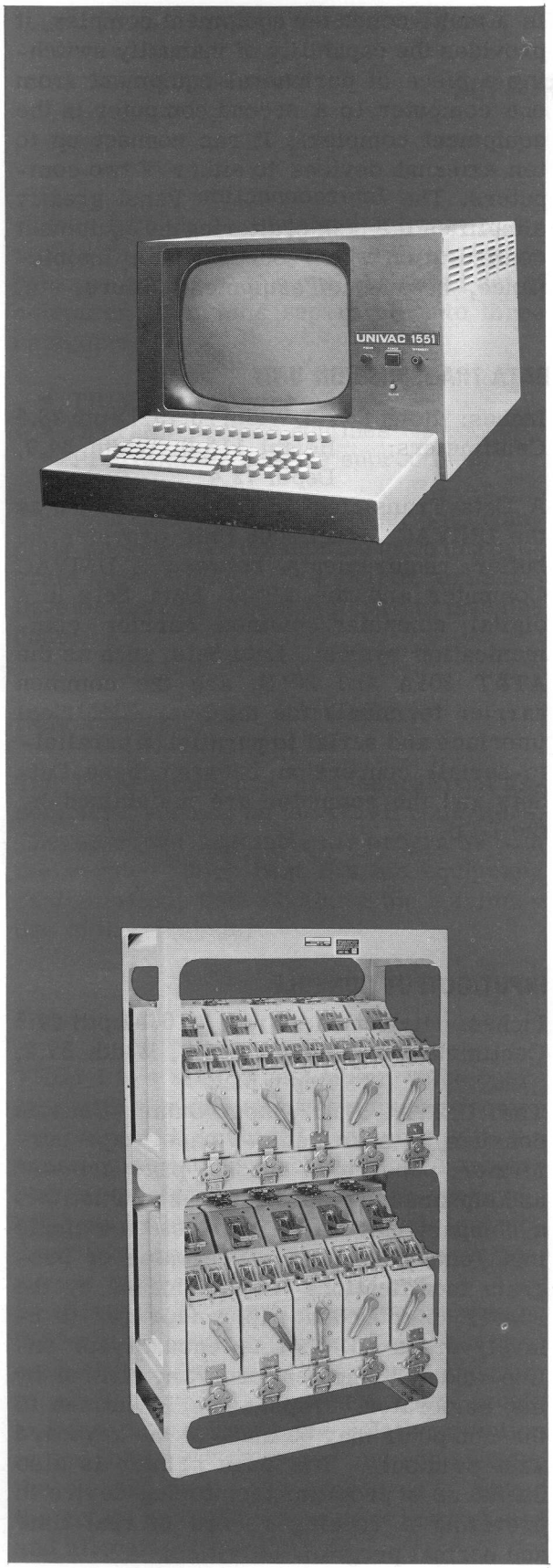
TELETYPE MODEL ASR-28 AND ADAPTER

Inches: Height 40.5, Width 46.0, Depth 26.5
Centimeters: Height 103.0, Width 116.5, Depth 67.3

The Teletype* Model ASR-28 Automatic Send-Receive set is a code-actuated page printer and 5-level paper tape punch and reader with keyboard input. It is not only capable of operating at rates of 60, 75, and 100 words (5 characters plus a space) per minute, but can also prepare printed copy from keyboard input, remotely generated signals, or 5-level paper tape. It can produce: 5-level punched paper tape from keyboard input, remotely generated signals, or from another 5-level tape; output signals from transmission either from keyboard input or 5-level tape; and printed copy, punched paper tape, and remote output signals concurrently.

Via the Teletype Adapter, the Teletype provides one medium for data entry into and retrieval from the 1219B. The Teletype Adapter contains the logic necessary to match the interfaces of the two pieces of equipment. Teletype signals are serial (one pulse at a time), whereas the computer transfers data in the parallel mode.

*Trademark of Teletype, Inc.



ALPHANUMERIC DISPLAY UNIT

Inches: Height 16, Width 24, Depth (with keyboard) 34

Centimeters: Height 40.6, Width 61, Depth (with keyboard) 86.4

The UNIVAC 1551 Alphanumeric Display Unit is a highly reliable display device for presenting and monitoring information processed within a computing system. It consists of a desk-top cabinet containing a 17-inch cathode ray tube for displaying 25 lines of 80 characters each in text-type format, a keyboard for data control and function selection, a power supply, core memory, and the necessary logic, character generating, and control electronics. The unit is completely modular and is packaged in a ruggedized cabinet. Included among the functional characteristics are the following:

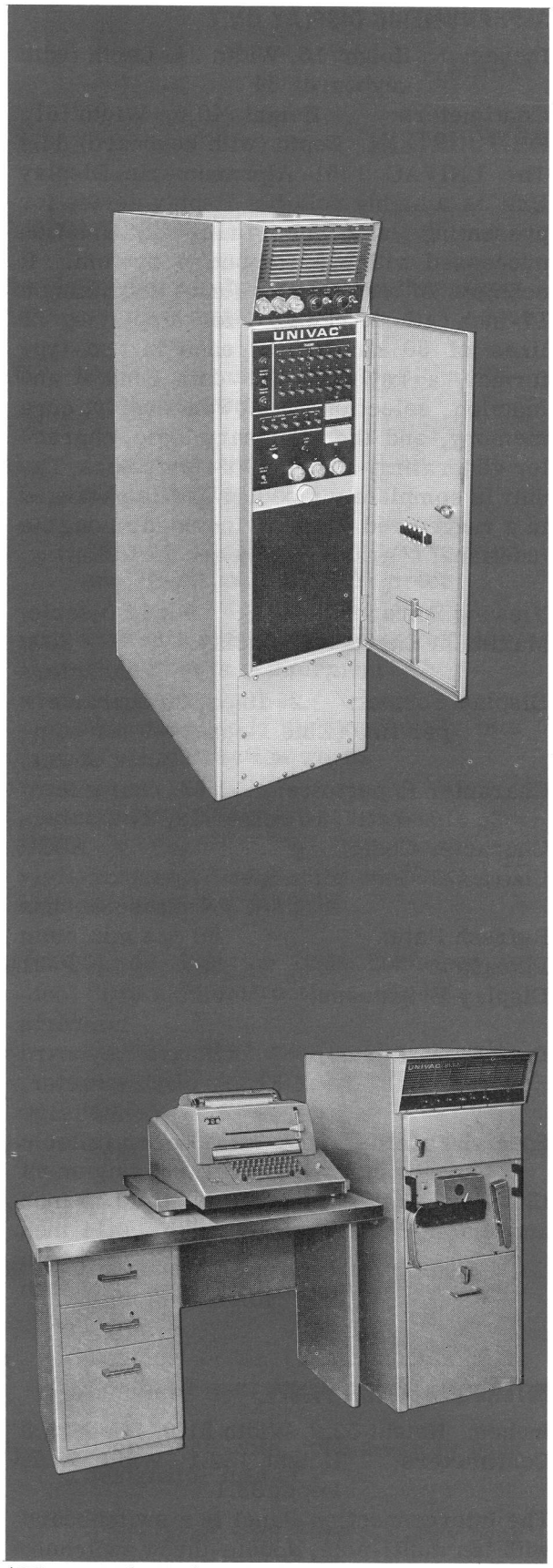
Viewing Surface:	9 x 12 (usable)
Maximum Character Display:	2000 characters
Display Format:	25 lines, 80 characters per line (Line 1 reserved for computer status)
Character Repertoire:	63 characters: 26 alphabetic, 10 numerals, 27 symbols
Character Code:	ASCII
Character Generation Speed:	Approximately 4 microseconds
Refresh Rate:	50 cps minimum
Phosphor:	P-31
Display Brightness:	Variable 0-50 foot-lamberts
Data Entry Device:	Standard keyboard
Editing Capability:	Erasure by character, line, or message
Special Function Keys:	14 keys available for customer function assignment
Character Size and Spacing:	Height - 0.15 inch Width - 0.11 inch Horizontal Spacing - 0.15 inch Vertical Spacing - 0.36 inch

INTERCONNECTION PANEL

Inches: Height 52.0, Width 31.0, Depth 15.0

Centimeters: Height 132.1, Width 78.7, Depth 38.1

The Interconnection Panel is a switchboard with ten multi-pole, double-throw switches.



*The Keyboard and Printer are optional items.

In a multi-computer equipment complex, it provides the capability of manually switching a piece of peripheral equipment from one computer to a second computer in the equipment complex. It can connect up to ten external devices to either of two computers. The Interconnection Panel greatly simplifies the task of altering the equipment configuration when necessary for maintenance, because of equipment failure, etc.

DATA TRANSMISSION UNIT

Inches: Height 44.5, Width 13.75, Depth 28.5
Centimeters: Height 113.0, Width 34.9, Depth 72.4

A Data Transmission Unit (DTU), such as the UNIVAC Model 2008 Unit, provides the buffer requirements between a UNIVAC Computer and the AT&T Data Sets in a digital computer/common carrier communication system. Data Sets, such as the AT&T 201A and 201B, are the common carrier terminals for the line. Electrical interface and serial to parallel (& parallel-to-serial) conversion between these Data Sets and the computer are maintained by the DTU.

INPUT/OUTPUT CONSOLE

Inches: Height 57, Width 22.5, Depth 24.5
Centimeters: Height 144.7, Width 57.2, Depth 62.2

The UNIVAC 1532 Input/Output Console consists of a ruggedized paper tape perforator, paper tape reader, page printer* and alphanumeric keyboard* assembled into a compact unit which operates on a single input/output channel. Programs or program modifications may be loaded by the reader on punched paper tape (5- to 8-level) prepared off-line manually or on-line under computer program control by the perforator. Alphanumeric entries to the computer may be made at the keyboard with printout. The page printer is also useful as a program monitoring device in providing a running record of real-time and normal program activities.

SUPPORT SOFTWARE

Univac furnishes a standard software package with 1219B systems. This software package consists of a well balanced set of computer programs separating into three categories:

- TRIM assemblers
- Operator service routines
- Programmer service subroutines.

The programs come to the user with complete program documentation including program specifications, flow charts, and side-by-side listings (TRIM symbols vs machine code).

TRIM ASSEMBLERS

The TRIM family has three operational assemblers running on the 1219B computers. The user can assemble his programs with the version which best fits his equipment configuration, thus obtaining the maximum use of the computer.

Trim I

TRIM I is a simple assembler which operates with a minimum of equipment, requiring only a 4K memory computer with the paper tape reader-punch unit. The assembler translates monocode (one-to-one) symbolic operations into machine code instructions with appropriate address allocation.

In operation, TRIM I makes two passes on the source program tapes. The first pass stores the labels from the source program to allow forward references. These labels and indicators giving the relative position in the program are stored and retained for the second pass. The second pass makes the actual assembly of machine instructions and allocates the addresses.

Trim II

The TRIM II program is a two pass assembler which operates on an 8K memory computer with a paper tape reader-punch unit. It will convert a source program written symbolically, absolutely, or in combination thereof into an object program tape suitable for loading into the computer. The TRIM II program accepts both monocode (one-to-one) mnemonic operations and polycode (one-to-many) mnemonic operations in the source program. The source language also has available debugging aids which cause dumps of registers and memory locations whenever desired by the programmer. Paper tape outputs are available with TRIM II.

Trim III

TRIM III is an assembler which operates on a 16 K memory computer with a UNIVAC Magnetic Tape Unit (2 transport) paper tape reader-punch unit, a console typewriter, and a high-speed printer.*

Provided to users on magnetic tape, this single pass assembler converts a source program written symbolically, absolutely or in combination thereof into an object program tape suitable for loading into the computer. TRIM III accepts both monocode (one-to-one) mnemonics and polycode (one-to-many) mnemonic operations in the source program. The assembler has available a source language librarian to aid the programmer in selecting subroutines for incorporation into the program during the assembly process, debugging aids which will dump registers, memory locations or will cause program stops whenever desired by the programmer, and a source language correction capability in conjunction with an assembly run.

*Optional

The source programming language includes the operations of TRIM I and TRIM II. Debugging-aid operations in this language cause generations which present diagnostic information to the programmer during a run. This works with the TRIM DEBUGGING PAK discussed later. Programmers have complete assembler "Control" via CONTROL operations and library retrieval commands via CALL operations.

A typical TRIM source program appears below:

```
CVRT[12]  PROG • WGH • FEB[63]
CVRT[12]  0 • 0                  DECIMAL TO OCTAL CONVERSION
          ENTAU • TMPR[4]        WORD TO CONVERT
CVRT[13]  ENTAL • CVRT[73]       WORKING STORAGE
          LSHAL • [2]
          ADDAL • CVRT[73]
          LSHAL • [1]
          STRAL • CVRT[72]       INTERMEDIATE WORKING STRGE
          ENTALK • [0]
          LSHA • [6]             LEFTMOST CHAR TO A
          ADDAL • CVRT[72]       ADD INTERMED WORKING STRGE
          STRAL • CVRT[73]       STORE IN CUMULATIVE WORKING
                                 STRGE
          BJP • CVRT[13]
          IJP • CVRT[12]
TMPR[4]   0 • 0
CVRT[72]  0 • 0
CVRT[73]  0 • 0
• •
```

Trim Library Builder

The TRIM III Library Builder program allows the user to:

- Record a library directory and a library of program routines on magnetic tape.
- Correct (insert, replace, or delete) any of the library program-routines and update the library directory.
- Perforate a tape with the library directory, library, or any individual library routine.
- Print the library directory, library, or any individual library routine.

Trim Assembler Outputs

The following outputs are available with TRIM Assemblers.

Trim Assembler Outputs		TRIM I	TRIM II	TRIM III
Output No. 1	Program Summary			TRIM
Output No. 2	Program summary and side-by-side listing in source code on perforated tape.	X	X	X
Output No. 3	Absolute assembled program in source code on perforated tape.	X	X	X
Output No. 4	Absolute assembled program in bioctal code on perforated tape.	X	X	X
Output No. 5	Relocatable assembled program in bioctal code on perforated tape.	X	X	X
Output No. 6	Allocation information on perforated tape.		X	X
Output No. 7	Ignore all CALL operations.			X
Output No. 10	Assembled program (combination absolute and relocatable) on magnetic tape.			X
Output No. 11	Source program on perforated tape in source code.		X	X
Output No. 12	Side-by-side 1004 Printer listing.			X
Output No. 13	Assembled program on 80 column cards (absolute or relocatable).			X
Output No. 14	Similar to Output No. 12 for 11 x 8½ printer page size.			X
Output No. 15	Source program on 80 column cards (one source statement per card).			X
Output No. 16	Source program on magnetic tape.			X

Trim Debugging PAK

This diagnostic routine presents information about the user's program while it runs on the computer. It acts under direction of debugging-aid statements placed in the source program by the programmer.

Trim Corrector

The TRIM Corrector program is a companion to the UNIVAC TRIM Assemblers using paper tape source program input. The correction process, constituting a separate computer run, provides a convenient method for correcting source programs for subsequent assembly. Input to the corrector consists of one or more correction tapes and the source program tape(s) to be corrected. Three types of corrections can be made: (1) insertions, (2) deletions, and (3) replacements. The output is a single punched paper tape consisting of the corrected source program.

Track

Up to five designated areas of a program being executed on the computer can be traced by the TRACK program. The following information is available as an output.

1. Address of the executed instruction
2. The executed instruction
3. Contents of pertinent registers
4. The executed instruction with the operator in TRIM symbolic language.

OPERATOR SERVICE ROUTINES

Operator routines are those used by the computer operator, under console control, to perform computing center operations. Such routines perform handling service to the user; however, they do not become integrated into his program. Operations performed by the utility package are listed in Table 2.

Upak I

UPAK I is a collection of seven routines on perforated tape in relocatable format which provides paper tape input/output functions and examination and alteration of memory for program debugging.

The load and dump routines are operable under control or manually from the computer console. The inspect and change and store constant in memory routines are operable from the computer console only.

Upak M

UPAK M is a modified version of UPAK I. In addition to providing the same functions of UPAK I, it facilitates six-digit addressing when desired.

Upak II

UPAK II is a collection of eight routines on perforated tape in relocatable format which provides input/output functions and examination and alteration of memory for program debugging.

The load and dump routines are operable under program control or manually from the computer console. The inspect and change and store constant in memory routines are operable from the computer console only. The UPAK II program may be loaded anywhere in the computer memory with the single restrictions that the entire package must be wholly contained within a single memory bank.

TABLE 2 UTILITY PACKAGE OPERATIONS

1219B UPAK Routines	UPAK I	UPAK M	UPAK II	UPAK III
TRIM III output No. 10 load			X	X Module 4
Absolute typewriter code load	X	X	X	
Absolute bioctal code load	X	X	X	
Relocatable bioctal code load	X	X	X	
Absolute typewriter code dump	X	X	X	
Absolute bioctal code dump	X	X	X	
Inspect and change	X	X	X	X Module 5
Store constant in memory	X	X	X	X Module 5
Paper tape handler				X Module 1
Magnetic tape handler				X Module 2
Magnetic tape duplication				X Module 3

Upak III

UPAK III is a modular utility system comprised of stacked programs on magnetic tape. A control program loaded by magnetic tape bootstrap or paper tape load program accomplishes loading of one or more of the component modules.

PROGRAMMER SERVICE SUBROUTINES

Programmer subroutines are those service routines which the programmer assembles with his routine. These subroutines exist in source language for easy integration into the user's program.

Listed are some of the standard subroutines provided:

Floating Point Pak - This performs floating operations using a 36-bit floating number (two 18-bit words). This has a sign bit, an 8-bit characteristic, and a 27-bit mantissa. It performs add, multiply and divide operations.
Square Root
Sine
Cosine
Arctangent
Natural Log
Arcsine
Octal-to-Decimal Conversion
e^x
Decimal-to-Octal Conversion

OPTIONAL SOFTWARE

Larger, more efficient and convenient language processors are available from Univac where certain minimum system configurations exist.

- ULTRA/18 Macro Assembler
- FORTRAN/18 Compiler
- SYCOL/18 Compiler

ULTRA/18 is a powerful and flexible two-pass macro assembler that converts programs coded in simple symbolic mono instruction and programmer-defined macro language into relocatable object machine code. A 32K memory computer coupled with a monitoring typewriter, card reader, card punch, high-speed line printer, and two-transport magnetic tape unit comprise the minimum ULTRA/18 operable system. With this easy-to-learn assembler that translates simple free-form English-oriented syntax statements, operating programs with complete documentation can be generated for any computer.

- Machine independent macro interpretation with nesting up to 63 levels.
- Optional generation for other computers.
- Assembler directives for control, flexibility, and versatility.
- Interprogram and library program unit linkage.
- Imbedded or substituted computer instruction repertoire.
- Error flagging, program documentation, and listing.
- Arithmetic, logical, and conditional expression evaluation with multiple level nesting.

The algebraic FORTRAN/18 Compiler converts programs written in a problem-oriented language, closely resembling mathematical expressions, into machine code for an 18-bit UNIVAC computer. This one-pass compiler requires a 16K memory computer with a monitoring typewriter, card reader, card punch, high-speed line printer, and two magnetic tape transports for its minimum operable configuration. Compiling the BASIC FORTRAN language has been enhanced in FORTRAN/18 by providing additional language features and the ability to add TRIM III subroutines to the FORTRAN library. The compiled object program can be placed on cards or magnetic tape for future operating runs, or through the "compile and go" feature can be initiated for execution immediately.

SYCOL/18, designed primarily for use in real-time command and control applications, is a problem oriented language compiler that operates on a system configura-

tion containing a minimum of a 16K memory computer, perforated paper tape reader and punch, monitoring typewriter, card reader, card punch, high-speed line printer, and four magnetic tape transports. TRIM-compatible outputs of the generated machine operating program are provided on perforated paper tape or on magnetic tape.

The SYCOL/18 language is a subset of the basic SYCOL and includes the fixed point data declarations and problem solving operations. Its assembly language subset is the TRIM language. Almost complete freedom is allowed in mixing assembly language and problem-oriented language thereby making it a flexible tool for the programmer.

SYMBOLS CONVENTIONS AND INSTRUCTION WORD FORMATS

SYMBOL CONVENTIONS

The following symbols are used throughout the descriptive material.

AU	Upper accumulator, 18-bit arithmetic register
AL	Lower accumulator, 18-bit arithmetic register
A	AU and AL linked together to form one 36-bit arithmetic register
B	Contents of the active index register, 18-bit one's complement
f	Function code, high order six bits of all instruction words
F	Function register, seven bits
k	Designator contained in Format II instructions, six bits
m	Minor function code contained in Format II instructions, six bits
M	Memory word specified by (y),(y+B), L(y) (AU) or L (y+B) (AU) of Compare Instruction
NI	Next instruction
P	The Program Address register
SR	Special register, 5-bit core memory bank designator
u	The low order 12 bits contained in Format I instruction words
u_P	u prefaced with core memory bank designator bits of P
u_{SR}	u prefaced with core memory bank designator bits of SR
y	u extended or u_P or u_{SR}
Y	The address or constant formed by y or y+B with or without sign extension
( )	Contents of the address or register
$( )_i$	Initial contents of the address or register
$( )_f$	Final contents of the address or register
$( )_n$	Designates any single nth bit of the contents of a register
(Y+1, Y)	Designates the contents of two consecutive memory locations linked together to form a 36-bit word. Address Y+1 contains the most significant half of the word while address Y contains the least significant half
:	The colon in a logical expression indicates COMPARISON
L() () or ()⊛()	The bit-by-bit or logical product (logical AND) defined by the table:

	01
0	00
1	01

()v()	Logical sum, or inclusive OR defined by the table:

	01
0	01
1	11

$()\dot{\oplus}()$	Half add, half subtract, or exclusive OR defined by the table:

	01
0	01
1	10

( )' or $\overline{( )}$ The one's complement of the contents of the address or register

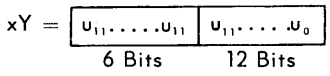
( ) ( ) Algebraic product of the contents of two locations

(Y) When the contents of Y are used as an address, only that lower portion of the word that can be contained in S is transferred.

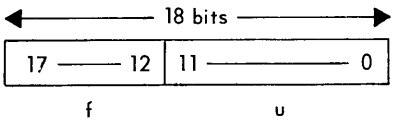
➡ Transfer the quantity stated at the left of the symbol to the address or register stated at the right of the symbol

"Console" and "Control Panel" are used to designate I/O console or the computer control panel.

xY x preceding some symbol indicates that the sign of the 12-bit constant has been extended to produce an 18-bit word, i.e.,



FORMAT I

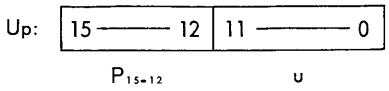


f: function code, six high order bits
u: twelve low order bits

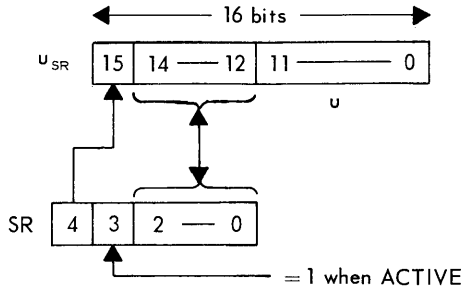
The definition and usage of u are determined by the function code utilizing u in two distinct manners:

- u used as a constant. In this case, u itself is the operand and requires no further memory reference; however, u is "extended" to 18 bits. (Refer to List of Instructions.)
- u used as an address. In this case, u is used as the lower order 12 bits of the base address referring to a memory cell. The base address is 16 bits, designated as u_P or u_{SR}, and is described below:

u_P is defined as a 16-bit address whose four high order bits consist of the four higher order bits of P and whose twelve low order bits are u.

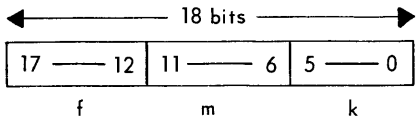


u_{SR} is defined as a 16-bit address whose four high order bits consist of the three lower order bits and the high order bit of SR and whose twelve low order bits are u.



Certain Format I instructions allow the use of either u_P or u_{SR} as the operand address; for these instructions u_{SR} is ACTIVE and u_P is used whenever SR is INACTIVE. (Refer to List of Instructions.)

FORMAT II



f: six-bit function code (always equal to octal 50)
m: six-bit minor function code
k: six low order bits

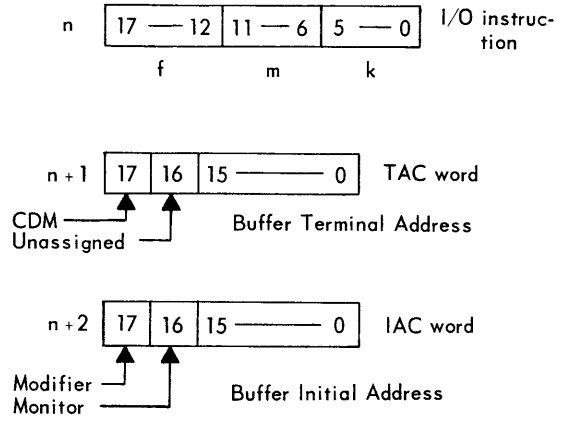
Format II instructions perform a variety of operations and can be classified in two instruction categories:

- No memory address needed. In this case, the information existing in the

computer's arithmetic or control registers and the value k are sufficient to perform the specified operation.

- Initiate input/output buffer. In this case, the two memory cells immediately following the instruction are used to contain the buffer control words. The complete instruction must therefore occupy three sequential memory cells; the format is as follows:

Any address



bit 17: CDM of n+1 (Terminal Address Control Word) is the Continuous Data Mode Identifier. If equal to one, the computer I/O section operates in the continuous data mode. If equal to zero, a normal buffer is executed.

bit 17: Modifier of n+2 (Initial Address Control Word); if equal to one, the Buffer Current (Initial) Address is decremented for each word transferred in or out; if equal to zero, the Buffer Current Address is incremented.

bit 16: Monitor of n+2 (Initial Address Control Word); if equal to one, the Monitor Interrupt occurs upon successful completion of the last transfer; if equal to zero, no Monitor Interrupt will occur.

NOTE: Normal buffer termination occurs when the incremented/decremented Buffer Current Address is equal to the Buffer Terminal Address. A buffer is terminated when tests in the control section detect Buffer Control Address equality.

I/O BUFFER INITIATING INSTRUCTIONS

During the execution of any instruction that initiates a buffer, three main memory references are involved.

- The I/O instruction is extracted from memory and interpreted by the control section and sets I/O active on the specified channel.
- The Terminal Address Control word is transferred from the location following the I/O instruction to the Control Memory location assigned to that type buffer Terminal Address Control Word.
- The Initial Address Control word is transferred from the location following the TAC word in main memory to the Control Memory location assigned to that type Buffer Current Address Word.

Computer control reads the next sequential instruction and continues the program leaving the Input/Output Section with the task of handling the transfers. The Input/Output Section generates the addresses in Control Memory to examine the control words placed there by the steps above when it receives a request for word transfer from the device on the activated channel. For the actual word transfer the I/O Section robs one Main Memory cycle from the program being executed.

LIST OF INSTRUCTIONS

FORMAT I INSTRUCTIONS

The following pages list the repertoire of instructions for the computer. Common usage and example cases are included for instructions where the meaning may not be obvious; no attempt has been made to indicate more sophisticated use. The instructions are listed and defined in the following format:

- (Octal Code) (Instruction Name) ("TRIM" Code) (symbolic summary)
- (execution time)
- (definition of the "y" address or constant)
- (text defining the instruction in detail)
- (examples and/or notes if any)

The "symbolic summary" expression will use the symbol "Y" to include "y" or "y+B", whichever is stated in the text for that instruction.

00 FAULT CODE; JUMP TO FAULT ENTRANCE REGISTER

Execution time: 2 microseconds

01 FAULT CODE; JUMP TO FAULT ENTRANCE REGISTER

Execution time: 2 microseconds

02 COMPARE AL (CMAL) (AL) : (Y)

Execution time: 4 microseconds

$y = u_P$ or u_{SR}

Compare algebraically (AL) with (y) and set the comparison designator as follows:

Set the "COMPARE" stage

Set the "LESS THAN" stage if (AL) < (y)

Set the "EQUALS" stage if (AL) = (y)

$(AL)_f = (AL)_i$

NOTE: The comparison designator is cleared by the execution of any subsequent instruction other than codes 60-67, and no interrupt will be honored while the designator is SET. (Refer to paragraph entitled Conditional Jump Features.)

03 COMPARE AL (CMALB) (AL) : (Y)

Execution time: 4 microseconds

$y = u_P$ or u_{SR}

Compare algebraically (AL) with (y+B) and set the comparison designator as follows:

Set the "COMPARE" stage

Set the "LESS THAN" stage if (AL) < (y + B)

Set the "EQUALS" stage if (AL) = (y + B)

$(AL)_f = (AL)_i$

NOTE: The comparison designator is cleared by the execution of any subsequent instruction other than codes 60 - 67, and no interrupt will be honored while the designator is SET. (Refer to subsequent paragraph entitled Conditional Jump Features.)

04 SELECTIVE SUBSTITUTE (SLSU) L(AU)' (AL)+L (AU) (Y) ➡ AL or $(Y)_n$ ➡ AL_n for $(AU)_n = 1$

Execution time: 4 microseconds

$y = u_P$ or u_{SR}

Replace the individual bits of (AL) with bits of (y) corresponding to ones in (AU), leaving the remaining bits of (AL) unaltered.

$(AU)_f = (AU)_i$

Example of selective substitute:

$(AU)_i$ = 007777 Mask

(y) = 123451

$(AL)_i$ = 666666

$(AL)_f$ = 663451

05 SELECTIVE SUBSTITUTE (SLSUB) L(AU) (AL) + L(AU) (Y) ➡ AL

Execution time: 4 microseconds

$y = u_P$ or u_{SR}

Replace the individual bits of (AL) with bits of (y+B) corresponding to ones in (AU), leaving the remaining bits of (AL) unaltered.

$(AU)_f = (AU)_i$

06 COMPARE WITH MASK (CMSK) L(AU) (AL) : L(AU) (Y)

Execution time: 4 microseconds

$y = u_P$ or u_{SR}

Compare algebraically selected bits of (AL) with corresponding bits of (y) and set comparison designator as follows:

Set the "COMPARE" stage

Set the "LESS THAN" stage if L(AL) (AU) < L(y) (AU)

Set the "EQUALS" stage if L(AL) (AU) = L(y) (AU)

$(AL)_f = (AL)_i : (AU)_f = (AU)_i$

NOTE: The comparison designator is cleared by the execution of any subsequent instruction other than codes 60-67, and no interrupt will be honored while the designator is SET. (Refer to paragraph on Conditional Jump Feature.)

Example of Compare with Mask:

$(AU)_i$ = 007777 Mask

(y) = 123451

$(AL)_i$ = 222351

Compare 2351 with 3451

$(AU)_f$ = 007777 : $(AL)_f$ = 222351

07 COMPARE WITH MASK (CMSKB) L(AU) (AL) : L(AU) (Y)

Execution time: 4 microseconds

$y = u_P$ or u_{SR}

Compare algebraically selected bits of (AL) with corresponding bits of (y + B) and set the comparison designator as follows:

Set the "COMPARE" stage

Set the "LESS THAN" stage if L(AL) (AU) < L(y+B) (AU)

Set the "EQUALS" stage if L(AL) (AU) = L(y+B) (AU)

$(AL)_f = (AL)_i : (AU)_f = (AU)_i$

NOTE: The comparison designator is cleared by the execution of any subsequent instruction other than codes 60-67, and no interrupt will be honored while the designator is SET. (Refer to paragraph entitled Conditional Jump Features.)

10 ENTER AU (ENTAU) (Y) ➡ AU

Execution time: 4 microseconds

$y = u_P$ or u_{SR}

Clear AU. Then transmit (y) to AU.

11 ENTER AU (ENTAUB) (Y) ➡ AU

Execution time: 4 microseconds

$y = u_P$ or u_{SR}

Clear AU. Then transmit (y + B) to AU.

12 ENTER AL (ENTAL) (Y) ➡ AL

Execution time: 4 microseconds

$y = u_P$ or u_{SR}

Clear AL. Then transmit (y) to AL.

13 ENTER AL (ENTALB) (Y) ➡ AL

Execution time: 4 microseconds

$y = u_P$ or u_{SR}

Clear AL. Then transmit (y + B) to AL.

14 ADD AL (ADDAL) (AL) + (Y) ➡ AL

Execution time: 4 microseconds

$y = u_P$ or u_{SR}

Add (y) to (AL) and leave the result in AL. Set OVERFLOW designator if overflow occurs.* $(AL)_f$ are all ones if $(AL)_i$ and (y) are all ones.

15 ADD AL (ADDALB) (AL) + (Y) ➡ AL

Execution time: 4 microseconds

$y = u_P$ or u_{SR}

Add (y + B) to (AL) and leave the result in AL. Set OVERFLOW designator if overflow occurs.* $(AL)_f$ are all ones if $(AL)_i$ and (y + B) are all ones.*

*The OVERFLOW designator is cleared only by the execution of instruction SKIP ON OVERFLOW (f, m = 50, 52) or instruction SKIP ON NO OVERFLOW (f, m = 50, 53).

16 SUBTRACT AL (SUBAL) (AL) - (Y) ➡ AL

Execution time: 4 microseconds

$y = u_P$ or u_{SR}

Subtract (y) from (AL) and leave the difference in AL. Set OVERFLOW designator if overflow occurs.* $(AL)_f$ are all ones if $(AL)_i$ are all ones, and (y) are all zeros.

17 SUBTRACT AL (SUBALB) (AL) - (Y) ➡ AL

Execution time: 4 microseconds

$y = u_P$ or u_{SR}

Subtract (y+B) from (AL) and leave the difference in AL. Set OVERFLOW designator if overflow occurs.* $(AL)_f$ are all ones if $(AL)_i$ are all ones and (y+B) are all zeros.

20 ADD A (ADDA) (A) + (Y+1, Y) ➡ A

Execution time: 6 microseconds

$y = u_P$ or u_{SR}

Add to (A) the double-length (36-bit) number contained in storage cells y+1, y and leave the result in A. Set OVERFLOW designator if overflow occurs.* The least significant half is in cell y, and the most significant half is in y+1. The sign of the double-length number is indicated by the most significant bit of (y+1). Address y must be even, i.e., the rightmost octal digit must be 0, 2, 4, or 6.

NOTE: The instruction is executed in the following manner:
Clear the BORROW designator. The AU and AL registers are linked to form a continuous 36-bit A-register. Therefore, any borrow for AL comes from AU; and any End Around Borrow for AU is blocked and recorded in the BORROW designator leaving A uncorrected. The "Skip On No Borrow" instruction (Code 50, 51) is used to test for required correction. Only "ADD A" or "SUBTRACT A" instructions set the designator.

Example of a double add with y = 07506

$(A)_i$ = 201007430145
address 07506 = 351123 (least significant half)
address 07507 = 077430 (most significant half)
$(A)_f$ = 300440001271 (unadjusted sum)

*The OVERFLOW designator is cleared only by the execution of instruction SKIP ON OVERFLOW (f, m = 50, 52) or instruction SKIP ON NO OVERFLOW (f, m = 50, 53).

21 ADD A (ADDAB) (A) + (Y+1, Y) ➡ A

Execution time: 6 microseconds

$y = u_P$ or u_{SR}

Add to (A) the double-length (36-bit) number contained in storage cells y+B+1, y+B, leaving the result in A. Set OVERFLOW designator if overflow occurs.* The least significant half is in cell y+B, and the most significant half is in cell y+B+1. The sign of the double-length number is the sign of (y+B+1). Address y+B must be even. (See "Note" of instruction 20.)

22 SUBTRACT A (SUBA) (A) - (Y+1, Y) ➡ A

Execution time: 6 microseconds

$y = u_P$ or u_{SR}

Subtract from (A) the double-length (36-bit) number contained in storage cells y+1,y, and leave the difference in A. Set OVERFLOW designator if overflow occurs.* The least significant half is in cell y and the most significant half is in cell y+1. The sign of the double-length number is the sign (y+1). Address y must be even. The computer executes SUBTRACT A in a manner analogous to the ADD A instruction. (See "Note" of instruction 20.)

23 SUBTRACT A (SUBAB) (A) - (Y+1, Y) ➡ A

Execution time: 6 microseconds

$y = u_P$ or u_{SR}

Subtract from (A) the double-length number contained in storage cells y + B + 1, y+B, and leave the difference in A. Set OVERFLOW designator if overflow occurs.* The least significant half is in cell y+B, and the most significant half is in cell y+B+1. The sign of the double length number is the sign of (y+B+1). Address y+B must be even. The computer executes SUBTRACT A in a manner analogous to the ADD A instruction. (See "Note" of instruction 20.)

24 MULTIPLY AL (MULAL) (AL) (Y) ➡ A

Execution time: 14 microseconds

$y = u_P$ or u_{SR}

Multiply (AL) by (y) leaving the double length product in A. If the factors are considered integers, the product is an integer in A. The multiplication process is executed on the absolute values of the factors, then corrected for algebraic sign.

*The OVERFLOW designator is cleared only by the execution of instruction SKIP ON OVERFLOW (f, m = 50, 52) or instruction SKIP ON NO OVERFLOW (f, m = 50, 53).

25 MULTIPLY AL (MULALB) (AL) (Y) ➡ A

Execution time: 14 microseconds

$y = u_P$ or u_{SR}

Multiply (AL) by (y+B) leaving the double length product in A. If the factors are considered integers, the product is an integer in A. The multiplication process is executed on the absolute value of the factors, then corrected for algebraic sign.

26 DIVIDE A (DIVA) (A)÷(Y) ➡ AL; REMAINDER ➡ AU

Execution time: 14 microseconds

$y = u_P$ or u_{SR}

Divide (A) by (y) leaving the quotient in AL and the remainder in AU. The remainder always bears the sign of the dividend "A_i" with the results satisfying the relationship: dividend = quotient x divisor + remainder. Set OVERFLOW designator if overflow occurs.* If overflow occurs, (AL) becomes 0.

Examples of the four possible sign combinations of the dividend/divisor and the results:

Dividend	Divisor	Quotient	Remainder
+5	+4	+1	+1
+5	-4	-1	+1
-5	+4	-1	-1
-5	-4	+1	-1

27 DIVIDE A (DIVAB) (A)÷(Y)➡AL; REMAINDER ➡ AU

Execution time: 14 microseconds

$y = u_P$ or u_{SR}

Divide (A) by (y+B) leaving the quotient in AL and the remainder in AU. The remainder bears the sign of the dividend "A_i". (See Instruction 26.)

30 INDIRECT RETURN JUMP (IRJP) (P)+1➡(Y); (Y)+1➡P

Execution time: 6 microseconds

Instruction executed from running program:

$y = u_P$

Store (P)+1 at the address given in the low order 16 bits of (y), then increment that address by one and enter into the Program Address register.

*The OVERFLOW designator is cleared only by the execution of instruction SKIP ON OVERFLOW (f, m = 50, 52) or instruction SKIP ON NO OVERFLOW (f, m = 50, 53).

Instruction executed from Entrance register on interrupt:

y = u

Store (P) at the address which is the low order 16 bits of (y), then increment that address by one and enter into the Program Address register.

Example of an indirect return jump executed from address 22000:

Address	Initial Contents	Final Contents	Explanation
22000	30 6500	Same	Execute subroutine from main program
26500	x1 7420	Same	Constant defining location of desired subroutine
17420	37 2164	02 2001	Subroutine exit address
17421	.. ....	Same	Subroutine entrance address (control is transferred here from indirect return jump)

The effect of the above sequence upon execution of the indirect return jump at address 22000 is to transfer control to the subroutine starting at 17421, but at the same time, letting the subroutine "know" where to return control.

31 INDIRECT RETURN JUMP (IRJPB) (P)+1➡(Y); (Y)+1➡P

Execution time: 6 microseconds

Instruction executed from running program:

$y = u_P$

Store (P)+1 at the address given in the low order 16 bits of (y+B), then increment that address by one and enter it into the Program Address register.

Instruction executed from Entrance register on interrupt:

y = u

Store (P) at the address which is the low order 16 bits of (y+B), then increment that address by one and enter it into the Program Address register.

32 ENTER B (ENTB) (Y)➡B

Execution time: 4 microseconds

$y = u_P$ or u_{SR}

Transmit (y) to B_{ICR}

The full 18 bits of (y) are transmitted to the B-register (a normally addressable core cell).

33 ENTER B (ENTBB) (Y) ➡ B

Execution time: 4 microseconds

$y = u_P$ or u_{SR}

Transmit (y+B) to B_{ICR}

The full 18 bits of (y+B) are transmitted to the B-register (a normally addressable core cell).

34 DIRECT JUMP (JP) Y ➡ P; NI = (Y)

Execution time: 2 microseconds

$y = u_P$

Unconditionally jump to y. (Reset P = y)

35 DIRECT JUMP (JPB) Y ➡ P ; NI = (Y)

Execution time: 2 microseconds

$y = u_P$

Unconditionally jump to y + B.

NOTE: Because B is an 18-bit one's complement number, care must be taken when using this instruction; in addition, it is possible that address, y+B, may not be relative to the same core bank from which the (35) DIRECT JUMP was executed. Consider a direct jump with y = 03560 and B = 010000; in this case y+B = 03560 + 010000 = 13560.

36 ENTER B WITH CONSTANT (ENTBK) xY ➡ B

Execution time: 2 microseconds

y = u (sign extended to 18 bits)

Clear B. Then transmit y to B.

NOTE: u is a 12-bit one's complement number contained within the instruction; it does not refer to an address. Example of ENTER B with constant when u = 7701:

B_i = any value

B_f = 777701

37 MODIFY B WITH CONSTANT (ENTBKB) $B_i + xY \rightarrow B$

Execution time: 2 microseconds

y = u (sign extended to 18 bits)

Add y to B (add a constant to B).

The effect of this instruction is to increment B. Note that u is a 12-bit one's complement number contained within the instruction and can be used to increment or decrement B.

40 CLEAR Y (STORE ZERO) (CL) $0 \rightarrow Y$

Execution time: 4 microseconds

$y = u_P$ or u_{SR}

Store an 18-bit word of zeros at storage address y.

41 CLEAR Y (STORE ZERO) (CLB) $0 \rightarrow Y$

Execution time: 4 microseconds

$y = u_P$ or u_{SR}

Store an 18-bit word of zeros at storage address y + B.

42 STORE B (STRB) $B \rightarrow Y$

Execution time: 4 microseconds

$y = u_P$ or u_{SR}

Store B at storage address y.

43 STORE B (STRBB) $B \rightarrow Y$

Execution time: 4 microseconds

$y = u_P$ or u_{SR}

Store B at storage address y+B.

44 STORE AL (STRAL) $(AL) \rightarrow Y$

Execution time: 4 microseconds

$y = u_P$ or u_{SR}

Store (AL) at storage address y.

$(AL)_f = (AL)_i$

45 STORE AL (STRALB) (AL) ➡ Y

Execution time: 4 microseconds

$y = u_P$ or u_{SR}

Store (AL) at storage address y+B.

$(AL)_f = (AL)_i$

46 STORE AU (STRAU) (AU) ➡ Y

Execution time: 4 microseconds

$y = u_P$ or u_{SR}

Store (AU) at storage address y.

$(AU)_f = (AU)_i$

47 STORE AU (STRAUB) (AU) ➡ Y

Execution time: 4 microseconds

$y = u_P$ or u_{SR}

Store (AU) at storage address y + B.

$(AU)_f = (AU)_i$

50 See Format II instructions immediately following function code 77.

51 SELECTIVE SET (SLSET) (AL) v (Y) ➡ AL
or SET $(AL)_n$ for $(Y)_n = 1$

Execution time: 4 microseconds

$y = u_P$

Set the individual bits of (AL) to ones corresponding to ones in (y), leaving the remaining bits of (AL) unaltered. This is a bit-by-bit inclusive OR.

Example of selective set:

$(AL)_i = 123456$

$(y) = 000077$

$(AL)_f = 123477$

52 SELECTIVE CLEAR (SLCL) L(AL) (Y) ➡ AL
or CLEAR $(AL)_n$ for $(Y)_n$=0

Execution time: 4 microseconds

$y = u_P$

Clear the individual bits of (AL) corresponding to zeros in (y), leaving the remaining bits of (AL) unaltered. The effect of this instruction is to compute the bit-by-bit (or logical) product of (AL) and (y), leaving the result in AL.

Example of selective clear:

$(AL)_i$ = 123456
(y) = 707070
$(AL)_f$ = 103050

53 SELECTIVE COMPLEMENT (SLCP) L(AL)' (Y)+ L(AL) (Y)' ➡ AL
or COMPLEMENT $(AL)_n$ for $(Y)_n$ = 1

Execution time: 4 microseconds

$y = u_P$

Complement the individual bits of (AL) corresponding to ones in (y), leaving the remaining bits of (AL) unaltered; i.e., complement $(AL)_n$ for $(y)_n$ = 1. This is a bit-by-bit exclusive OR.

Example of selective complement instruction:

$(AL)_i$ = 123456
(y) = 070007
$(AL)_f$ = 153451

54 INDIRECT JUMP AND REMOVE INTERRUPT LOCKOUT (IJPEI) (Y) ➡ P and RIL

Execution time: 4 microseconds

$y = u_P$ Address = $(y)_{15-0}$

Remove interrupt lockout (enable interrupts). Then jump to the address which is the low order 16 bits of (y). An application of this instruction is the termination of a subroutine activated by an interrupt.

55 INDIRECT JUMP (IJP) Y ➡ P

Execution time: 4 microseconds

$y = u_P$ Address = $(y)_{15-0}$

Jump to the address which is the low order 16 bits of (y).

56 B SKIP (BSK) If B = (Y), SKIP NI
If B ≠ (Y), Advance B by 1 and Execute NI

Execution time: 4 microseconds

$y = u_P$

Test B and (y) for equality. Skip instruction if equal; otherwise, increment B by 1 and read the next instruction.

57 INDEX SKIP (ISK) If (Y) = 0, SKIP NI
If (Y) ≠ 0, Decrement (Y) by 1 and Execute NI

If $(y)_i$ = 777777, then
$(y)_f$ = 777776 and there is no skip.

Execution time: 6 microseconds

$y = u_P$

If (y) ≠ 0, subtract one from (y) leaving the result in y, and take the next instruction; otherwise, skip the next instruction leaving (y) unaltered.

60 JUMP AU ZERO (JPAUZ) If (AU) = 0, ⊕ (AL) = M, ⊕ L(AL) (AU) = M, Y ➡ P

Execution time: 2 microseconds

$y = u_P$

JUMP to y, i.e., Reset P = y, if:

"COMPARE" stage of the comparison designator is NOT SET and (AU) = 0 (Negative ZERO acts as NOT ZERO); or

"COMPARE" stage of the comparison designator is SET, and the "EQUALS" stage of the comparison designator is SET.

Otherwise, execute next instruction.

NOTE: Refer to paragraph entitled Conditional Jump Features.

61 JUMP AL ZERO (JPALZ) If (AL) = 0, ⊕
(JPEQ) If (AL) = M ⊕ L(AL) (AU) = M, Y ➡ P

Execution time: 2 microseconds

$y = u_P$

JUMP to y, i.e., Reset P = y if:

"COMPARE" stage of the comparison designator is NOT SET and (AL) = 0 (Negative ZERO acts as NOT ZERO), or if "COMPARE" stage of the comparison designator is SET, and the "EQUALS" stage of the comparison designator is SET.

Otherwise, execute next instruction.

NOTE: Refer to paragraph entitled Conditional Jump Features.

62 JUMP AU NOT ZERO (JPAUNZ) If (AU) $\neq$ 0 $\oplus$
If (AL) $\neq$ M $\oplus$ L(AL) (AU) $\neq$ M, Y ➡ P

Execution time: 2 microseconds

y = u_p

JUMP to y, i.e., Reset P = y, if:

"COMPARE" stage of comparison designator is NOT SET and (AU) $\neq$ 0; or

"COMPARE" stage of comparison designator is SET, and the "EQUALS" stage of the comparison designator is NOT SET.

Otherwise, execute next instruction.

NOTE: Refer to paragraph entitled Conditional Jump Features.

63 JUMP AL NOT ZERO (JPALNZ) (JPALZ) if (AL) $\neq$ 0
(JPNOT) $\oplus$ If (AL) $\neq$ M $\oplus$ L(AL)(AU) $\neq$ M, Y ➡ P

Execution time: 2 microseconds

y = u_p

JUMP to y, i.e., Reset P = y, if:

"COMPARE" stage of comparison designator is NOT SET and (AL) $\neq$ 0; or

"COMPARE" stage of comparison designator is SET, and the "EQUALS" stage of the comparison designator is NOT SET.

Otherwise, execute next instruction.

NOTE: Refer to paragraph entitled Conditional Jump Features.

64 JUMP AU POSITIVE (JPAUP) If (AU) POS, $\oplus$
If (AL) $\geq$ M $\oplus$ L(AL) (AU) $\geq$ M, Y ➡ P

Execution time: 2 microseconds

y = u_p

JUMP to y, i.e., Reset P = y, if:

"COMPARE" stage of comparison designator is NOT SET and (AU) $\geq$ 0; or

"COMPARE" stage of comparison designator is SET, and the "LESS THAN" stage of comparison is NOT SET.

Otherwise, execute next instruction.

NOTE: Refer to paragraph entitled Conditional Jump Features.

65 JUMP AL POSITIVE (JPALP) If (AL) POS,
(JPMLEQ) ⊕ If (AL) ≥ M ⊕ L(AL) (AU) ≥ M, Y ➡ P

Execution time: 2 microseconds

y : u_P

JUMP to y, i.e., Reset P = y, if:

"COMPARE" stage of comparison designator is NOT SET and (AL) ≥ 0; or

"COMPARE" stage of comparison designator is SET, and the "LESS THAN" stage of comparison designator is NOT SET.

Otherwise, execute next instruction.

NOTE: Refer to paragraph entitled Conditional Jump Features.

66 JUMP AU NEGATIVE (JPAUNG) If (AU) NEG,
⊕ If (AL) < M ⊕ L(AL) (AU) < M, Y ➡ P

Execution time: 2 microseconds

y = u_P

JUMP to y, i.e., Reset P = y, if:

"COMPARE" stage of comparison designator is NOT SET and (AU) < 0; or

"COMPARE" stage of comparison designator is SET, and the "LESS THAN" stage of comparison designator is SET.

Otherwise, execute next instruction.

NOTE: Refer to paragraph entitled Conditional Jump Features.

67 JUMP AL NEGATIVE (JPALNG) If (AL) NEG,
(JPMGR) ⊕ If (AL) < M⊕L(AL) (AU) < M, Y➡P

Execution time: 2 microseconds

y = u_P

JUMP to y, i.e., Reset P = y, if:

"COMPARE" stage of comparison designator is NOT SET and (AL) < 0; or

"COMPARE" stage of comparison designator is SET, and the "LESS THAN" stage of comparison designator is SET.

Otherwise, execute next instruction.

NOTE: Refer to paragraph entitled Conditional Jump Features.

70 ENTER AL WITH CONSTANT (ENTALK) xY ➡ AL

Execution time: 2 microseconds

y = u (with sign extended to 18 bits).

Clear AL. Then transmit y to AL.

Example of enter AL with constant when u = 0001

$(AL)_i$ = any value

$(AL)_f$ = 000001 (+1)

Example of enter AL with constant when u = 7776

$(AL)_i$ = any value

$(AL)_f$ = 777776 (-1)

NOTE: u is a 12-bit one's complement number contained within the instruction; it does not refer to an address.

71 ADD CONSTANT TO AL (ADDALK) (AL) + xY ➡ AL

Execution time: 2 microseconds

y = u (sign extended to 18 bits)

Add y to (AL) and leave the result in AL. The effect of this instruction is to increment/decrement (AL) with a constant contained within the instruction.

Example of add constant to AL when u = 0002 (+2)

$(AL)_i$ = 057777

$(AL)_f$ = 060001 (incremented)

Example of add constant to AL when u = 7775 (-2)

$(AL)_i$ = 067055

$(AL)_f$ = 067053 (decremented)

72 STORE INDEX CONTROL REGISTER (STRICR) (ICR) ➡ Y_{5-0}

Execution time: 4 microseconds

$y = u_P$

Replace the least significant 6 bits of the (y) with a 6-bit value equal to the memory address of the Index Register defined by ICR. As this instruction effects a 6-bit partial transfer, the upper 12 bits of (y) remain unchanged.

NOTE: ICR = 0, produces memory address 10. ICR = 1 through 7, memory addresses 01 through 07 respectively.

73 B JUMP (BJP) If B $\neq$ 0, B-1 ➡ B and Y ➡ P
If B = 0, Execute NI

Execution time: 2 microseconds

$y = u_p$

If B $\neq$ 0, subtract one from B then jump to y; otherwise, execute the next instruction leaving B unaltered. (Negative Zero $\neq$ 0.)

NOTE: As B is a one's complement number and can take values less than zero, the B JUMP will be effective only for program loops where B is initially positive.

74 STORE ADDRESS(STRADR) $(AL)_{11-0}$ ➡ Y_{11-0}

Execution time: 4 microseconds

$y = u_p$

Replace the low order 12 bits of (y) with the low order 12 bits of (AL). As this instruction effects a partial transfer, the higher order six bits of (y) remain undisturbed.

$(AL)_f = (AL)_i$

Example of a store address instruction:

$(AL)_i = 762504$

$(y)_i = 567777$

$(y)_f = 562504$

75 STORE SPECIAL REGISTER (STRSR) (SR) ➡ Y_{5-0}

Execution time: 4 microseconds

$y = u_p$

Replace the low order six bits of(y) with a 6-bit value of which the low order 5 bits are equal to the contents of the Special register with the remaining bit equal to zero; store the result at y, then clear the Special register. As this instruction effects a 6-bit partial transfer, the upper 12 bits of (y) remain undisturbed.

NOTE: This instruction deactivates the Special register.

76 DIRECT RETURN JUMP (RJP) (P) + 1 ➡ Y; Y + 1 ➡ P

Execution time: 4 microseconds

$y = u_p$

Store (P) + 1 at y, then jump to y + 1. This instruction transfers to y a full 18-bit word, the lower 16 bits being the address (P) + 1 with the upper two bits set to zero. When this instruction is executed from an Interrupt Entrance register by an Interrupt, store (P). Do not initiate the "(P) + 1 sequence".

77 FAULT CODE; JUMP TO FAULT ENTRANCE REGISTER

Execution time: 2 microseconds

FORMAT II INSTRUCTIONS

The following are Format II, Type 3 instructions and require a combination of a 50-function code and a subfunction code that determines the operation to be performed. The 50-function code is detected when read from memory and causes $1 \rightarrow F_6$ and Z_{11-6} F_{5-0} for execution. The computer maintains its regular timing sequence.

50 00 FAULT CODE; JUMP TO FAULT ENTRANCE REGISTER

Execution time: 2 microseconds

50 01 SET INPUT ACTIVE (SIN)

Execution time: 2 microseconds

Set input channel k to the active state. The buffer control words stored in memory locations 60 + 2k and 61 + 2k or as specified by the Externally Specified Index or Externally Specified Address will control the transfers.

50 02 SET OUTPUT ACTIVE (SOUT)

Execution time: 2 microseconds

Set output channel k to the active state. The buffer control words stored in memory locations 40 + 2k and 41 + 2k or as specified by the ESI or ESA will control the transfers.

50 03 SET EXTERNAL FUNCTION ACTIVE (SEXF)

Execution time: 2 microseconds

Set channel k external function mode active. The buffer control words stored in memory locations 40 + 2k and 41 + 2k will control the transfers.

50 04 Not Used

50 05 Not Used

50 06 Not Used

50 07 Not Used

50 10 Not Used

50 11 INPUT TRANSFER (IN)

(P+1) ➡ 60+2k
(P+2) ➡ 61+2k
SET INPUT ACTIVE ON CHAN. k

Execution time: 6 microseconds

Initiate Input Transfer on channel k.

Transfer buffer limit address words (for input buffer) from the following two instruction locations to the Input Buffer Control registers for the designated channel. Other I/O channel and processor activity proceeds normally.

50 12 OUTPUT TRANSFER (OUT)

(P+1) ➡ 40+2k
(P+2) ➡ 41+2k
SET OUTPUT ACTIVE ON CHAN. k

Execution time: 20 microseconds

Initiate Output Transfer on channel k.

Transfer buffer limit address words (for output buffer) from the following two instruction locations to the Output Buffer Control registers for the designated channel. Other I/O channel and processor activity proceeds normally.

50 13 EXTERNAL FUNCTION (EXF)

(P+1) ➡ 40+2k
(P+2) ➡ 41-2k
SET EXTERNAL FUNCTION ON CHAN. k

Execution time: 20 microseconds

Initiate External Function Mode on channel k.

Transfer buffer limit addresses (for the function buffer) from the following two instruction locations to the Output Buffer Control registers for the designated channel.

50 14 ENABLE REAL-TIME CLOCK MONITOR (RTC)

Execution time: 2 microseconds

Enable the Real-Time Clock Monitor Interrupt. Ignore k. After execution of this instruction, equality between the RTC register (location 15) and the RTC Monitor Word register (location 14) will interrupt the computer program. The next instruction is taken from the RTC Monitor Interrupt Entrance register (location 12), and the RTC Monitor is disabled.

50 15 TERMINATE INPUT (INSTP)

CLEAR INPUT ACTIVE CHANNEL k

Execution time: 2 microseconds

Terminate Input on channel k.

No Monitor interrupt will occur as a result of the execution of this instruction.

50 16 TERMINATE OUTPUT (OUTSTP) CLEAR OUTPUT ACTIVE CHAN. k

Execution time: 2 microseconds

Terminate Output or External Function Mode on channel k.

No Monitor interrupt will occur as a result of the execution of this instruction.

50 17 TERMINATE EXTERNAL FUNCTION (EXFSTP)

CLEAR EXTERNAL FUNCTION ACTIVE CHAN. k

Execution time: 2 microseconds

Terminate External Function Mode or Output on channel k.

No Monitor interrupt will occur as a result of the execution of this instruction.

50 20 SET RESUME (SRSM)

Execution time: 2 microseconds

Set the "RESUME" designator for channel k group to permit honoring the next requesting output function on that group. Loss of any information currently held by the output register(s) for a peripheral device is allowed by this instruction.

50 21 SKIP ON INPUT INACTIVE (SKPIIN)

Execution time: 2 microseconds skip or no skip

Test for input activity on channel k. If inactive, skip the next instruction; otherwise, take the next instruction.

50 22 SKIP ON OUTPUT INACTIVE (SKPOIN)

Execution time: 2 microseconds skip or no skip

Test for output activity on channel k. If inactive, skip the next instruction; otherwise, take the next instruction. This instruction tests the output active flip-flop.

50 23 SKIP ON EXTERNAL FUNCTION INACTIVE (SKPFIN)

Execution time: 2 microseconds skip or no skip

Test for External Function activity on channel k. If inactive, skip the next instruction; otherwise, take the next instruction. This instruction tests the External Function Mode flip-flop.

50 24 WAIT FOR INTERRUPT (WTFI)
or
50 25 Execution time: 2 microseconds

Stop the computer until any interrupt occurs and allow I/O to continue; ignore k, then execute the instruction located in the Interrupt Entrance register designated by the interrupt.

50 26 OUTPUT OVERRIDE (OUTOV)

Execution time: 2 microseconds

Wait for the output device to accept the word in the C-register(s). Then simulate an Output Request on channel k and transfer the word designated by the address in the Output Buffer Control register for that channel. Ignore the ESI Mode if active. This instruction will transfer a word whether the buffer is active or not. The transfer takes place under control of the word in the Buffer Control register.

50 27 EXTERNAL FUNCTION OVERRIDE (EXFOV)

Execution time: 2 microseconds

Wait for the output device to accept the word in the C-register(s). Then simulate an External Function Request on channel k and transfer the word designated by the address in the Output Buffer Control register for that channel. Ignore the ESI Mode if active. This instruction will transfer a word whether the buffer is active or not. The transfer takes place under control of the word in the Buffer Control register.

50 30 REMOVE INTERRUPT LOCKOUT (RIL)
or
50 31 Execution time: 2 microseconds

Remove the interrupt lockout; enable all external and monitor interrupts, all channels. Ignore k. This instruction complements SIL.

50 32 REMOVE EXTERNAL INTERRUPT LOCKOUT (RXL)
or
50 33 Execution time: 2 microseconds

Enable external interrupts, all channels. Ignore k. This instruction complements SXL.

50 34 SET INTERRUPT LOCKOUT (SIL)
or
50 35 Execution time: 2 microseconds

Set the interrupt lockout; disable all external and monitor interrupts, all channels. Ignore k.

50 36 SET EXTERNAL INTERRUPT LOCKOUT (SXL)
or
50 37 Execution time: 2 microseconds

Disable external interrupts, all channels. Ignore k.

50 40 Not Used

50 41 RIGHT SHIFT AU (RSHAU)

Execution time: 2 microseconds (k=0); 4 to 12 microseconds (k≠0)

Shift (AU) to the right k-bit positions. The higher order bits are replaced with the original sign bit, AU_{17}, as the value is shifted. This is an end-off shift (i.e., the low order bits are "lost" upon completion of the shift).

Example of right shift AU with k = 2

$(AU)_i$ (positive)	= 370000
after first shift	174000
after second shift	076000
$(AU)_i$ (negative)	= 400000
after first shift	600000
after second shift	700000

NOTE: Execution time for all shift instructions:

if k equals (octal)	microseconds
0	2
1-4	2+2
5-10	2+4
11-14	2+6
15-20	2+8
21-24	2+10
25-30	2+12
31-34	2+14
35-40	2+16
41-44	2+18

50 42 RIGHT SHIFT AL (RSHAL)

Execution time: 2 microseconds (k=0); 4 to 12 microseconds (k≠0)

Shift (AL) to the right k-bit positions. The higher order bits are replaced with the original sign bit, AL_{17}, as the value is shifted. This is an end-off shift (i.e., the low order bits are "lost" upon completion of the shift).

50 43 RIGHT SHIFT A (RSHA)

Execution time: 2 microseconds (k=0); 4 to 20 microseconds (k≠0)

Shift (A) to the right k-bit positions. The higher order bits are replaced with the original sign bit, A_{35}, as the value is shifted. This is an end-off shift (i.e., the low order bits are "lost" upon completion of the shift).

Example of right shift A with k = 2:

$(A)_i$ (positive)	= 370000 000000
after first shift	174000 000000
after second shift	076000 000000
$(A)_i$ (negative)	= 400000 000000
after first shift	600000 000000
after second shift	700000 000000

50 44 SCALE FACTOR (SF)

Execution time: 4 microseconds (k = 0); 4 to 20 (k ≠ 0)

Shift (A) circularly to the left until either $A_{35} \neq A_{34}$ or k minus shift count = 0; then store the positive quantity k minus shift count at memory address 00017. The effect of the instruction is to "normalize" (A) to the left subject to k. SCALE FACTOR is extremely useful when working with numerical values in floating point notation.

(a) Example of scale factor with k = 7:

$(A)_i$ = 170000 000000 (positive, not normalized) after first shift 360000 000000 (positive, normalized)

The computer, sensing (A) nor normalized, stores k-shift count (7-1) = the 18-bit quantity "000006" ➡ 00017.

(b) Example of scale factor with k = 3:

$(A)_i$ = 600000 000000 (negative, not normalized) after first shift 400000 000001 (negative, normalized)

The computer then stores the quantity "000002" ➡ 00017.

(c) Example of scale factor with k = 1:

$(A)_i$ = 070000 000000 (positive, not normalized) after first shift 160000 000000 positive, not normalized)

The computer, having exhausted k, stores the quantity "000000"➡00017 leaving (A) only partially normalized.

50 45 LEFT SHIFT AU (LSHAU)

Execution time: 2 microseconds (k = 0); to 12 microseconds (k ≠ 0)

Shift (AU) circularly to the left k-bit positions. The lower order bits are replaced with the higher order bits as the word is shifted.

Example of left shift AU with k = 2:

$(AU)_i$	= 300000
after first shift	600000
after second shift	400001

No bits are "lost" with the execution of left shift instructions.

50 46 LEFT SHIFT AL (LSHAL)

Execution time: 2 microseconds (k = 0); 4 to 12 microseconds (k ≠ 0)

Shift (AL) circularly to the left k-bit positions. The lower order bits are replaced with the higher order bits as the word is shifted. No bits are "lost" with the execution of left shift instructions.

50 47 LEFT SHIFT A (LSHA)

Execution time: 2 microseconds (k = 0); 4 to 20 microseconds (k ≠ 0)

Shift (A) circularly to the left k-bit positions. The lower order bits are replaced with the higher order bits as the word is shifted. No bits are "lost" with the execution of left shift instructions.

Example of left shift A with k = 2:

$(A)_i$	= 300000 000000
after first shift	600000 000000
after second shift	400000 000001

50 50 SKIP ON KEY SETTING (SKP)

Execution time: 2 microseconds skip or no skip

If bit 4,3,2,1 or 0 of k is one and the corresponding SKIP KEY 4, 3, 2, 1, or 0 is set... or ... if bit 5 of k is a one (unconditional skip) ... skip the next instruction. Otherwise, take the next instruction.

Examples of skip with:

k = 01 (bit 0)	skip if skip key 0 is set
k = 02 (bit 1)	skip if skip key 1 is set
k = 04 (bit 2)	skip if skip key 2 is set
k = 10 (bit 3)	skip if skip key 3 is set
k = 20 (bit 4)	skip if skip key 4 is set
k = 40 (bit 5)	skip unconditionally
k = 03 (bits 1, 0)	skip if either key 1 or 0 is set

50 51 SKIP ON NO BORROW (SKPNBO)

Execution time: 2 microseconds skip or no skip

If the last previous ADD A or SUBTRACT A required a borrow, take next instruction; otherwise, skip the next instruction. Ignore k. The skip occurs if no correction to (A) is needed. This allows a correcting instruction to be inserted to save program steps. The correcting instruction will be "SUBTRACT A" where (Y + 1, Y) = 000000000001.

50 52 SKIP ON OVERFLOW (SKPOV)

Execution time: 2 microseconds skip or no skip

If an overflow condition occurred on a previous arithmetic instruction, skip the next instruction; otherwise, take the next instruction. Ignore k and clear the OVERFLOW designator.

50 53 SKIP ON NO OVERFLOW (SKPNOV)

Execution time: 2 microseconds skip or no skip

If an overflow condition did not occur on a previous arithmetic instruction, skip the next instruction; otherwise, take the next instruction. Ignore k and clear the OVERFLOW designator.

50 54 SKIP ON ODD PARITY (SKPODD)

Execution time: 2 microseconds skip or no skip

If the sum of the bits resulting from the bit-by-bit product of (AL) and (AU) is odd, skip the next instruction; otherwise, take the next instruction. Ignore k.

$(AU)_f = (AU)_i$; $(AL)_f = (AL)_i$

Example of skip odd parity:

(AU)	000077	mask
(AL)	127723	
bit-by-bit product	= 000023	
bit sum	= 3	

Since the bit sum is odd, the next instruction is skipped.

50 55 SKIP ON EVEN PARITY (SKPEVN)

Execution time: 2 microseconds skip or no skip

If the sum of the bits resulting from the bit-by-bit product of (AL) and (AU) is even, skip the next instruction; otherwise, take the next instruction. Ignore k.

$(AL)_f = (AL)_i$; $(AU)_f = (AU)_i$

50 56 STOP ON KEY SETTING (STOP)

Execution time: 2 microseconds

If bit 4, 3, 2, 1, or 0 of this one and the corresponding console STOP KEY 4, 3, 2, 1, or 0 is SET ... or ... if bit 5 of k is a one (unconditional stop) ... stop the computer; otherwise, take the next instruction.

Examples of stop with:

k = 01 (bit 0)	stop if stop key 0 is set
k = 02 (bit 1)	stop if stop key 1 is set
k = 04 (bit 2)	stop if stop key 2 is set
k = 10 (bit 3)	stop if stop key 3 is set
k = 20 (bit 4)	stop if stop key 4 is set
k = 40 (bit 5)	stop unconditionally
k = 03 (bits 1, 0)	stop if either stop key 1 or 0 is set

50 57 SKIP ON NO RESUME (SKPNR)

Execution time: 2 microseconds skip or no skip

If the "RESUME" designator on channel k is not SET (indicating unsuccessful transfer of a word to an output device), skip the next sequential instruction; otherwise, take the next instruction.

50 60 ROUND AU (RND) If (AU) pos., (AU) + AL_{17} ➡ AL

Execution time: 2 microseconds If (AU) neg., (AU) - $\overline{AL}_{17}$ ➡ AL

If (AU) is positive, add bit position 17 of AL to (AU); if (AU) is negative, subtract the complement of bit position 17 of AL from AU and leave the resultant rounded (AU) in AL. Ignore k. $(AU)_i = (AU)_f$. An application of this instruction would be: a double length value in A is normalized as far as possible to the left; however, only a rounded single length number is required for the accuracy desired.

50 61 COMPLEMENT AL (CPAL) (AL) ➡ AL

Execution time: 2 microseconds

Complement (AL), leaving the result in AL. Ignore k.

NOTE: This instruction effects a bit-by-bit complement with the following exception: all "zeros" (positive zero) will remain all "zeros".

50 62 COMPLEMENT AU (CPAU) (AU)' ➡ AU

Execution time: 2 microseconds

Complement (AU), leaving the result in AU. Ignore k. (See Note: Instruction 50 61.)

50 63 COMPLEMENT A (CPA) (A)' ➡ A

Execution time: 2 microseconds

Complement (A) leaving the result in A. Ignore k. (See Note: Instruction 50 61.)

50 64 Not Used

50 65 Not Used

50 66 Not Used

50 67 Not Used

50 70 Not Used

50 71 Not Used

50 72 ENTER INDEX CONTROL REGISTER (ENTICR) $k_{2-0} \Rightarrow$ ICR

Execution time: 2 microseconds

Clear the Index Control register. Then transmit the three low order bits of k to the ICR.

50 73 ENTER SPECIAL REGISTER (ENTSR) $k_{4-0} \Rightarrow$ SR

Execution time: 2 microseconds

Clear the Special register. Then transmit the five low order bits of k to the SR. ($SR_3 = 1$ activates the SR.)

50 74 Not Used

50 75 Not Used

50 76 Not Used

50 77 FAULT CODE - 2 microseconds

CONDITIONAL JUMP FEATURES

The Arithmetic Conditional Jump instructions may be used with associated Compare instructions to obtain certain results according to the state of the Comparison Designator or may be used independently with other results.

The Comparison Designator is a 3-stage bistable register which records the results of a Compare instruction (02, 03, 06, and 07) as follows:

- The "COMPARE" stage is SET upon the computer's execution of any one of the Compare instructions;
- The "LESS THAN" stage is SET if a Compare instruction finds (AL) less than the contents of an addressed memory location, or L(AL) (AU) less than the logical product of (AU) and the contents of the addressed memory location (whichever applies);
- The "EQUALS" stage is SET if a Compare instruction finds (AL) equal to the contents of an addressed memory location or finds the logical product of (AL) and (AU) equal to the logical product of (AU) and the contents of the addressed memory location (whichever applies).

The Comparison Designator is cleared by the execution of any instruction other than the Arithmetic Conditional Jump instructions (codes 60-67). Therefore, in order to set the compare stages desired, a Compare instruction must immediately precede a single 60-67 instruction,

or immediately precede the first of a consecutive string of 60-67 instructions. Otherwise, these jump instructions are executed without reference to the Comparison Designator.

The Arithmetic Conditional Jump instructions 60-67 are used with or without an associated Compare instruction (02, 03, 06, and 07). If used without a preceding Compare instruction, the JUMP is executed upon satisfying the condition directly stated by the instruction. If a Compare instruction is used in conjunction with one or more Conditional Jump instructions, the satisfaction of a Jump condition is dependent on the SET or NOT SET state of certain stages of the Comparison Designator.

The accompanying tabulation shows the Jump or No Jump conditions resulting from the combined and separate uses of the Compare and Arithmetic Conditional Jump instructions. The Compare instructions use the following operands for comparison with (AL): (y) for 02; (y+B) for 03; for comparison with L(AL) (AU); L(y) (AU) for 06; and L(y+B)(AU) for 07.

Therefore, "M" in the tabulation will represent (y), (y+B), L(y) (AU), or L(y+B) (AU) whichever applies. The NO JP condition will cause the computer to execute the next sequential instruction. (AL) always will be masked by (AU) by instruction 06 or 07; i.e., selected bits of (AL) will be compared with the corresponding bits of (y) or (y+B).

JUMP OR NO JUMP CONDITIONS

(AL) is masked by (AU) if instructions "06" or "07" were used.

JUMP INSTR CODE	COMPARE DESIGNATOR NOT SET	COMPARE DESIGNATOR SET				RESULTS IF A JUMP OCCURS
		EQUALS STAGE		LESS THAN STAGE		
		SET	NOT SET	NOT SET	SET	
60	JP if (AU) = 0	JP if (AL) = M	No JP	*	*	(AU) = 0 or (AL) = M
61	JP if (AL) = 0	JP if (AL) = M	No JP	*	*	(AL) = 0 or M
62	JP if (AU) ≠ 0	No JP	JP if (AL) ≠ M	*	*	(AU) ≠ 0 or (AL) ≠ M
63	JP if (AL) ≠ 0	No JP	JP if (AL) ≠ M	*	*	(AL) ≠ 0 or M
64	JP if (AU) ≧ 0	*	*	JP if (AL) ≧ M	No JP	(AU) POS. or (AL) ≧ M
65	JP if (AL) ≧ 0	*	*	JP if (AL) ≧ M	No JP	(AL) POS. or (AL) ≧ M
66	JP if (AU) < 0	*	*	No JP	JP if (AL) < M	(AU) NEG. or (AL) < M
67	JP if (AL) < 0	*	*	No JP	JP if (AL) < M	(AL) NEG. or (AL) < M

*Does not apply, and the next sequential instruction is executed.

Examples in the uses of Compare and Arithmetic Conditional Jump instructions follow.

Problem statement:

Test for a positive value in AL which would be less than M (as defined for the table).

If found, jump to "LETS".

Algorithm: $0 \leq (AL) < M$

P = 3XXXX ; Y = u_p for function codes 60-67.

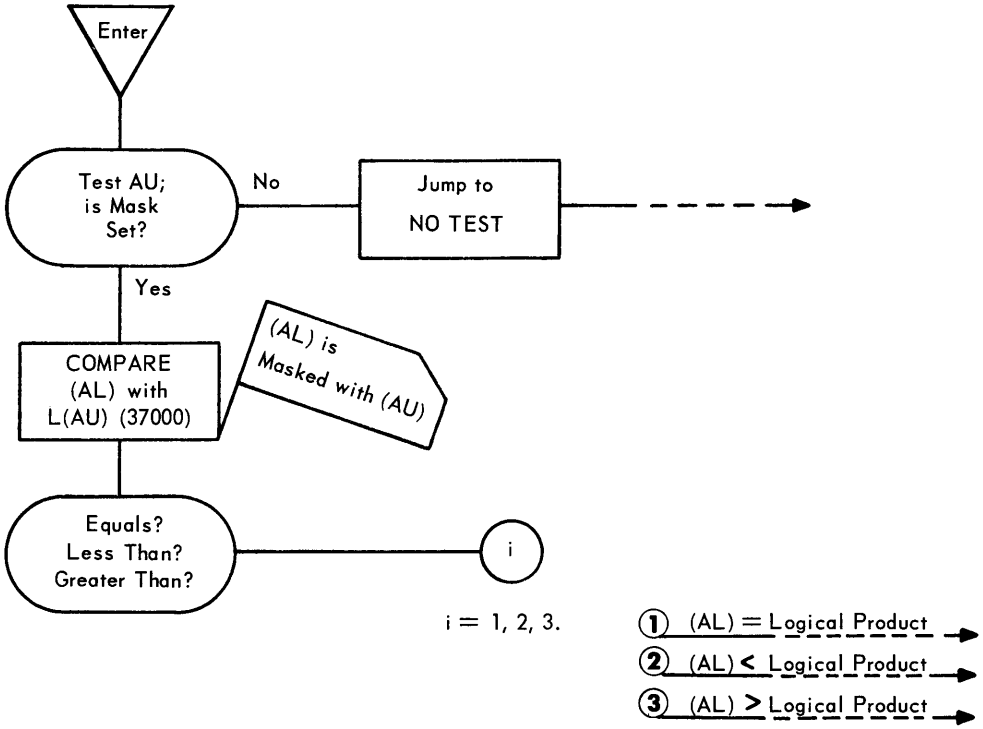
(a) The following routine results in a Jump if condition is true.

Address		Instruction
31000	Some Arithmetic instruction	
	65 4562	Test for (AL) ≥ 0 Jump to 34562 if (AL) is positive.
	.	
	.	
	.	
	.	
34562	02 7000	Set "COMPARE" stage, also SET "LESS THAN" stage if (AL) < (37000).
	67 7500	JP to 37500 if "LESS THAN" stage is set.

(b) The following routine results in sequential execution of instruction if condition is true.

Address		Instruction
32000	Some Arithmetic instruction	
	67 5000	Jump if (AL) is negative.
	02 7000	SET "LESS THAN" stage if (AL) < (37000).
	65 5000	Jump if "LESS THAN" stage is NOT SET.
		IF SET — execute next instruction.
	Next Instruction.	

(c) The following routine shows the use of more than one consecutive arithmetic conditional jump instruction following a Compare instruction. The routine also requires a mask in AU.



INSTR Address	f	y	
33000	60	4000	if (AU) = 0, JP to 4000 Extended to 34000
	06	7000	Test L (AL) (AU) $\leq$ L (AU) (37000) Set Designator
	60	4100	Jump to 34100 if "EQUALS" stage set
	67	4200	Jump to 34200 if "LESS THAN" stage set
	65	4300	Jump to 34300 if "LESS THAN" stage is NOT SET

PROGRAMMING CONSIDERATIONS

1. Bank Overflow — TRIM makes no note of this except forces zeros if polycode generation overflows bank.
2. Avoid using last cell of bank.
3. IRJP, IJP — Only means of jumping from bank to bank (keep bank independence in mind).
4. SR Activity — Keep in mind SR sensitive instructions.
5. B Registers — 1 through 7 are at addresses 00001 through 00007 respectively. B Register 0 is at address 00010.
6. STRICR • O — Store 10.
7. ENTBK, ENTALK, ADDALK — Sign extended.
8. B Register can be used to pick Operand from another bank or store an Operand in another bank.
9. Put IRJP to available address in the Interrupt Entrance address location.
10. ADDA, SUBA, ADD + 0 to +0 = + 1, SUBTRACT - 0 from + 0 = + 1.
11. After Compare — Clear designator before doing JPALZ or similar instruction.
12. RTC Overflow FF is not cleared unless checked.
13. Double length Add or Subtract Operand must be at an even numbered address.
14. The 5030K (Remove Interrupt Lockout) instruction enables all interrupts (except those previously locked out by executions of the 5034K (Set Ext. Int. Lockout) instruction.
15. 5026 and 5027 have the same meaning — will force one word out and depending on which mode is active it will send either External Function or Output Acknowledge.
16. 5016 and 5017 have the same meaning — terminates whichever mode is currently active.
17. External Function and CDM use the same Buffer Locations; exercise caution when programming with CDM.
18. Intercomputer time out interrupt will not release channel. This must be done by appropriate Terminate Instruction.
19. 5057 skip on no resume, the k portion of the instruction specifies the channel group.
20. For Output — If TACW = IACW a 1 word transfer will occur.

UNIVAC 1219B Computer Repertoire of Instructions

ULTRA SYMBOL	CODE	TRIM SYMBOL	DESCRIPTION	TIME μS
CL	02	CMAL	Compare Y	4
	03	CMALB	Compare Y+B	4
MSL	04	SLSU	Selective Substitute	4
	05	SLSUB	Selective Substitute Y+B	4
CLM	06	CMSK	Masked Compare Y	4
	07	CMSKB	Masked Compare Y+B	4
LU	10	ENTAU	Enter AU, Y	4
	11	ENTAUB	Enter AU, Y+B	4
LL	12	ENTAL	Enter AL, Y	4
	13	ENTALB	Enter AL, Y+B	4
AL	14	ADDAL	Add Y, 18 bit	4
	15	ADDALB	Add Y+B, 18 bit	4
ANL	16	SUBAL	Subtract Y, 18 bit	4
	17	SUBALB	Subtract Y+B, 18 bit	4
AA	20	ADDA	Add Y, 36 bit	6
	21	ADDAB	Add Y+B, 36 bit	6
ANA	22	SUBA	Subtract Y, 36 bit	6
	23	SUBAB	Subtract Y+B, 36 bit	6
M	24	MULAL	Multiply Y	14
	25	MULALB	Multiply Y+B	14
D	26	DIVA	Divide, Y	14
	27	DIVAB	Divide,Y+B	14
SLJI	30	IRJP	Indirect RJP, Y	6
	31	IRJPB	Indirect RJP, Y+B	6
LB	32	ENTB	Enters B, Y	4
	33	ENTBB	Enter B, Y+B	4
J	34	JP	Jump,Y	2
	35	JPB	Jump,Y+B	2
LBK	36	ENTBK	Enter, B, U	2
	37	ENTBKB	Modify B, U	2
CY	40	CL	Store Zero, Y	4
	41	CLB	Store Zero, Y+B	4
SB	42	STRB	Store, B, Y	4
	43	STRBB	Store B, Y+B	4
SL	44	STRAL	Store AL, Y	4
	45	STRALB	Store AL, Y+B	4
SU	46	STRAU	Store AU, Y	4
	47	STRAUB	Store AU, Y+B	4
OR	51	SLSET	Selective Set (IOR), Y	4
AND	52	SLCL	Selective Clear (AND), Y	4
XOR	53	SLCP	Selective Complement (XOR), Y	4
EJI	54	IJPEI	Indirect Jump (RIL), Y	4
JI	55	IJP	Indirect Jump, Y	4
TB	56	BSK	Increment B, Skip, Y	4
TZ	57	ISK	Decrement Index, Skip, Y	6
JE* JUZ	60	JPAUZ	Jump AU Zero, Y	2
JLZ	61	JPALZ	Jump AL Zero, Y	2
JNE* JUNZ	62	JPAUNZ	Jump AU Not Zero, Y	2
JLNZ	63	JPALNZ	Jump AL Not Zero, Y	2
JNLS* JUP	64	JPAUP	Jump AU Positive, Y	2
JLP	65	JPALP	Jump AL Positive, Y	2

ULTRA SYMBOL	CODE	TRIM SYMBOL	DESCRIPTION	TIME μS
JUN JLS*	66	JPAUNG	Jump Au Negative, Y	2
JLN	67	JPALNG	Jump AL Negative, Y	2
LLK	70	ENTALK	Enter AL, Y	2
ALK	71	ADDALK	Add U, 12 bits	2
SIR	72	STRICR	Store ICR, Y	4
JBNZ	73	BJP	Decrement B, Jump, Y	2
SAD	74	STRADR	Store Address, Y	4
SSR	75	STRSR	Store SR, Deactivate SR, Y	4
SLJ	76	RJP	Return Jump, Y	4
SIC	5001	SIN	Set Input Active	2
SOC	5002	SOUT	Set Output Active	2
SFC	5003	SEXF	Set External Function Active	2
LIC	5011	IN	Initiate Input Buff, k	6
LOC	5012	OUT	Initiate Output Buff, k	6
LFC	5013	EXF	External Function	6
ACI	5014	RTC	Enable Real-Time Clock	2
STIC	5015	INSTP	Terminate Input, k	2
STOC	5016	OUTSTP	Terminate Output, k	2
STEF	5017	EXFSTP	Terminate External Function, k	2
SRD	5020	SRSM	Set Resume ff (Intercomp)	2
TIC	5021	SKPIIN	Skip Input Inact, k	2
TOC	5022	SKPOIN	Skip Output Inact, k	2
TFC	5023	SKPFIN	Skip on Ext. Fnct. Inact.	2
WFI	5024	WTFI	Wait for Interrupt	2
OV	5026	OUTOV	Force Output One Word, k	2
EFOV	5027	EXFOV	Force Ext Function One Word, k	2
AAI	5030	RIL	Remove Interrupt Lockout	2
AFI	5032	EXL	Remove Ext Interrupt Lockout	2
PAI	5034	SIL	Set Interrupt Lockout	2
PFI	5036	SXL	Set Ext Interrupt Lockout	2
SRU	5041	RSHAU	Right Shift AU, k	4-10
SRL	5042	RSHAL	Right Shift AL, k	4-10
SRA	5043	RSHA	Right Shift A, k	4-20
SCA	5044	SF	Scale A Left, k, SF	4-20
SLU	5045	LSHAU	Left Shift AU, k	4-10
SLL	5046	LSHAL	Left Shift AL, k	4-10
SLA	5047	LSHA	Left Shift A, k	4-20
TK	5050	SKP	Skip Console Key, k	2
TNB	5051	SKPNBO	Skip No Borrow	2
TOF	5052	SKPOV	Skip Overflow	2
TNO	5053	SKPNOV	Skip No Overflow	2
TOP	5054	SKPODD	Skip L(AU, AL) Odd Parity	2
TEP	5055	SKPEVN	Skip L(AU, AL) Even Parity	2
SK	5056	STOP	Stop Console Key, k	2
TRD	5057	SKPNR	Skip No Resume ff (Intercomp)	2
RND	5060	RND	Round AU	2
CPL	5061	CPAL	Complement AL	2
CPU	5062	CPAU	Complement AU	2
CPA	5063	CPA	Complement A	2
LIR	5072	ENTICR	Enter ICR, k	2
LSR	5073	ENTSR	Enter, SR, k	2

*If compare designator is set.

UNIVAC

FEDERAL SYSTEMS DIVISION

St. Paul, Minnesota

PX 5010