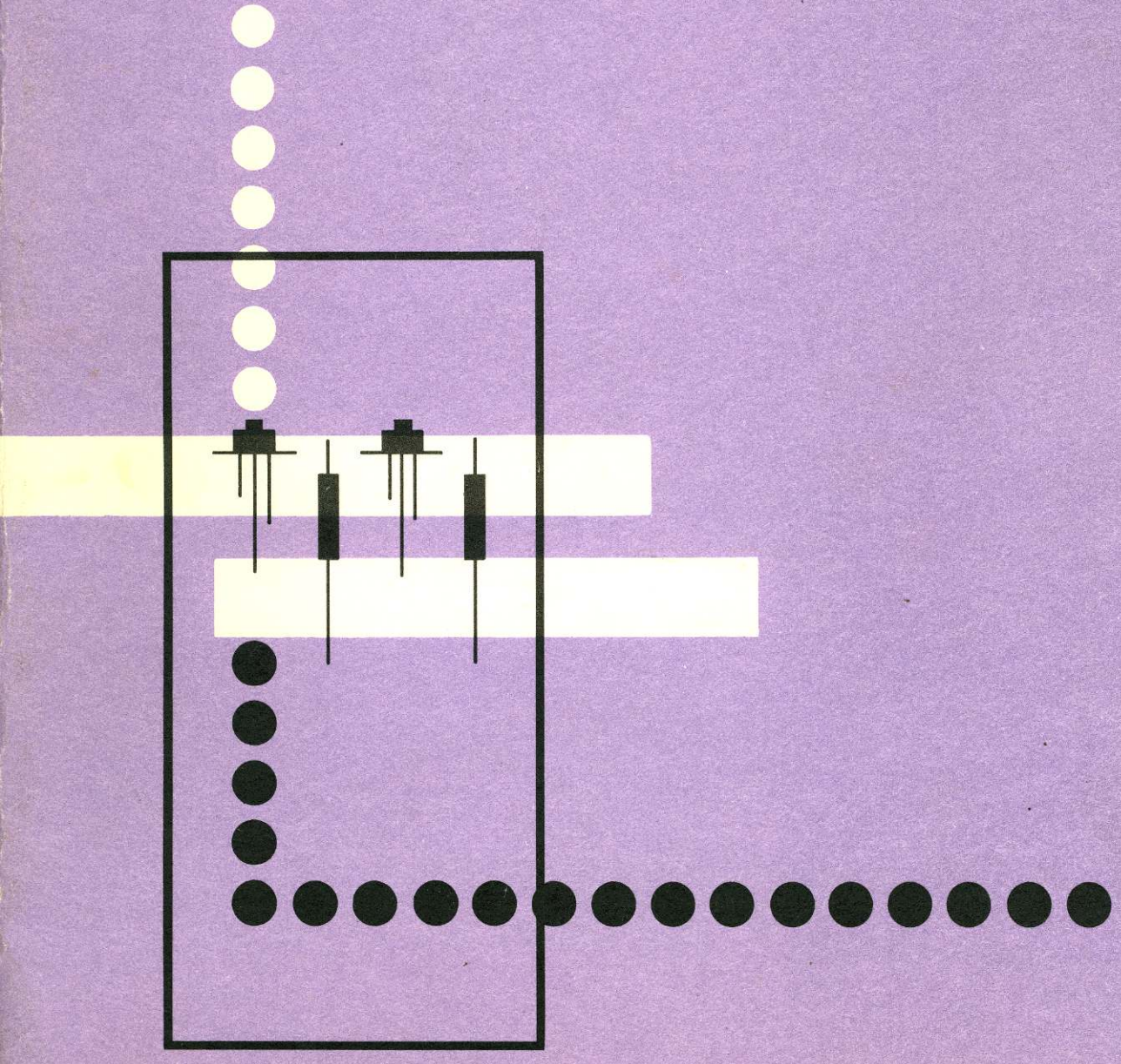
UNIVAC® 1219

MILITARY COMPUTER

TECHNICAL DESCRIPTION

≈ 160K

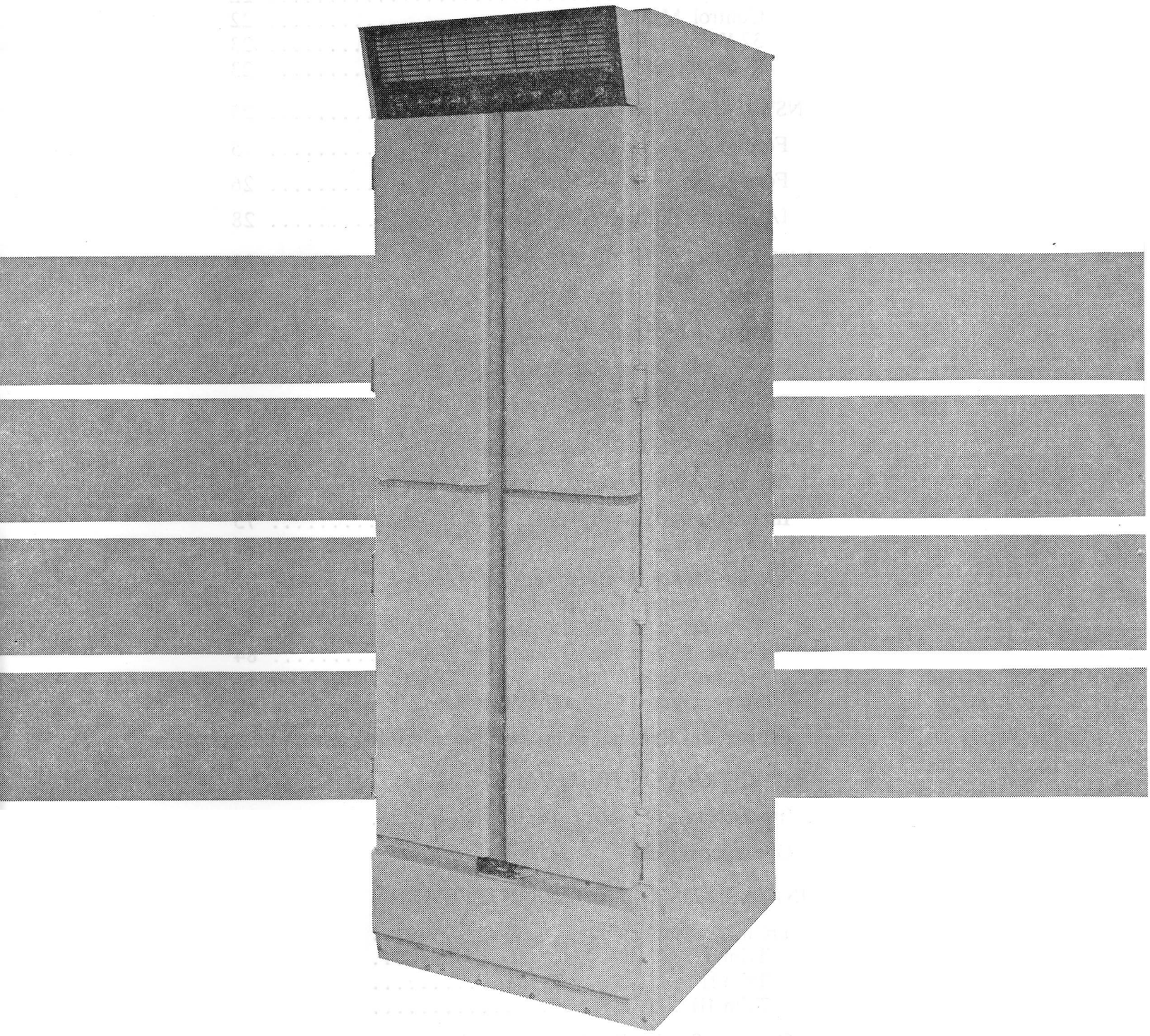


UNIVAC 1219

Technical Description

UNIVAC
DIVISION OF SPERRY RAND CORPORATION
DEFENSE OPERATIONS • UNIVAC PARK • ST. PAUL, MINN.

CONTENTS



UNIVAC® 1219 Military Computer

Section 1.

Description of Computer

General Characteristics

The UNIVAC 1219 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 faster version of the widely-used UNIVAC 1218 Computer and is functionally compatible with that machine.

To meet the extreme requirements of real time and concurrent batch processing operations, the UNIVAC 1219 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 microseconds and a fast 500-nanosecond Control Memory. In addition to this, other random-access storage devices connected to input/output channels provide unlimited memory capacities. A portion (32 word locations) of Core Memory has a characteristic non-destructive feature in which are stored 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 1219 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.

The UNIVAC 1219 Computer with a 32,768 word core memory, power supply, and all logic and control circuitry is contained in a single cabinet which occupies less than 32 cubic feet and requires less than five square feet of floor area.

A simplified block diagram of the computer appears in Figure 1. Abbreviations on the diagram are explained as the operation of the various computer sections are discussed.

UNIVAC 1219 Physical Description

The computer was built to conform to specification MIL-E-16400 (i.e. for ruggedized equipment). The single cabinet (72.0 inches high by 25.88 inches wide by 30.3 inches deep, weighing approximately 900 pounds) contains: power supply, logic circuits, core memory, (up to 32,768 words) control panel on front of cabinet, and cooling system.

Physical Size and Weight

Height:	71.75 inches	Depth:	30.03 inches
Width:	25.88 inches	Weight:	900-1000 pounds

Environment

Operating temperatures 0°C to 50°C

Non Operating temperatures —62°C to +75°C

Humidity—Relative Humidity to 95%

Power Requirements

115 volt±5 percent, 3-phase, 400 cps, 2000 watts maximum, aircooled.

115 volt±5 percent, 3-phase, 400 cps, 3000 watts maximum, water cooled.

Operational Characteristics

Memory

Control Memory

Cycle Time:	500 nanoseconds
Capacity:	128 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, or 32,768 18-bit words (standard options) 65,536 word memory (additional option)
Type:	Coincident current, magnetic core
Purpose:	I/O interrupt registers, program and data storage.

NDRO Memory

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

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/second (maximum) Multi-channel—500,000 18-bit words/second (maximum)
Operation:	Each channel fully buffered and once activated operates without program attention, asynchronous, at the rate of the peripheral unit.

Information Transfers

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

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 Channel):

18-bit parallel data transfers with storage address indirectly specified by external device; useful for multiplexing decommutating data to/from computer.

Externally Specified Address (Dual Channel):

18-bit parallel data transfers with storage 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

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

Clock

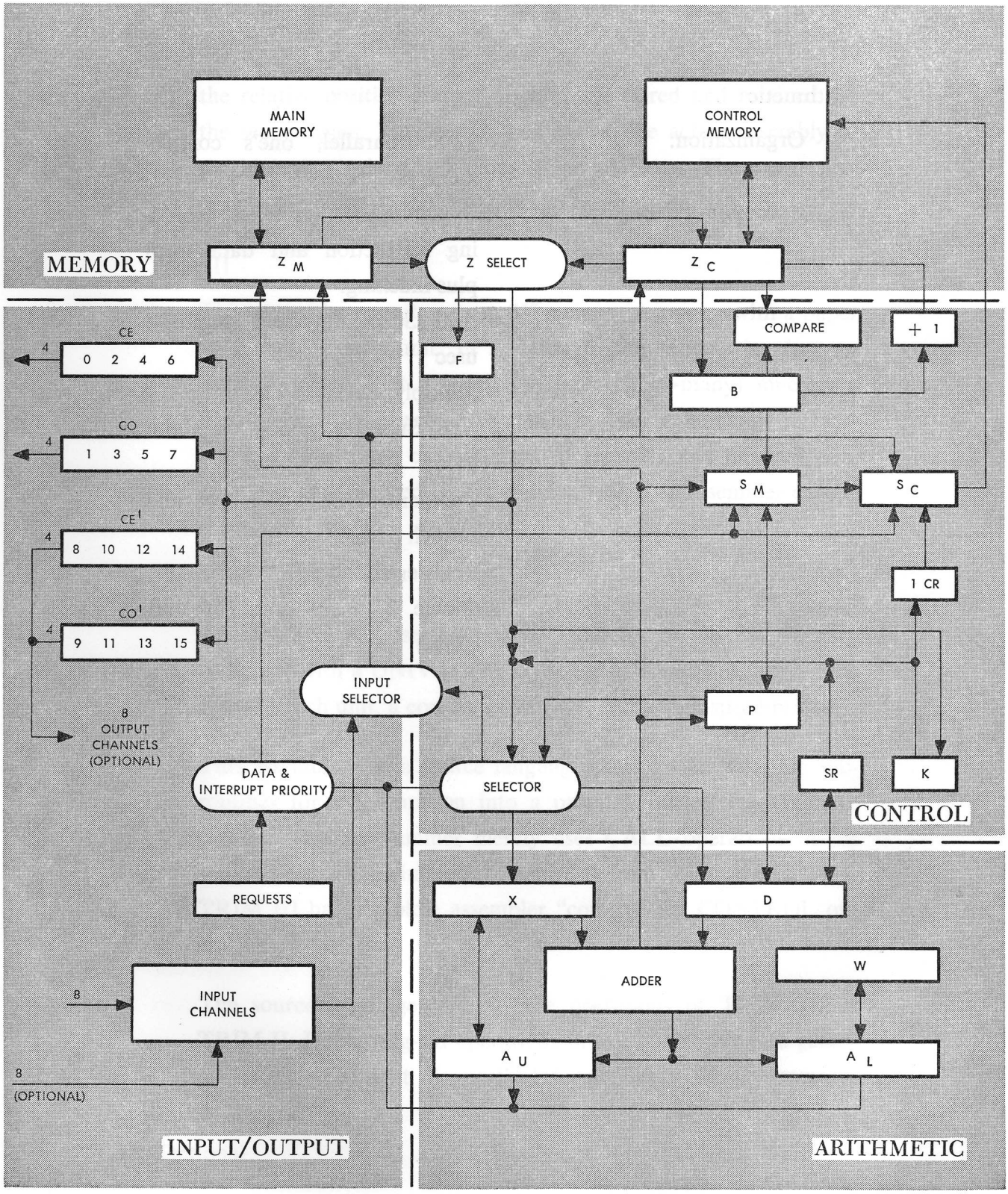
Type:	Automatic, additive, under program control
Location:	Control Memory
Duration:	Established under program control
Granularity:	LSB represents 1/1024 second
Interrupt:	Interrupt occurs when program pre-set value is reached.

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 Times:	Typical execution times, including instruction and data fetch plus indexing.
	Add, Subtract (single length) 4 usec
	Multiply/Divide 14 usec
	Add, Subtract (double length) 6 usec
	Compare/Masked Compare and Branch 6 usec
	Register shifts: right, left, single, double 2+ .5n usec (n=shift count)



UNIVAC 1219 COMPUTER BLOCK DIAGRAM

Registers and Their Contents

Information

All registers in the computer may be classified as addressable or non-addressable. Those discussed here will be other than the normal core storage registers. Addressable registers are directly available to the programmer through computer instructions. The other functional registers are non-addressable.

Addressable Registers

A A 36-bit arithmetic accumulator which:

(1) contains the product of two 18-bit quantities

(2) contains the 36-bit dividend for a divide instruction

(3) is used as an accumulator for double length arithmetic and logical functions

(4) has shifting capabilities and complementing capabilities

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) has shifting capabilities

(4) 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"

(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 instruction less than 50_8 are entered into the

18-bit B-register in the control section. "B" is an 18-bit one's complement number that may be used to increment or decrement. When the quanity "y+B" is used as an address, only the number of lower order bits sufficient to fill the S-register are transmitted. The B-register is a transient register in the control section used for address and operand modification and for 1/0 sequences to hold one of the buffer control words.

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_8 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). Refer to Section I-C.

Non-Addressable Registers

CO and CE — 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.

CO′ and CE′ — 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 operations.

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 con-

nected to a device capable of specifying an address. Sm (16 bits) is associated with Main Memory and Sc (7 bits) is associated with Control 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.

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: |0 1 0|0 0 1|0 1
() v ()	Logical sum, or inclusive OR defined by the table: 1|0 1 |0 1 1|1 1
() ⊕ ()	Half add, Half subtract, or exclusive OR defined by the table: |0 1 0|0 1 1|1 0
()′ 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.,

$$xY = \boxed{u_{11}.....u_{11} \mid u_{11}\text{————}u_0}$$

6 Bits 12 Bits

Real-Time Processing

The ability of the UNIVAC 1219 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). 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 is 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 1219 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 2-miscroseconds. All control and timing sequences for the various functions the computer performs are based on this 2-miscrosecond 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 microseconds. Any modification to this instruction and complete translation is 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.

Section 2.

Functional Information

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 particular control memory addresses which dictate the memory area affected by this input or output operation (buffer control words) and particular core memory addresses which contain instructions executed when particular 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 core 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 typewritter. Both *data* and *external function* information transfers are controlled by buffer control words in core memory.

Either an Output Buffer or an External Function mode may be active. Both use the same assigned locations in control memory for buffer control words. Thus initiating an Output Buffer will cancel any External Function Buffer on the same channel and initiating an External Function mode will cancel any active Output Buffer on the same channel.

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. The following is a list of these operating modes in descending order of priority:

(1) Real Time Clock Update
(2) External Interrupt Status Word Storage
(3) Output Request or Input Request (alternate if both)
(4) Fault
(5) Intercomputer Timeout Interrupt
(6) RTC Monitor Interrupt
(7) RTC Overflow Interrupt

(8) Synchronizing Interrupt
(9) External Interrupt
(10) Internal Interrupt from Output or External Function Transfer with Monitor
(11) Internal Interrupt from Input Transfer with Monitor

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.

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 which is stored at the address following the Interrupt Entrance register at the time the computer honors the interrupt. When an interrupt is honored, the computer will generate the addresses required to call out the instruction 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 request the computer to perform a given routine, this synchronizing input will be used to alert the computer. Top priority is given to this Interrupt, and control is transferred to the instruction located in memory address 00016.

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

---

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

(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).

When single channel operation (18-bit word transfers) is used, control is transferred to the even-numbered address for that channel and the Interrupt code is stored in the odd-numbered address for External Interrupt operation. 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.

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

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

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

- To ignore the interrupt, a 503000 instruction will remove the interrupt lockout; and the program will continue with the normal execution of instructions since the P-register is not affected by the interrupt itself.
- A response to the interrupt requires a Return Jump (usually an Indirect Return Jump), to the interrupt routine, in the interrupt entrance register. The Return Jump instruction saves the

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

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.

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 interrupts are generated by the Input/Output Section of the computer whenever a buffer, which has been initiated with a monitor imposed, terminates.

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

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).

RTC Monitor Interrupt

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).

A fault interrupt is a special case of internal interrupt caused by executing a meaningless function code, i.e., 00, 01, 77, or 50 00.

Externally Specified Index

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

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 are 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 and Input/Output control. The Arithmetic Section and Memory are shared by the Control Section and the Input/Output Section. After an instruction has been executed, the Input/Output Section may gain control of Memory for an input or output transfer on an active channel. As soon as the Arithmetic Section is available, the Buffer Control Initial (or Current) Address is compared with the Terminal Address, updated, and restored. It is also supplied to the S-register of Memory control for the reference required to transfer the word. 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.

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	Inter-Computer 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)

00040-00057	Output Buffer Control Registers (Channels 0-7)
00060-00077	Input Buffer Control Registers (Channels 0-7)

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

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

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

32-Word NDRO Memory (Bootstrap)

The computer is provided with 32_{10} non-destructive 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 magnetic cores which operate in a special type of nondestructive readout mode. They are not accessible to the programmer for store-type instructions. The Bootstrap program is assigned to memory locations 00500_8 to 00537_8 inclusive.

Main Memory

The main memory consists of a 4 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:

Main Memory Address	*Assignment*
000100-000117	External Interrupt Registers (Channels 0-7)
000120-000137	UNASSIGNED
000140-000157	Output Monitor Interrupt Registers (Channels 0-7)
000160-000177	Input Monitor Interrrupt Registers (Channels 0-7)
000300-000317	External Interrupt Registers (Channels 8-15)
000320-000337	UNASSIGNED
000340-000357	Output Monitor Interrupt Registers (Channels 8-15)
000360-000377	Input Monitor Interrupt Registers (Channels 8-15)
000400-000477	UNASSIGNED
000540-177777*	UNASSIGNED

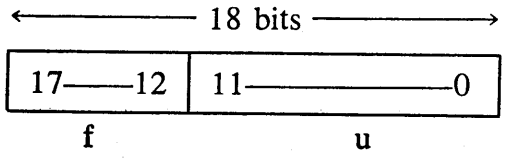
*000540 to 177777 for computers equipped with 65,536 words of memory.

Section 3.

Instruction Word Formats

Two basic instruction word formats are used by the computer.

Format I:



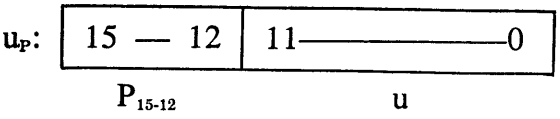
f: function code, six high order bits

u: twelve low order bits

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

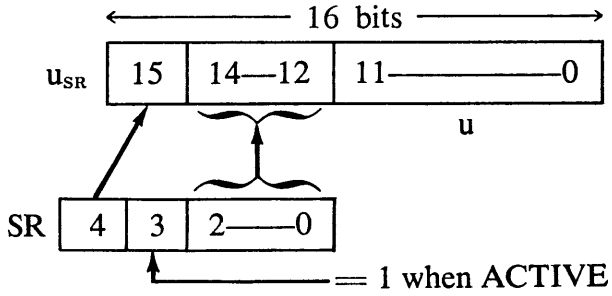
- u used as a constant. In this case, u itself is the operand and requires no further memory reference; however, u is "extended" to 18 bits. (Refer to Section 4 entitled "List of Instructions".)
- u used as an address. In this case, u is used as the lower order 12 bits of the base address referring to 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.



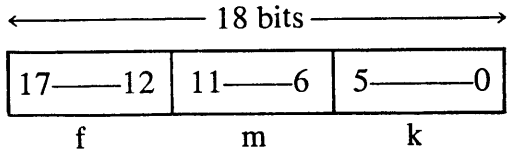
u_{SR} is defined as a 16-bit address whose four high order bits consist

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 used if SR is ACTIVE and u_P is used whenever SR is INACTIVE. (Refer to Section 4 entitled "List of Instructions".)

Format II:



f: six-bit function code (always equal to octal 50)

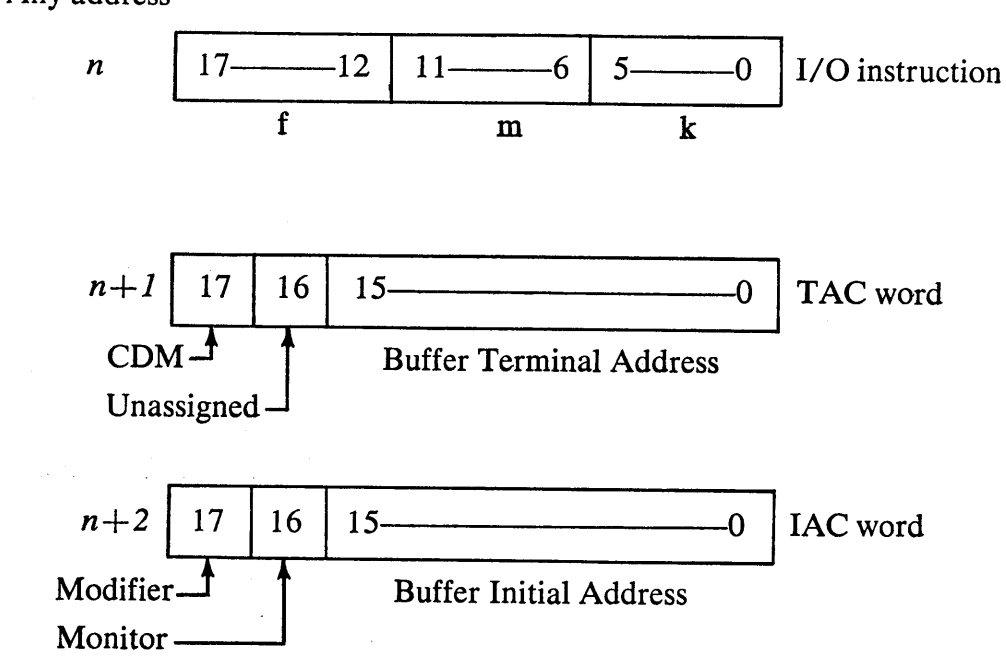
m: six-bit minor function code

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:



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 com-

pletion of the last transfer; if equal to *zero* no Moniter 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 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.

Section 4.

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 ILLEGAL CODE-JUMP TO FAULT ENTRANCE REGISTER

Execution time: 2 microseconds

01 ILLEGAL 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 0 - 67, and no interrupt will be honored while the designator is Set. (Refer to page 64 Conditional Jump Features following)

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 paragraph Conditional Jump Features.)

04 SELECTIVE SUBSTITUTES (SLSU) $L(AU)^1(AL)+L(AU)(Y) \rightarrow AL$

or $(Y)_n \rightarrow AL_n$ 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)^1(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 Conditional Jump Feature.)

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) $(Y) \rightarrow AU$

Execution time: 4 microseconds

$y = u_P$ or u_{SR}

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

12 ENTER AL (ENTAL) $(Y) \rightarrow AL$

Execution time: 4 microseconds

$y = u_P$ or u_{SR}

Clear AL. Then transmit (y) to AL.

13 ENTER AL (ENTALB) $(Y) \rightarrow AL$

Execution time: 4 microseconds

$y = u_P$ or u_{SR}

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

14 ADD AL (ADDAL) $(AL)+(Y) \rightarrow AL$

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) $(AL)+(Y) \rightarrow AL$

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.*

16 SUBTRACT AL (SUBAL) $(AL) - (Y) \rightarrow AL$

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.

---

*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).

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)

21 ADD A (ADDAB) $(A)+(Y+1,Y) \rightarrow A$

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 instructions 20.)

22 SUBTRACT A (SUBA) $(A) - (Y+1,Y) \rightarrow A$

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.)

*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).

23 SUBTRACT A (SUBAB) $A - (Y+1,Y) \rightarrow A$

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) $(AL)\ (Y) \rightarrow A$

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.

25 MULTIPLY AL (MULALB) $(AL)\ (Y) \rightarrow A$

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 factor, then correced 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).

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 × 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; Rem → 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

---

*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 running program:

$y = u_P$

Store (P)+1 at the address given in the low order 15 bits of (y), then increment that address by one and enter into the Program Address register.

Instruction executed from Entrance register on interrupt:

$y = u$

Store (P) at the address which is the low order 15 bits of (y), then increment that address by one (1) 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	71 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)

$(P)+1 \rightarrow (Y);\ (Y)+1 \rightarrow P$

Execution time: 6 microseconds

Instructions executed from running program:

$y = u_P$

Store (P)+1 at the address given in the low order 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) $(Y) \rightarrow B$

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) $(Y) \rightarrow B$

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) $Y \rightarrow P\ ;\ NI = (Y)$

Execution time: 2 microseconds

$y = u_P$

Unconditionally jump to y. (Reset P = y)

35 DIRECT JUMP (JPB) $Y \rightarrow P$; $NI = (Y)$

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) $xY \rightarrow B$

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) $(AL) \rightarrow Y$

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

Example of selective set:

$(AL)_i = 123456$

$(y) \quad = 000077$

$(AL)_f = 123477$

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)

$L(AL)'(Y) + L(AL)(Y)' \rightarrow AL$

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 LOCK-OUT (IJPEI) $(Y) \rightarrow P$ and RIL

Execution time: 4 microseconds

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

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) $Y \rightarrow P$

Execution time: 4 microseconds

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

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

56 B SKIP (BSK)

If B = (Y), SKIP NI

If B $\neq$ (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) $\neq$ 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) $\neq$ 0, substract 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, $\oplus$ (AL) = M, $\oplus$ L(AL) (AU) = M, $Y \rightarrow P$

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, Conditional Jump Features, following.

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

Otherwise, execute next instruction.

NOTE: Refer to paragraph, Conditional Jump Features, following.

62 JUMP AU NOT ZERO (JPAUNZ)
If (AU) ≠ 0 ⊕
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 "EQUALS" stage of the comparison designator is NOT SET.

Otherwise, execute next instruction.

NOTE: Refer to paragraph, Conditional Jump Features, following.

63 JUMP AL NOT ZERO (JPALNZ)

If (AL) $\neq 0$

$\oplus$ If (AL) $\neq$ M $\oplus$ L(AL) (AU) $\neq$ M, Y $\rightarrow$ 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, Conditional Jump Features, following.

64 JUMP AU POSITIVE (JPAUP)

If (AU) POS, $\oplus$

If (AL) $\geqq$ M $\oplus$ L(AL) (AU) $\leqq$ M, Y $\rightarrow$ 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) $\geqq 0$; 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, Conditional Jump Features, following.

65 JUMP AL POSITIVE

(JPALP) If (AL) POS,

(JPMLEQ) $\oplus$ If (AL) $\geqq$ M $\oplus$ L(AL) (AU) $\geqq$ M, Y $\rightarrow$ P

Execution time: 2 microseconds

$y : u_P$

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

"COMPARE" stage if comparison designator is NOT SET and (AL) $\geqq$ 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, Conditional Jump Features, following.

66 JUMP AU NEGATIVE (JPAUNG)

If (AU) NEG,

$\oplus$ If (AL) < M $\oplus$ L(AL) (AU) < M, Y $\rightarrow$ 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, Conditional Jump Features, following.

67 JUMP AL NEGATIVE

(JPALNG) If (AL) NEG,

(JPMGR) $\oplus$ If (AL) < M $\oplus$ L(AL) (AU) < M, Y $\rightarrow$ 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, Conditional Jump Features, following.

70 ENTER AL WITH CONSTANT (ENTALK) xY $\rightarrow$ 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 with 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 ≠ 0, B-1 → B and Y → P

If B = 0, Execute NI

Execution time: 2 microseconds

y = u_P

If B ≠ 0, subtract one from B then jump to y; otherwise the next instruction leaving B unaltered. (Neg Zero ≠ 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) $(AL)_{11\text{-}0} \rightarrow Y_{11\text{-}0}$

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) $(SR) \rightarrow Y_{4\text{-}0}$

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) $(P) + 1 \rightarrow Y; Y + 1 \rightarrow P$

Execution time: 4 microseconds

$y = u_P$

Store (P) + 1 at y, then jump to y + 1. This instruction trans-

fers to y a full 18-bit word, the lower 15-bits being the address (P) + 1 with the upper three 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 ILLEGAL CODE 2 microseconds JUMP TO FAULT ENTRANCE REGISTER

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 $Z_{11\text{-}6} \rightarrow F_{5\text{-}0}$ for execution. The computer maintains its regular timing sequence.

50 00 ILLEGAL CODE 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
Execution time: 6 microseconds (P+2) → 61+2k
SET INPUT ACTIVE ON CHAN. k
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
Execution time: 20 microseconds (P+2) → 41+2k
SET OUTPUT ACTIVE ON CHAN. k
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
Execution time: 20 microseconds (P+2) → 41+2k
SET EXTERNAL FUNCTION ON CHAN. k
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 CHAN. 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.

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.

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 External function 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.

50 32 REMOVE EXTERNAL INTERRUPT LOCKOUT

or (RXL)

50 33 Execution time: 2 microseconds

Enable external interrupts, all channels. Ignore k.

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 10 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" 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

50 42 RIGHT SHIFT AL (RSHAL)

Execution time: 2 microseconds (k=0). 4 to 10 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: 4.0 microseconds (k=0); 4 to 20 ($k \neq 0$)

Shift (A) circularly to the left until *either* $A_{35}=A_{34}$ 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) now 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 000000 (negative, normalized)
The computer then stores the quantity "000002" → 00017.

(c) Example of scale factor with k=1:
$(A)_i$=070000 000000 (positive, nor normalized)

after first shift 160000 000000 (positive, nor 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); 4 to 10 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 10 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 10 microseconds (k≠0)

Shift (A) circularly to the left k-bit positions. The low-

er 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 cor-

recting 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 k is *one* and the corresponding consol 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 o)	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 is 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., $(AU)+AL_{17}\rightarrow AL$

If (AU) neg., $(AU)-AL_{17}\rightarrow AL$

Execution time: 2 microseconds

If (AU) are positive, add bit position 17 of AL to (AU);

if (AU) are 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) $(AL)' \rightarrow AL$

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) $(AU)' \rightarrow AU$

Execution time: 2 miscroseconds

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

50 63 COMPLEMENT A (CPA) $(A)' \rightarrow A$

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 ILLEGAL 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 for other results.

The Comparison Designator is a 3-state bi-stable 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.

Table 1 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 table 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).

TABLE 1. 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

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: $0 \leqq (AL) < M$

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

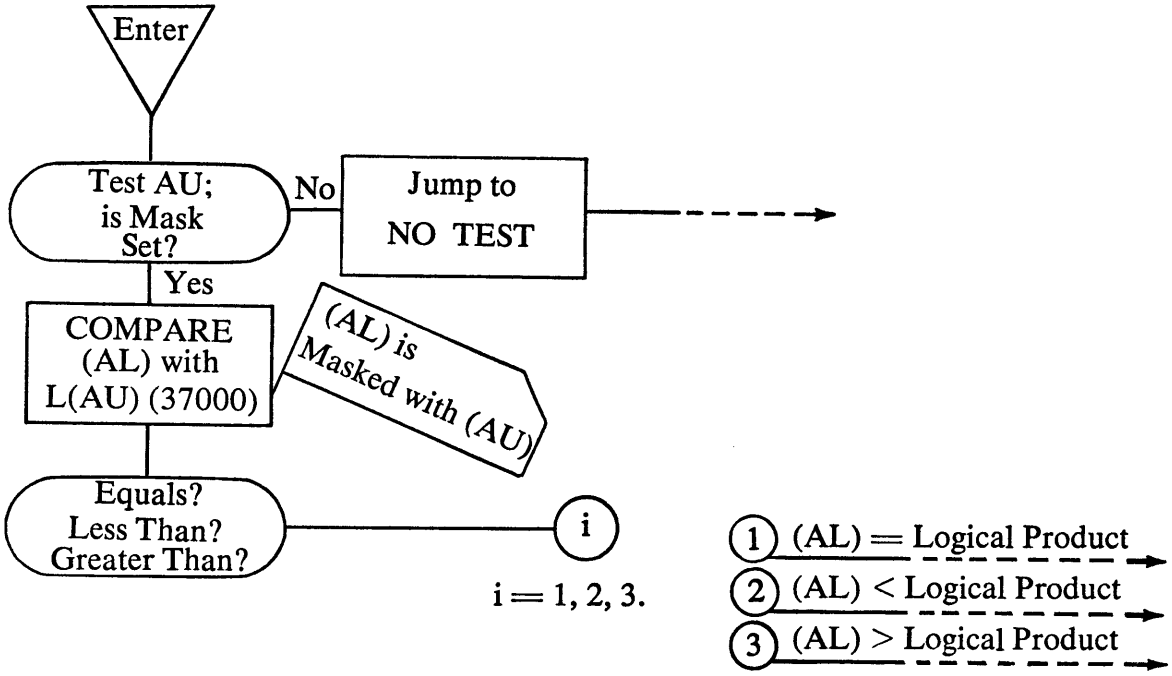
(a). The following routine results in a Jump if condition is true.

Address			*Instruction*
31000	Some Arithmetic instruction		
	65	4562	Test for (AL) $\geqq$ 0 Jump to 34562 of (AL) is positive.
	.		
	.		
	.		
	.		
34562	02	7000	SET "COMPARE" stage, also SET "LESS THAN" stage if (AL) < (37000)
	67	7500	JP to 37500 of "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 conditonal 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 O is at address 00010.
6. STRICR • O—Store 10.
7. ENTBK, ENTALK, ADDALK—Sign extend.
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 Int. Ent. 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 which ever mode is currently active.

17. 5012 and 5013; executing one will nullify the other.

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. Output—If TACW = IACW a 1 word transfer will occur.

Section 5.

Input/Output Characteristics

General Operation

Communication with the UNIVAC 1219 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 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.

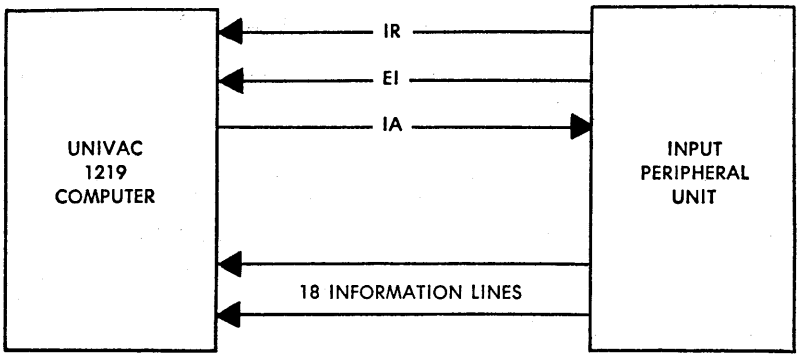
The computer is scanning for input/output or interrupts during the time it is transferring input/output data or performing an instruction. If it finds an input or output request it will not look for interrupt since I/O Scan is performed ahead of Interrupt scan. If both Input and Output requests are present,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

All references to input or output in Table 3 and Figure 2 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 Description of Input/Output Operation).

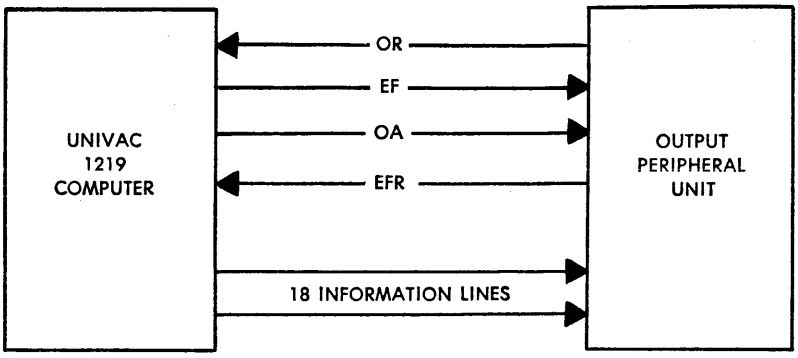
TABLE 3

DESCRIPTION OF INPUT OUTPUT CONTROL 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 1219 INPUT CHANNEL



ONE 1219 OUTPUT CHANNEL

Figure 2. Input and Output Connections

Input/Output Priority

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

The I/O scan sequence selects top function priority in order as follows:

- RTC update
- External Interrupt status word storage
- Output Request (or External Function) if preceding I/O scan sequence honored input
  Input Request; if preceding I/O scan honored output or E.F. Output or input if only one request exists

If none of the above functions are detected during the I/O scan sequence the interrupt scan is initiated to operate during the next microsecond of the computer cycle. Otherwise, the I/O sequence is initiated to read control words from control memory for the transfer being honored. Scan for interrupts is inhibited during extended sequences because interrupts cannot be honored during the execution of an instruction. The scanning sequence for interrupts is not effective when an interrupt lock-out exists.
The interrupt scan sequence selects top interrupt priority as follows if no lock-out exists.

- Fault
- Intercomputer time out interrupt

- RTC monitor interrupt
- RTC overflow interrupt
- Synchronizing interrupt
- External interrupt
- Internal Interrupt from Output or External Function Transfer with Monitor
- Internal Interrupt from Input Transfer with Monitor

Control Signals

Control signals exchanged between the computer and peripheral equipment are summarized in Table 3. All input and output statements are made with reference to the computer; that is, "input" is always input to the computer and "output" is output from the computer.

Examples will clarify the uses of the control lines. Figure 3 shows the computer communicating with a peripheral equipment over both input and output cables.

Notice the direction of information flow. "Request" and "Interrupt" signals always originate at the peripheral equipment. "Acknowledge" and "External Function" signals always originate at the computer. Examples:

(1) Normal input sequence for data transfer to the computer from peripheral equipment:

   a. Program control initiates input buffer for given channel.
   b. Peripheral equipment places data word on information lines.
   c. Peripheral equipment sets the Input Request line to indicate that it has data ready for transmission.

d. Computer detects the Input Request and clears any existing Input Acknowledge.

e. Computer samples the information lines at its own convenience.

f. Computer sets the Input Acknowledge line, indicating that it has sampled the data.

g. Peripheral equipment senses the Input Acknowledge line.

h. Peripheral equipment drops the Input Request line.

Steps b. through h. of this sequence are repeated for every data word until the number of words specified in the input buffer have been transferred.

(2) Sequence for peripheral equipment when transmitting an Interrupt code to computer:

a. Peripheral equipment places the Interrupt code on the information lines.

b. Peripheral equipment sets the Interrupt line.

c. Computer detects the Interrupt and when convenient clears any existing Input Acknowledge.

d. Computer samples the input lines and stores Interrupt Code in memory location 101 plus twice the channel number (301 plus twice the channel number for channels 10_8 to 17_8).

e. Computer sets the Input Acknowledge lines, indicating that it has sampled the information and when no data requests or interrupt lockout exist it reads its next instruction from memory location 100 plus twice the channel number (300 plus twice the channel number).

f. Peripheral equipment senses the Input Acknowledge line.

g. Peripheral equipment drops the Interrupt signal.

h. Peripheral equipment may change the data lines anytime after dropping the interrupt signal.

The Input Acknowledge is the computer response to either 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. Under emergency conditions, when data loss is of secondary importance, IR may be dropped but data lines must remain stable for not less than 4 microseconds. If, during this 4 miscroseconds interval, a IA is received, the peripheral equipment may assume successful transfer of last data word. At any time, after the 4 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.

(3) Normal output for data transfer from Computer to Peripheral Equipment:

a. Program control initiates output buffer for given channel.

b. Peripheral equipment sets the Output Request line when it is in a condition to accept data.

c. Computer detects Output Request and when convenient drops any existing Output Acknowledge or external function signal.

d. Computer (at its convenience) places data on the output information lines.

e. Computer sets the Output Acknowledge line, indicating that data are ready for sampling.

f. Peripheral equipment detects the Output Acknowledge.

g. Peripheral equipment may drop Output Request any time after detecting Output Acknowledge.

h. Peripheral equipment samples the data on the Output lines.

i. Computer drops Output Acknowledge and clears information lines.

All steps of this sequence except the first are repeated for every data word until the number of words specified in the output buffer have

been transferred. The computer also has the option of forcing any word of an output buffer; that is, it can, under program control, send an output data word regardless of the state of the Output Request line.

(4) Sequence for computer transmitting external function messages to peripheral equipment:

a. Program control initiates External Function buffer for given channel.

b. Peripheral equipment sets the External Function Request line when it is in a condition to accept External Function message.

c. Computer detects External Function Request and drops any existing Output Acknowledge or external function signal.

d. Computer (at its convenience) places External Function message on the output lines.

e. Computer sets the External Function line indicating that an External Function message is ready for sampling.

f. Peripheral equipment detects the External Function.

g. Peripheral equipment may drop the External Function Request any time after detecting the External Function.

h. Peripheral equipment samples the External Function message on the output lines.

i. Computer drops the External Function and clears the output lines.

All steps of this sequence except the first are repeated for every External Function message until the number of words specified in the External Function buffer have been transferred.

The computer also has the option of *forcing* any word of an External Function buffer; that is, it can, under program control, send an External Function message regardless of the state of the

EF request line for that channel. This option is necessary so that the computer can override whatever function the peripheral equipment is performing in order to re-establish positive control.

I/O Transfer Under Continuous Data Mode.

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 was set (Bit 17 of Terminal Address Buffer Control Word), the contents of control memory addresses 26 and 27 would be transfered to addresses 66 and 67 and the input buffer for Channel 3 will have been re-initiated 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 addresses 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.

Rules for Use of Control Signals

(1) 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. There is one exception to this rule, mentioned previously under Example (2)g. above. It is permissible for a peripheral device to drop its Input Request in order to interrupt when it needs to give the computer an urgent message and data loss is of secondary inportance. This case is philosophically similar to the computer sending a forced External Function since the computer risks destroying data or an unexecuted command when it sends an urgent External Function message.

(2) Information lines must be stable at the time they are gated into storage elements. This self-evidence 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 computer input, and computer detects the Input Request or the Interrupt before the sampling of 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 in 1 above).

(3) 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 generated 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.

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 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 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.

Because the I/O control words are located in control memory, a limitation is imposed on the ESI mode of operation. For this reason the control memory may, by option, be expanded to 256 words to gain additional control word storage. In the 1218 some delay arises when a subchannel buffer termination occurs as the computer program must search all subchannel buffer control words to determine which buffer terminated. This has been improved in the 1219 by automatically storing the subchannel address into a fixed location (the vacant address associated with the channel monitor interrupt entrance address) when termination occurs. In this manner the program can immediately determine which subchannel terminated.

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. 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 (Refer to Section 3). When the monitor interrupt occurs during an 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 instructions 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 or 5011.

Synchronizing Input

A synchronizing input is provided on the computer and is controlled by a 2-position control panel switch (External and Internal). The synchronizing input is a single line, not associated with any input or output channel, but having the same characteristics as an interrupt input. When this synchronizing line is raised from the zero state to the one state, the computer is forced to address a fixed location in memory, in which the programmer can store any instruction he wishes. The synchronizing line has the same electrical characteristics as any control line in the "slow" or "fast" interface, depending on the interface that is

supplied with the computer including the requirement that the line be dropped to zero before being set again.

If no external equipment is connected to the synchronizing input line, the computer will operate as if the synchronizing feature does not exist.

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 which 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 1219-to-1219 communication (see Figure 3), 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 should be 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.

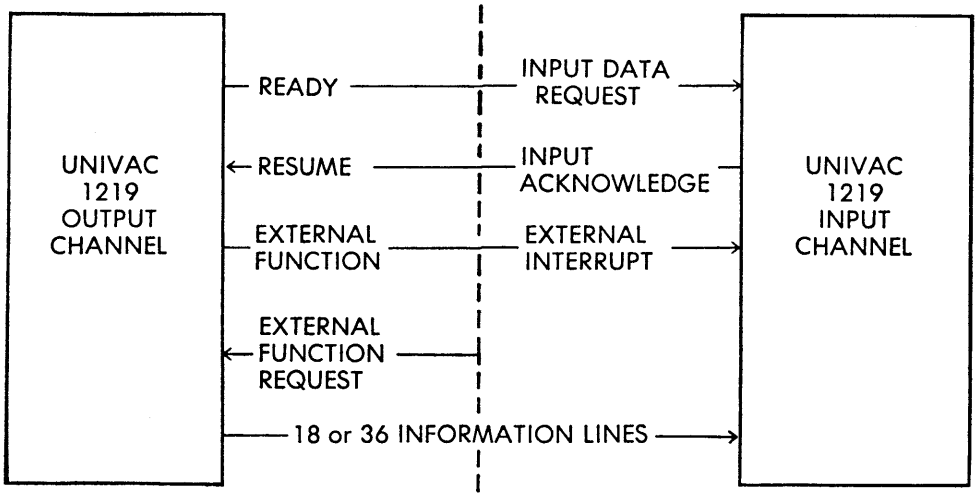


Figure 3. 1219-to-1219 Communication

Output and External Function Override Instruction

The computer has the ability, under program control, to force any External Function message of an External Function buffer or data word of an Output Buffer regardless of the state of the channel Request line. Peripheral devices should have the ability to accept such forced transmissions, realizing that loss of data or even loss of a previous External Function code is unimportant under conditions when this option is used.

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.

Input/output 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	RC-1	*Read Current address word* from Control Memory Location specified by channel address generator or Externally Specified Index.
2	RC-2	*Read Terminal Address word* from Control Memory From next sequential address; compare with terminal address word.

Order of I/O Sequence	I/O Sequence Name	Event Executed
3	WC	Modify and restore current address word to Control Memory
4	I/O-1	Transfer Data or Function word (single or dual channel). (If ESI, input from even channel, output on both).
5	I/O-2	Transfer second data or function word if dual channel operation.

The buffer terminates at the end of Sequence 4 or 5. Sequence 3 is concurrent with sequence 4.

EXTERNAL INTERRUPT timing in terms of Memory Cycle time.

Single Channel

Order of Sequence	Sequence Name	
1	I/O-1	Store the interrupt code at the odd address defined by 101*+ 2k.
2	I	Read the next instruction from the interrupt entrance address defined by 100*+ 2k.

Dual Channel and ESI

Order of Sequence	Sequence Name	
1	I/O-1	Store the odd-channel interrupt code at the odd address defined by 101*+ 2k (most significant 18-bits).
2	I/O-2	Store the even-channel interrupt code at the odd address defined by 101*+ 2k (least significant 18-bits).
3	I	Read next instruction from the odd-channel interrupt entrance address defined by 100*+ 2k.

NOTE: k = odd channel number

---

*Add 200_8 for channels 10_8 to 17_8

Section 6.

Computer Control Panel

Introduction

During the debugging or operational run of the object program, different modes of computer utilization are selectable at the computer control panel.

Operational Run

To initiate a program run:

a. Depress master clear switch.
b. Manually enter starting address in P-register.
c. Manually enter desired parameters in applicable registers.
d. Depress normal mode button.
e. Start switch.

To re-start, utilizing the bootstrap recovery memory:

a. Ensure program (on paper tape, mag tape, etc.) is properly mounted on applicable peripheral device.
b. Depress master clear switch.
c. Select load mode.
d. Depress start switch.

To step by instruction:

a. Push op-step mode button to stop the computer.
b. Depress start for each instruction.

To run at low speed:

a. Raise the start switch (to re-start position).

b. Adjust low speed control dial as desired.

c. To leave, return start switch to neutral position (this stops the computer).

NOTE: The following occurs for the various modes selected:

(1) Run mode—computer runs until it comes to stop instruction.

(2) Op-step mode: Intermittent stop at end of I sequence of each instruction.

Section 7.

UNIVAC 1219 Computer Software

UNIVAC furnishes a standard software package with 1219 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
- Fortran IV
- CS-I Compiler

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 1219 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. The source program is limited only in the number of labels used.

Trim II

This assembler operates on an 8K memory computer with a paper tape reader-punch unit. In addition to the monocodes (one-to-one) of TRIM I, it accepts polycode (one-to-many) mnemonic operations is the source program. The source language has debugging aids which cause dumps of registers and memory contents wherever desired by the programmer. The assembler can be instructed to ignore debugging operations if desired.

Trim III

TRIM III is an assembler which operates on an 8K memory computer with a UNIVAC 1240 Magnetic Tape Unit, paper tape reader-punch unit, a console typewriter, and a high-speed printer.*

This assembler has a source language librarian for selecting subroutines for incorporation into a program during the assembly process. The programmer merely uses CALL operations in his source program to implement retrieval from the source library. TRIM III has complete assembler "control" via CONTROL operations.

The source 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

---

*Optional

during a run. This works with the TRIM DEBUGGING PAK discussed later.

In operation, TRIM III makes one pass on the source program tape(s). Subroutines are rapidly retrieved from the magnetic tape source library. Assembled programs can then be written on magnetic tape or on paper tape. Edited side-by-side listing appear on the printer. The diagnostic errors are typed on the console typewriter. These features cut TRIM III assembly time to a minimum.

A typical TRIM I source program appears below:

```
CVRT12  PROG•WGH•FEB63
CVRT12  O•0              DECIMAL TO OCTAL CONVERSION
        ENTAU•TMPR4      WORD TO CONVERT
CVRT13  ENTAL•CVRT73     WORKING STORAGE
        LSHAL•2
        ADDAL•CVRT73
        LSHAL•1
        STRAL•CVRT72     INTERMEDIATE WORKING STRGE
        ENTALK•0

        LSHA•6             LEFT MOST CHAR TO A
        ADDAL•CVRT72     ADD INTERMED WORKING STRGE
        STRAL•CVRT73     STORE IN CUMULATIVE WORKING
        BJP•CVRT13                              STRGE
        IJP•CVRT12
TMPR4   0•0
CVRT72  0•0
CVRT73  0•0
. .
```

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.

Listed below are some of the operator service routines provided:

1218 UPAK—

This is a paper tape utility package which loads assembled program paper tapes and makes memory dumps on paper tapes. Other console conveniences such as inspect and change memory cell content, store memory, etc., are provided.

1218 UPAK II—

This is a utility package which loads assembled programs from paper tape or magnetic tape. The routine includes the same capabilities as UPAK.

UPAK III—

This is an expanded modular utility package. The modules of the package operate normally under console control; however, they can be activated under program control. The package has the following modules:

- Paper-tape handling
- Console operations such as inspect and change
- Magnetic tape handler for UNIVAC 1240 Tape Units. (This handler has built-in recovery techniques.)
- Magnetic tape duplicator
- Loader for assembler produced magnetic tape object programs

- Memory dump on magnetic tape for printing
- Tape-to-printer operations (UNIVAC)
- Concurrent tape-to-printer (UNIVAC)
- Memory-to-printer online (UNIVAC)
- UPAK III controlling routine

TRIM LIBRARY BUILDER—

This routine updates source magnetic tape libraries which are used with the TRIM III Assembler. It also has editing capabilities.

TRIM CORRECTOR—

This routine corrects source programs. It reads erroneous source tapes and a correction tape into the computer, makes the requested corrections, then punches a corrected source tape.

PROGRAM TRACE—

This program traces the execution sequence of a program during a processing run. It produces serial information pertaining to the address and contents of instruction executed, operand if applicable, B-register content, and the entire A register.

TRIM DEBUGGING PAK—

This is a diagnostic routine which 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.

CHANGED WORD POST MORTEM*—

This is a debugging-aid routine which runs with the programmer's routine while in the debugging phase to give information about the routine's performance. CWPM makes an image of the program being debugged by copying it in another memory area. Upon completion of each execution pass thereafter, CWPM compares the program being debugged with the image. Any changes between the program and the image become typed, punched, or printed (depending on equipment available). CWPM updates the image after presenting the changed-word message so that it is ready for the next pass.

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 users program.

Listed below 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

---

*Planned

NATURAL LOG
ARCSINE
e^x
OCTAL-TO-DECIMAL CONVERSION
DECIMAL-TO-OCTAL CONVERSION

UNIVAC 1219 COMPUTER
REPERTOIRE OF INSTRUCTIONS

CODE	SYMBOL	DESCRIPTION	TIME US
02	CMAL	Compare Y	4
03	CMALB	Compare Y + B	4
04	SLSU	Selective Substitute	4
05	SLSUB	Selective Substitute Y + B	4
06	CMSK	Masked Compare Y	4
07	CMSKB	Masked Compare Y + B	4
10	ENTAU	Enter AU, Y	4
11	ENTAUB	Enter AU, Y + B	4
12	ENTAL	Enter AL, Y	4
13	ENTALB	Enter AL, Y + B	4
14	ADDAL	Add Y, 18 bit	4
15	ADDALB	Add Y + B, 18 bit	4
16	SUBAL	Subtract Y, 18 bit	4
17	SUBALB	Subtract Y + B, 18 bit	4
20	ADDA	Add Y, 36 bit	6
21	ADDAB	Add Y + B, 36 bit	6
22	SUBA	Subtract Y, 36 bit	6
23	SUBAB	Subtract Y + B, 36 bit	6
24	MULAL	Multiply Y	14
25	MULALB	Multiply Y + B	14
26	DIVA	Divide, Y	14
27	DIVAB	Divide, Y + B	14
30	IRJP	Indirect RJP, Y	6
31	IRJPB	Indirect RJP, Y + B	6
32	ENTB	Enter B, Y	4
33	ENTBB	Enter B, Y + B	4
34	JP	Jump, Y	2
35	JPB	Jump, Y + B	2
36	ENTBK	Enter, B, U	2
37	ENTBKB	Modify B, U	2
40	CL	Store Zero, Y	4
41	CLB	Store Zero, Y + B	4
42	STRB	Store, B, Y	4
43	STRBB	Store B, Y + B	4
44	STRAL	Store AL, Y	4
45	STRALB	Store AL, Y + B	4
46	STRAU	Store AU, Y	4
47	STRAUB	Store AU, Y + B	4
51	SLSET	Selective Set (IOR), Y	4
52	SLCL	Selective Clear (AND), Y	4
53	SLCP	Selective Complement (XOR), Y	4
54	IJPEI	Indirect Jump (RIL), Y	4
55	IJP	Indirect Jump, Y	4
56	BSK	Increment B, Skip, Y	4
57	ISK	Decrement Index, Skip, Y	6
60	JPAUZ	Jump AU Zero, Y	2
61	JPALZ	Jump AL Zero, Y	2
62	JPAUNZ	Jump AU Not Zero, Y	2
63	JPALNZ	Jump AL Not Zero, Y	2
64	JPAUP	Jump AU Positive, Y	2
65	JPALP	Jump AL Positive, Y	2
66	JPAUNG	Jump AU Negative, Y	2
67	JPALNG	Jump AL Negative, Y	2
70	ENTALK	Enter AL, Y	2
71	ADDALK	Add U, 12 bits	2
72	STRICR	Store ICR, Y	4
73	BJP	Decrement B, Jump, Y	2
74	STRADR	Store Address, Y	4
75	STRSR	Store SR, Deactivate SR, Y	4
76	RJP	Return Jump, Y	4
5001	SIN	Set Input Active	2
5002	SOUT	Set Output Active	2
5003	SEXF	Set External Function Active	2
5011	IN	Initiate Input Buff, k	6
5012	OUT	Initiate Output Buff, k	6
5013	EXF	External Function	6
5014	RTC	Enable Real-Time Clock	2
5015	INSTP	Terminate Input, k	2
5016	OUTSTP	Terminate Output, k	2
5020	SRSM	Set Resume ff (Intercomp)	2
5021	SKPIIN	Skip Input Inact, k	2
5022	SKPOIN	Skip Output Inac, k	2
5024	WRFI	Wait for Interrupt	2
5026	OUTOV	Force Output One Word, k	2
5027	EXFOV	Force Ext Function One Word, k	2
5030	RIL	Remove Interrupt Lockout	2
5032	EXL	Remove Ext Interrupt Lockout	2
5034	SIL	Set Interrupt Lockout	2
5036	SXL	Set Ext Interrupt Lockout	2
5041	RSHAU	Right Shift AU, k	4-10
5042	RSHAL	Right Shift AL, k	4-10
5043	RSHA	Right Shift A, k	4-20
5044	SF	Scale A Left, k, SF	4-20
5045	LSHAU	Left Shift AU, k	4-10
5046	LSHAL	Left Shift AL, k	4-10
5047	LSHA	Left Shift A, k	4-20
5050	SKP	Skip Console Key, k	2
5051	SKPNBO	Skip No Borrow	2
5052	SKPOV	Skip Overflow	2
5053	SKPNOV	Skip No Overflow	2
5054	SKPODD	Skip L(AU,AL) Odd Parity	2
5055	SKPEVN	Skip L(AU,AL) Even Parity	2
5056	STOP	Stop Console Key, k	2
5057	SKPNR	Skip No Resume ff (Intercomp)	2
5060	RND	Round AU	2
5061	CPAL	Complement AL	2
5062	CPAU	Complement AU	2
5063	CPA	Complement A	2
5072	ENTICR	Enter ICR, k	2
5073	ENTSR	Enter SR, k	2

UNIVAC
DIVISION OF SPERRY RAND CORPORATION
DEFENSE MARKETING
UNIVAC PARK, ST. PAUL, MINN. 55116
AREA CODE 612, 698-2451

UNIVAC Regional Offices

MO 8864

WASHINGTON, D. C., 20007, *2121 Wisconsin Avenue, 338-8510* ★ HOUSTON, TEXAS, 77058, *Suite 122, Alpha Building, 16811 El Camino Real, HU 8-2240* ★ COCOA BEACH, FLORIDA, 32931, *Suite 176, Holiday Office Center, 1325 No. Atlantic Avenue, 783-8461* ★ LOS ANGELES, CALIFORNIA, 90045, *Suite 220, 5316 West Imperial Highway, 678-2531* ★ WALTHAM, MASS., 02154, *69 Hickory Drive, 899-4110* ★ SAN BERNARDINO, CALIFORNIA, 92410, *Suite 219, East Mill Street, 889-1096* ★ SAN DIEGO, CALIFORNIA, 92110, *3045 Rosecrans, 224-3333*