AN/UYK-7

TECHNICAL
DESCRIPTION

SPERRY UNIVAC

AN/UYK-7

TECHNICAL DESCRIPTION

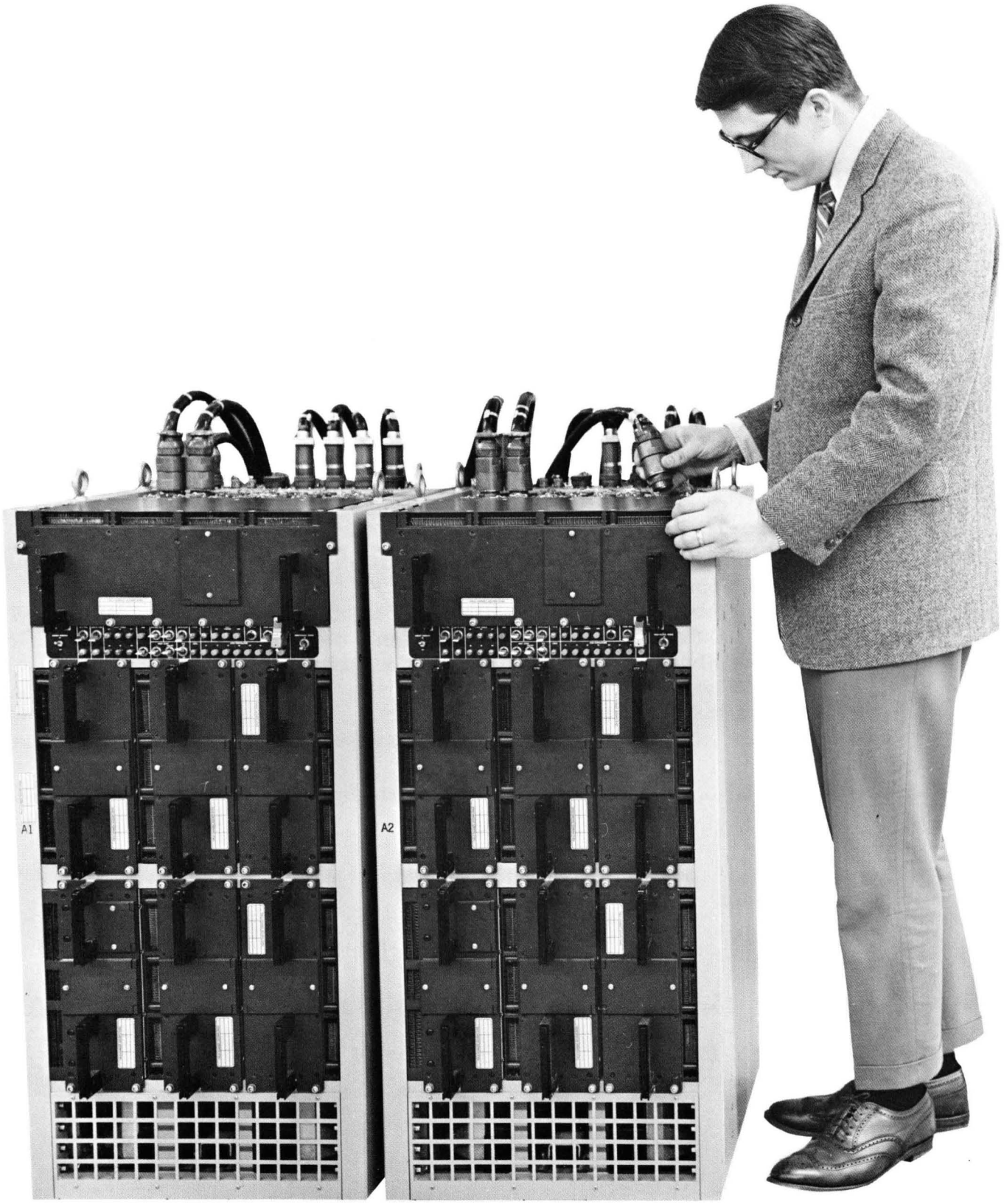
RESTRICTIVE NOTICE

This manual is intended to inform the reader regarding the general construction, operational characteristics, and capabilities of the AN/UYK-7 computer. It should not, however, be considered as an equipment specification, and Univac in no way warrants the accuracy or completeness of the manual for procurement purposes.

TABLE OF CONTENTS (CONT.)

LIST OF ILLUSTRATIONS

LIST OF TABLES



UNIVAC® AN/UYK-7 Military Computer

INTRODUCTION

The AN/UYK-7 computer is a highly reliable, ruggedized multiple-processor system designed and manufactured by the Univac Division of Sperry Rand Corporation for military applications. Advanced design techniques and a versatile functional and physical architectural philosophy provide for efficient and convenient applications across the entire spectrum of military data-processing requirements. The AN/UYK-7 can be assembled to meet small, medium, or large system configuration requirements without sacrificing functional versatility. Logical, independently timed computer modules share functional operations with asynchronous intercommunication in either the small systems, or in those involving complex manipulations of massive amounts of data, in real-time and/or scheduled activities.

Past experience in meeting the stringent environmental and functional specifications imposed on military systems guided Univac engineers in designing the computer to MIL-E-16400 (ship and shore) environmental requirements. Combinations of micro-electronic, monolithic integrated circuits, new logic, and high-density packaging techniques, facilitate application where reduced volume and weight are important considerations. The functional and physical modularity of the system affords a variety of processing and input/output capabilities for immediate and future applications.

The AN/UYK-7 features many next-generation data-processing characteristics. Rapid transfer and partial data-processing are provided in communication between external devices and large internal random-access storage. Unique timing and access priority techniques used with a 1.5-microsecond (read-store cycle) main memory permits operation as several parallel memories, thus increasing computer operating speed. Simultaneous memory references can effectively increase total memory utilization by a factor of eight in the larger configurations and by a factor of two in the single-processor instruction execution times. This advanced timing feature produces an average command execution time as low as 1.5 microseconds, and an input transfer capability of over one million 32-bit words per second in the smaller AN/UYK-7 configurations.

Specific design studies on efficient executive program operation as encountered in the real-time world revealed desirable features now incorporated in the functioning modules. Separate sets of index, arithmetic and relative address (base) registers are provided so that no processing time is lost in capturing worker-program data when the executive is initiated. A set of 18 privileged instructions with special characteristics for executive control is reserved exclusively for the executive mode. Computer activity is monitored by a dynamic status register that directs computer control and provides the status information when required by executive programs. Enhancing the capability to meet military requirements in processing data is a memory-protect feature under executive control that guards accessibility to classified data. These are among many new and vital features incorporated in the AN/UYK-7 computer for greater systems control and systems utilization.

The AN/UYK-7 is designed to operate as a single processor or as a multiple-processor system. Word lengths of 32 bits facilitate processing for byte-oriented systems as well as for word- or bit-stream-oriented types. A single-cabinet, single-processor configuration can contain up to 48K words of memory and up to 16 input/output channels. It can, by its addressing structure, command additional external memory units containing up to thirteen 16,384-word modules of magnetic core memory. An AN/UYK-7 computer configuration consists of the following modules:

Central processor
Input-output controller
Input-output adapter (4, 8, 12 or 16 channels)
Magnetic core memory (16,384-word)
Power supply
Cabinet with blowers and operating panel
Remote operating control unit
Maintenance console unit
Dummy units for unused module spaces

A minimum system contains one central processor, one input-output controller, one 4-channel input-output adapter, one memory module, and one power supply in a single cabinet. The basic one-cabinet system has three memory modules (48K words), one input-output controller, an I-O adapter with 16 input and 16 output channels, a power supply, and a processor.

Figure 1 is a functional diagram illustrating the intercommunication capabilities among the functional modules, and indicating the variety of possible configurations. Expansion of a minimum or basic system can be performed in the field. Expansion beyond a three-processor, four-input-output-controller system with 262K words of shared memory is a further objective of the design. This can be achieved with other AN/UYK-7 computers, connected to input-output controllers on inter-computer channels, and through the use of shared memory modules. Priority of action is provided to prevent channel interference.

Any channel of an input-output controller can be an inter-computer channel. This characteristic permits one AN/UYK-7 to communicate with one or more other processors or compatible computers, which, in turn, can communicate with their own peripheral devices or other computers. Such a chain configuration of computers and peripheral devices can provide a broad system capability.

PHYSICAL CHARACTERISTICS

Emphasis on functional and physical modularity in architectural design permits a wide variety of configuration assemblies. Needed amounts of memory storage, processing capability and input/output capacity can be selected as required.

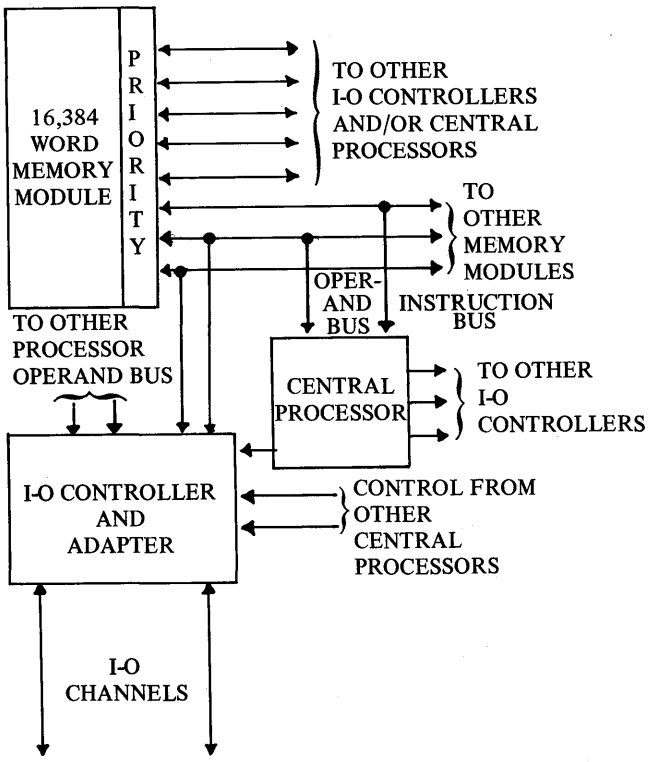


Figure 1. AN/UYK-7 Functional Diagram

Expanding the AN/UYK-7 system to meet additional requirements requires adding only those modules that perform the functions desired. Limits of expansion are determined by the amount of inter-module communication required and the addressing capability of each module. The operational size of an AN/UYK-7 computer system and the required number of cabinets result directly from the number of modules.

CABINET STRUCTURE

One basic UYK-7 welded aluminum frame cabinet accommodates all modules, including the maintenance panel which may be mounted on top. The cabinet contains module mounting slides and retaining hardware, module electrical connections and interconnecting wiring harness, operating panel, a blower, and a system of air ducts to draw cooling air through all module heat exchangers. Modules slide in from the front to make electrical connections with recepticles terminating the interconnecting harness in the rear. A combination

handle and locking mechanism on the front of each module mates with retaining forks on the cabinet for securing and releasing. This cabinet and module design permits rapid removal and replacement of plug-in modules. Electrical inter-module wiring and power distribution wiring are made accessible by removing the rear panel. The cooling blower is located at the bottom, the operator's panel at a convenient height on front. Figure 2 shows a cabinet with all modules in place.

Power connectors, input/output channel connectors, and interface circuitry are on top.

CABINET DIMENSIONS AND WEIGHT

The UYK-7 cabinet as illustrated in Figure 2 is 19.80 inches wide, 40.88 inches high and 22.34 inches deep (excluding the module handles); it occupies approximately 10.46 cubic feet. Cabinet depth, including the module handles, is 24.20 inches. The total weight of a UYK-7 computer and the power consumed are based on a specific configuration. When functional modules do not fill a cabinet, dummy units are installed in empty drawer positions to close the space and to rechannel the air flow. Weights and sizes of individual modules and dummy units are listed in Table 1.

The single processor cabinet can pass through an entry 21 inches wide and 42 inches high, or through a 30- by 30-inch hatch.

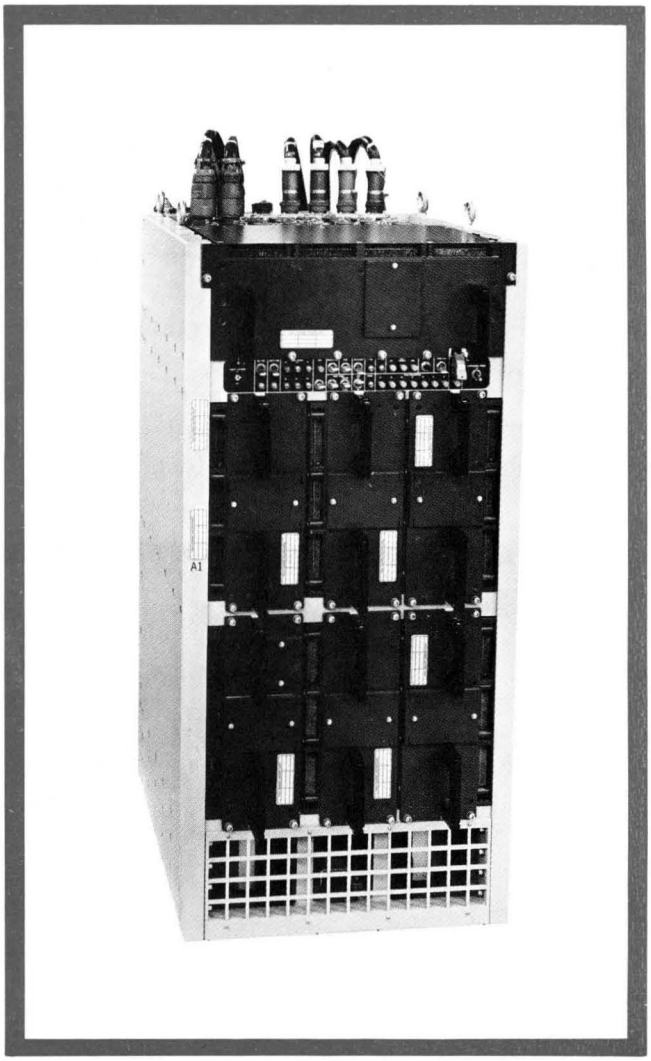


Figure 2. UYK-7 Cabinet

TABLE 1. UYK-7 MODULE/ELEMENT WEIGHT AND SIZE

MODULE/ELEMENT	WEIGHT (POUNDS)	SIZE (INCHES) (excluding handles)		
		W	H	D
Central Processor Unit	48	5.44	11.35	18.44
I-O Controller	46.5	5.44	11.35	18.44
I-O Adapter	50	17.92	6.62	18.44
Memory Unit	47.5	5.44	11.35	18.44
Power Supply	65	5.44	11.35	18.44
Cabinet (including blower)	169.5	19.80	40.88	22.34
Dummy Unit (IOC/MU/CO/PS)	6 (estimated)	5.44	11.35	18.44
Dummy I-O Adapter	12 (estimated)	17.92	6.62	18.44

MODULE STRUCTURE

Modules in the UYK-7 are built with common dimensions wherever possible (Table 1). With the exception of the power supply, each module has a wire-wrapped back panel terminating in recepticles that mate with the male connectors on printed circuit cards and memory modules (all of which are keyed and coded to prevent misinsertion).

All heat dissipated by circuit elements is transferred to the top of the card or memory assembly by thermal conduction to metallic "T" bars. The assembled module is closed by a heat-exchanger cover that makes thermal contact with all "T" bars. Ambient air, drawn through the exchanger by the cabinet cooling system, removes heat to the outside. Figures 3 and 4 show the central processor and I-O adapter modules with heat exchangers removed.

The power supply discrete components and monitoring circuits are removable to facilitate maintenance. Thermal conduction to heat exchangers is also accomplished in these modules. Figure 5 shows the power supply with bottom cover removed.

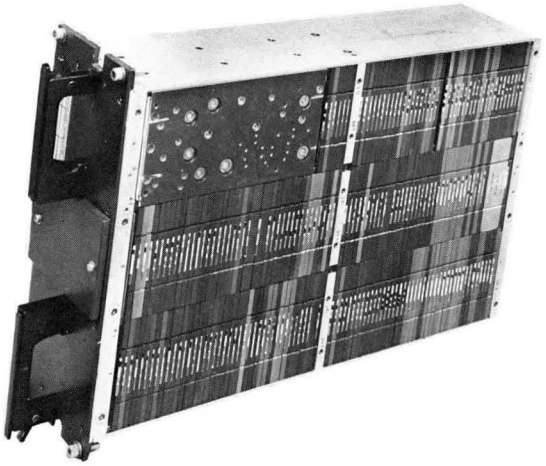


Figure 3. Central Processor (Card Side)

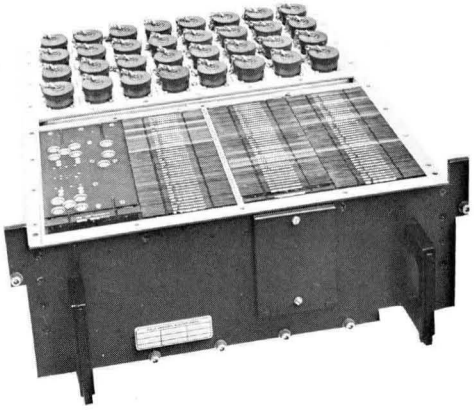


Figure 4. Input-Output Adapter (Card Side)

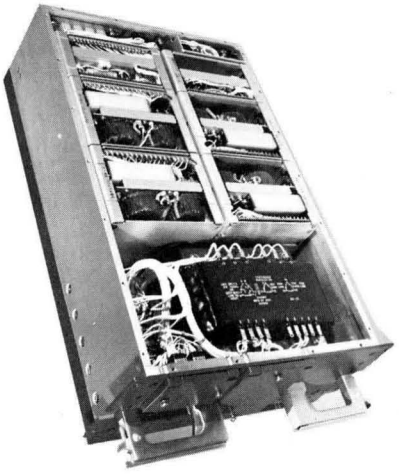


Figure 5. Power Supply

POWER REQUIREMENTS

The computer is designed to operate with primary power input as shown in Table 2. With the aid of appropriate programming, operation through transients that may cause data loss is avoided by design; contents of volatile registers are transferred to main memory via software and retained for immediate restart of processing after power returns to normal.

Power consumption for different computer configurations may be determined by using the estimates of individual module requirements shown in Table 3.

TABLE 2. PRIMARY POWER CHARACTERISTICS

PARAMETER	REQUIREMENT
Voltage	115 volts ± 10% (line-to-line) or 208 volts ± 10% (line-to-line)
Power	2500 watts (nominal for basic computer)
Phases	3 Delta (115V) or wye (208V)
Phase rotation	A: B: C
Frequency	400Hz ± 5%
Power factor	0.8 lagging (minimum)
Maximum transient voltage and frequency	As specified in MIL-E-16400

TABLE 3. MODULE/ELEMENT POWER REQUIREMENTS

MODULE/ELEMENT	POWER REQUIREMENTS
Power Supply Unit (with load of 1-CP, 1-IOC, 1-IOA, & 1 MU)	140 watts
Central Processor Unit (CP)	460 watts
Input-Output Controller (IOC)	405 watts
Input-Output Adapter (IOA)	205 watts
Memory Unit (MU)	210 watts
Maintenance Console (include PS loading)	70 watts
PS loading per added MU	23 watts
Power Transformer (with load of 1-CP, 1-IOC, 1-IOA, and 1-MU)	80 watts
Power Transformer loading per added MU	12 watts
Fans (per main cabinet)	150 watts

MODULE FUNCTIONAL CHARACTERISTICS

MEMORY MODULE

- Temperature stable coincident-current magnetic core
- Capacity: 16,384 32-bit words
- Eight access ports per module with priority selection
- 1.5-microsecond read-write cycle time
- Optional interleaved addressing between two modules

CENTRAL PROCESSOR MODULE

- Overlapped operation, two or more memory modules
- 130 basic whole and half-word instructions
  - Direct or indirect addressing
  - Variable length character addressing
  - Privileged instruction set
- Task and interrupt operating states
  - Two sets of seven index and eight base registers
  - Two sets of eight addressable accumulators
- Decremental monitor clock @ 1024 counts per second
- Interface capability
  - 16 memory modules (maximum)
  - 4 I-O controllers (maximum)
- NDRO memory 512 words
  - Initial load programs
  - Fault analysis and recovery
  - Hardware diagnostic program
- Arithmetic
  - 32-bit parallel, one's-complement, binary
  - Fixed and floating-point hardware
  - 8, 16, 32 or 64-bit operands

INPUT-OUTPUT CONTROLLER

- Direct-access data transfers to and from a maximum of 16 memory modules
- Control by 1, 2, or 3 central processors
- Programmed operations with command chaining capability
- Repertoire of 15 basic instructions
- Integrated-circuit control memory
  - Buffer control words
  - Command address pointers (function control fields)
  - Clock storage

Internal or External Real-Time Clock
- 1024 count per second (Internal)
- Up to 100 kHz (External)

Operating modes
- Normal buffer
- Externally specified index
- Externally specified address
- Command chaining
  - Continuous data mode
  - Pack and unpack 8, 16 or 32-bit bytes
- Inter-computer communication

Interface adapter
- 4, 8, 12 or 16 full-duplex input and output channels
- 32-bit parallel or optional bit-serial channels

Optional Electrical Interface			
Interface Voltage Levels in 4 Channel Groups	N15 (-15V)	N3 (-3V)	P3.5 (+3.5V)
Maximum Data Transfer Rate per Second per Channel	33,000	167,000	167,000
Bit-serial channel rate: 10 megabits per second			

POWER SUPPLY MODULE

Capacity:
- Power to six computer modules, remote operator's panel and maintenance console

Module Protection
- Overload, short-circuits or over-temperature

Power Failure Detection:
- Interrupt of central processor

Energy Storage:
- 250 microseconds after input power loss

OPERATOR'S AND MAINTENANCE PANEL

Maintenance controls, switches and indicators
Up to 15 feet of interconnecting cable
Separate cabinet

PHYSICAL CHARACTERISTICS

Militarized construction, welded aluminum cabinet with operating panel
Thermal conductive heat removal to air-cooled heat exchangers
Modular design enhances maintainability
Throw-away printed circuit cards
Convenient expansion and configuration modification:
- Single-to-multiple-processor expansion
- Shared-memory-configuration changes
- Increase of input/output channel capacity
- Intermix of main memory modules of different speeds

Basic computer configuration:
- Power supply, central processor, I/O controller, I/O interface adapter (16 channels) and 3 memory modules (49,152 words) in a single cabinet

	Computer	Maintenance Panel/Console
Size:	40.8"Hx19.8"Wx22.3"D	18.8"Hx18.5"Wx5.5"D
Volume:	10.4 cubic feet	1.2 cubic feet
Weight:	527 pounds	35 pounds

Power consumption: 2500 watts, 115V, 3-phase, 400-Hz, per MIL-STD-761A

SPECIFICATIONS AND STANDARDS USED FOR DESIGN OBJECTIVES

General Construction: MIL-E-16400 (Enclosure, Inclination)
Radio Frequency Interference: MIL-I-16910
Shock: MIL-S-901 Class I Medium Weight
Vibration: MIL-STD-167 Type I
Salt Spray: FED-STD-151 Method 811
Environmental Characteristics:
- Temperature Range:
  - -54°C to +65°C (Operating)
  - -62°C to +75°C (Storage)
- Relative Humidity to 95%

HARDWARE CONFIGURATIONS

Consideration of power requirements, power distribution, and length of the inter-module communication bus govern the positions, numbers, and types of modules per cabinet. One power supply can serve six functioning modules (one full cabinet); it is installed in the lower left position in every cabinet. Only one input-output controller and associated adapter may be installed in a cabinet; when used, they occupy top cabinet sections. Below the IOA is the local operators' panel. A cabinet may contain only one central processor. Up to five memory modules may be installed in a cabinet if positions are not used by a central processor or input-output controller. Figure 6 illustrates a 3-processor, 4-IOC multi-processor, five-cabinet configuration, with 262K memory.

MAINTENANCE UNIT

A remote maintenance unit may be placed atop each processor cabinet of the UYK-7, and connected via four cables 15 feet or less in length. This unit provides operating register displays, processor control displays, and switches for effective operating and maintenance procedures. Computer operation, however, does not require monitoring this unit. Registers are represented on the maintenance unit by rows of pushbutton/indicators, each of which can be used to enter a "1" into a corresponding bit position; and a CLEAR button to enter "0"s into all bit positions in the register. Many of the registers and indicators are involved only in the mechanics of executing instructions and input or output operations. They are not directly applicable to program manipulations, but are provided on the panel as a powerful tool for maintenance.

The maintenance unit is designed and constructed to the same ruggedized requirements as those governing the UYK-7 (Figure 7). This separate unit is contained in a cabinet 18.8 inches high by 18.5 inches wide by 5.5 inches deep, with components accessible from the front. Human engineering

IOA			DUMMY			IOA			IOA			IOA			4 IOA Modules
OP PANEL			OP PANEL			OP PANEL			OP PANEL			OP PANEL			5 Operator Panels
IOC_0	M_2	CP_0	M_7	M_6	M_5	IOC_1	M_{10}	CP_1	IOC_2	M_{13}	CP_2	IOC_3	D U M M Y	D U M M Y	4 IOC Modules 3 CP Modules Memory Modules as required
PS	M_1	M_0	PS	M_4	M_3	PS	M_9	M_8	PS	M_{12}	M_{11}	PS	M_{15}	M_{14}	5 Power Supplies Memory Modules as required.

By engineering preference, the Central Processor and the Input-Output Controller modules are always located in the upper right and upper left positions, respectively. These positions may alternatively contain Memory Unit modules if the system configuration dictates four or five memory units in a given cabinet. The Power Supply module is always located in the lower left position. In this configuration, not all memory modules are available to every CPU and IOC.

Figure 6. Full Five-Cabinet 3-by-4 Multi-processor Configuration

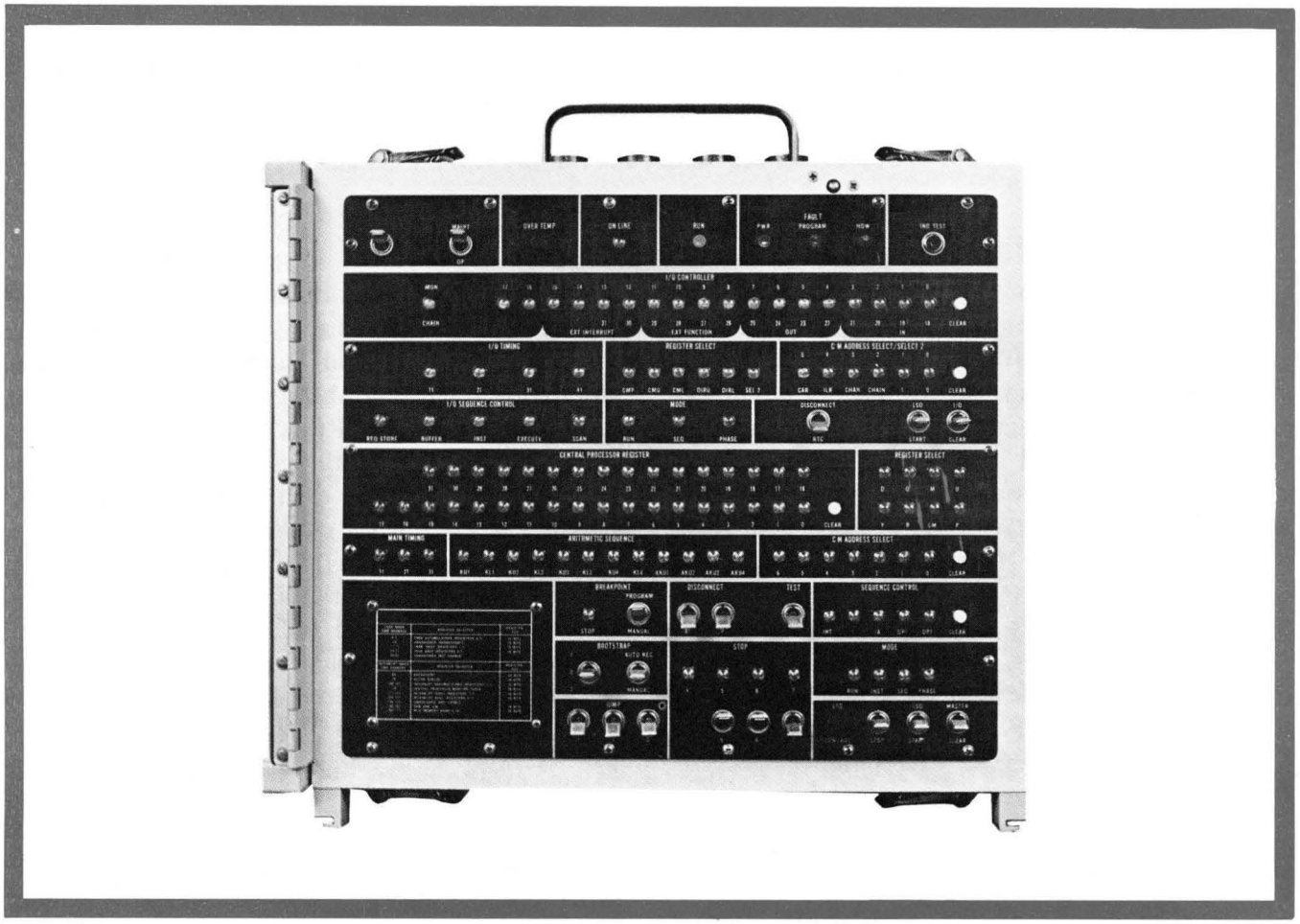


Figure 7. Maintenance Unit Panel

factors dictated the size and placement of controls used by operator and maintenance personnel. Interface and power cables are connected to the unit through corresponding jacks at the cabinet top.

REMOTE OPERATING CONTROL UNIT

A remote operating control unit can be connected to the UYK-7 to permit operational control from up to 300 feet away (total cable length). All necessary power is supplied by the cabinet power supply. Since the system operates mainly under program control, only the minimal but necessary indicators and switches are provided in the remote control panel.

INSTALLATION

Cabinet design provides front access for module removal and for normal maintenance. The minimum front clearance for module removal is 21.25 inches. Whenever the cabinet installation is subject to mechanical vibration, side and rear clearance of one inch is necessary. Figure 8 shows the dimensions of a single-cabinet installation. A minimum ceiling height of approximately 5.5 feet is required for mounting the maintenance unit, which is 18 inches high, on top of the cabinet during maintenance periods. Cable and interface connections are shown in Figure 9.

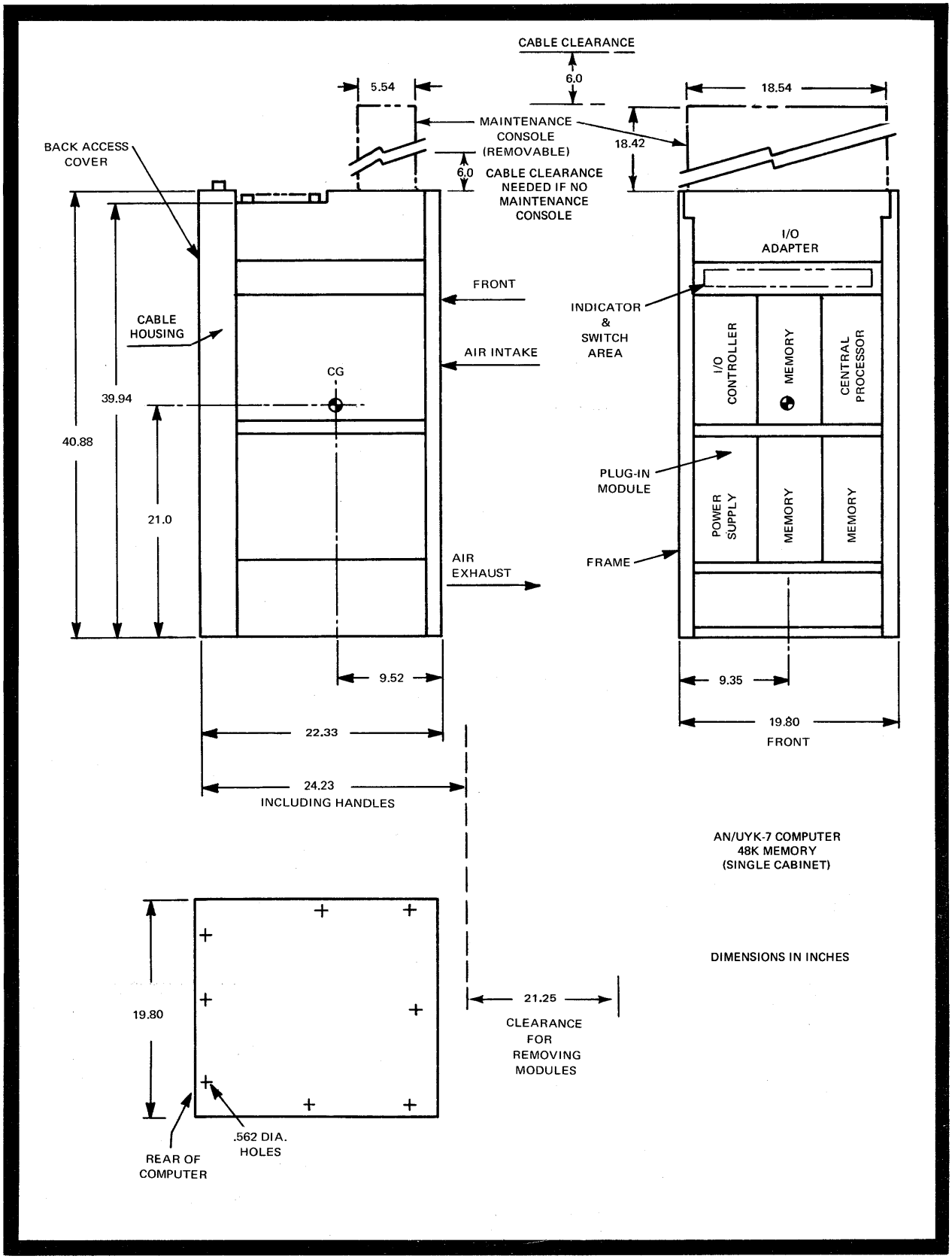
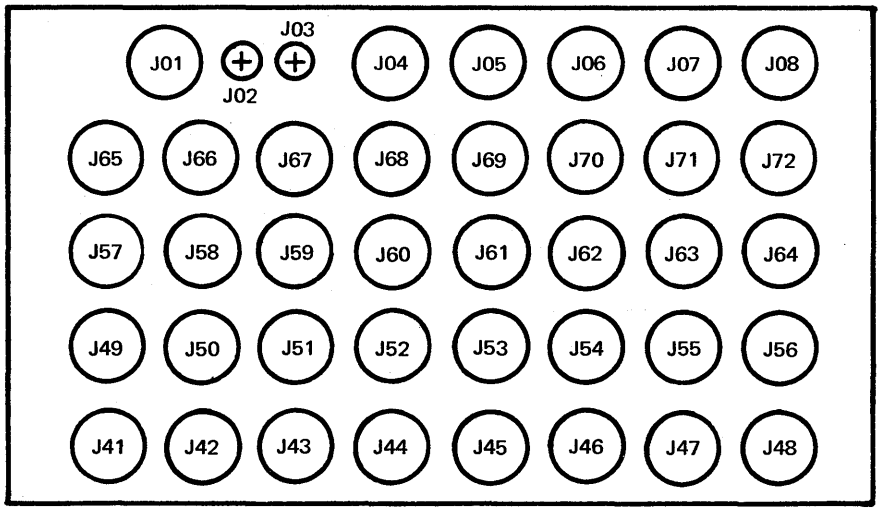


Figure 8. Cabinet Installation and Dimensional Diagram



CONNECTOR REF. DESIG.	CONNECTOR PART NUMBER	MATING CONNECTOR PART NUMBER	FUNCTION
J01	MS 3102R20-15P (905418-04)*	MS 3106R20-15S (905411-04)	PRIMARY POWER
J05	M81511/01E18-85S2 (7902701-01)	M81511/06E18-85P2 (7902700-01)	CONTROL INDICATE SIGNAL MAINTENANCE CONSOLE UNIT
J06	M81511/01E18-85S3 (7902701-02)	M81511/06E18-85P3 (7902700-02)	CONTROL INDICATE SIGNAL MAINTENANCE CONSOLE UNIT
J07	M81511/01E18-85S4 (7902701-03)	M81511/06E18-85P4 (7902700-03)	CONTROL INDICATE SIGNAL MAINTENANCE CONSOLE UNIT
J08	M81511/01E18-85S5 (7902701-04)	M81511/06E18-85P5 (7902700-04)	CONTROL INDICATE SIGNAL MAINTENANCE CONSOLE UNIT
J04	M81511/01E18-85S6 (7902701-05)	M81511/01E18-85P6 (7902700-05)	SYSTEM MONITOR PANEL OR REMOTE OPERATING UNIT
J02	MS 3112E8-4S (7900533-00)	MS 3116F8-4P (7901820-00)	EXTERNAL CLOCK (INPUT)
J03	MS 3112E8-4S (7900533-00)	MS 3116F8-4P (7901820-00)	EXTERNAL CLOCK (OUTPUT)
J41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 71	M81511/01E18-85P1 (7902698-00)	M81511/06E18-85S1 (7902699-00)	INPUT CHANNELS
J42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72	M81511/01E18-85P2 (7902698-01)	M8155/06E18-85S2 (7902699-01)	OUTPUT CHANNELS

Figure 9. Cable And Interface Connections

MAINTAINABILITY

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

Univac applies three important maintainability philosophies to the design of computing equipment:

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

MAINTENANCE DIAGNOSIS

Design of the UYK-7 permits front access to all printed-circuit card assemblies for direct testing and physical replacement. Each of the four types of logic module contains a wire-wrapped chassis of printed-circuit cards and other related components and connectors. When the module is withdrawn from the cabinet, a removable cold plate cover permits access to all cards and maintenance modules for easy replacement.

The computer system is logically and electrically designed for easy detection of malfunctioning cards or modules through diagnostic maintenance programs, in conjunction with the maintenance unit. Test points at the front of each unit assist diagnosis of malfunctions. Occasionally, standard test equipment is needed to support self-testing philosophies. However, with the diagnostic programs and related documentation, repair by module replacement and the subsequent checkout can normally be accomplished within 15 minutes.

PARTS REPLACEMENT AND COMMONALITY

The wire-wrapped chassis are assemblies of printed-circuit cards, containing integrated circuits and associated discrete components, interconnecting jacks, and a local power converter. Figure 10 is a photo of the printed-circuit cards; their dimensions are 3-3/8 by 3-3/8 by 1/4 inches (including a 56-pin connector). Any functional level can be tested by standard test equipment or diagnostic program. Established test procedures can isolate a malfunctioning card which can be removed and replaced by a spare. Larger parts, such as memory modules and power supply components, can be replaced using simple tools. Spare parts lists are compiled with commonality of parts and established failure rates as governing factors.

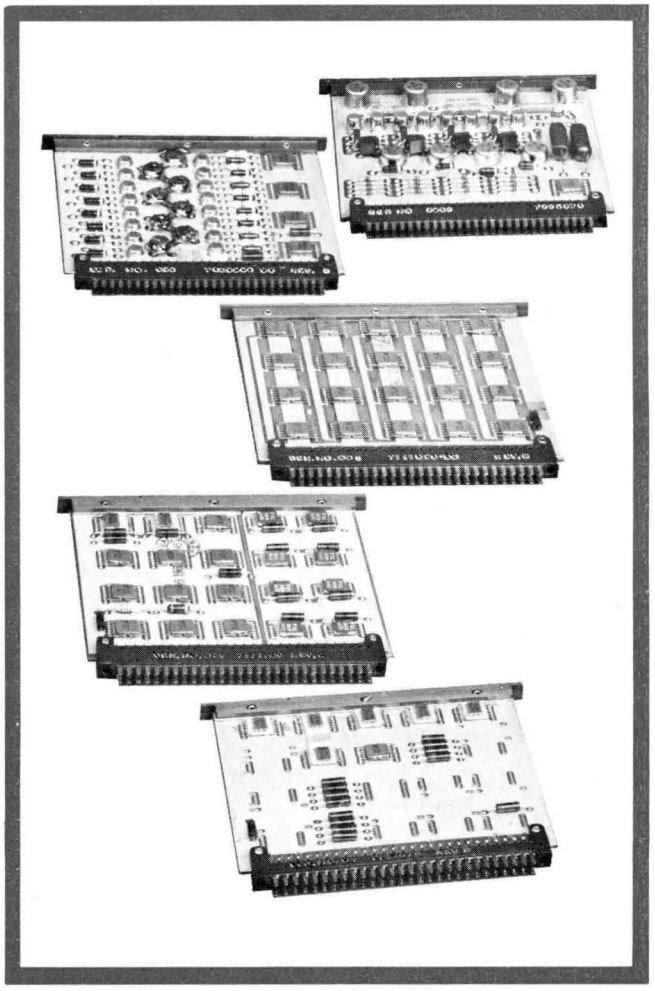


Figure 10. Printed-Circuit Cards

FUNCTIONAL DESCRIPTION

When design of functional capability looks to the future rather than to the state of the art, it dictates a division of effort among elements in a computing system, more than a stressing of speed in memories and central processors. Univac design engineers capitalize on this philosophy in the AN/UYK-7 by assigning to its various modules certain specialized tasks involved in the complete processing system.

For instance, the power supply monitors input power and senses possible failure; it monitors power consumed by the individual modules it supplies, and alerts the central processor to any detected abnormalities that could cause data loss. It also provides orderly computer start-up sequences when normal power is resumed after a shut-down. As another example, certain amounts of data preprocessing or format arrangement, buffer control, real-time clock functions, and interrupt procedures are assigned to the input-output controller. And, multiple requests addressed to a memory module at any time are retained and honored in a priority order by memory module priority circuits.

In essence, each functional module in the UYK-7 executes a certain portion of the tasks that normally have been performed by a central processor.

MAIN MEMORY

Main memory is composed of modules (banks) of random access, coincident-current destructive-readout core storage with a read-restore cycle of 1.5 microseconds. Each main memory module contains four core stacks, a power converter, address translation, and timing and control circuitry. Minimum memory for a computing system is one module. However, each central processor or input-output controller can address up to 262K words or 16 modules.

Eight interfacing paths (one bus and one port for each path), allowing access to memory, are provided in a 16K-word module for communication with other modules. Separate paths are used by the processor for storing and receiving data and for extracting instructions from storage. The interfaces are served in a priority order if simultaneous requests are presented. The order of priority is fixed at the time the interconnecting bus harness is manufactured. It, therefore, must be assigned, as desired, in the ordering document.

Ready and resume logic permits asynchronous operation with the processors and input-output controllers. Each bus that connects a memory port to the input-output controller or to the processor carries the service request and associated operand, or instruction address and the requested operands or instructions. For a specific memory reference the user (processor or input-output controller) presents a request signal and an address to memory on the interface bus. The memory module, identified by the user, responds to the request (read, write or read-and-write) when in the ready state and performs the function. Operands or instructions thus transferred are carried on the same bus as the respective addresses and requests. Sequencing and traffic direction on all busses connected to a memory module are controlled by the timing circuitry, the ready-resume logic and the priority network. When the module is not in the ready state at a particular port, any request on that port is held in the priority network for its ordered turn. This asynchronous operating philosophy permits as many read, write or read and write references to progress at any instant as there are memory modules in the system.

Contiguous Addressing

All memory modules with contiguous addressing contain addresses 00000 through 37777_8. Address translators interpret a 14-bit portion of an 18-bit address furnished by the user to select a word within an addressed module. Each such module responds to requests when the value on the selection lines corresponds to the number assigned to that module in the system.

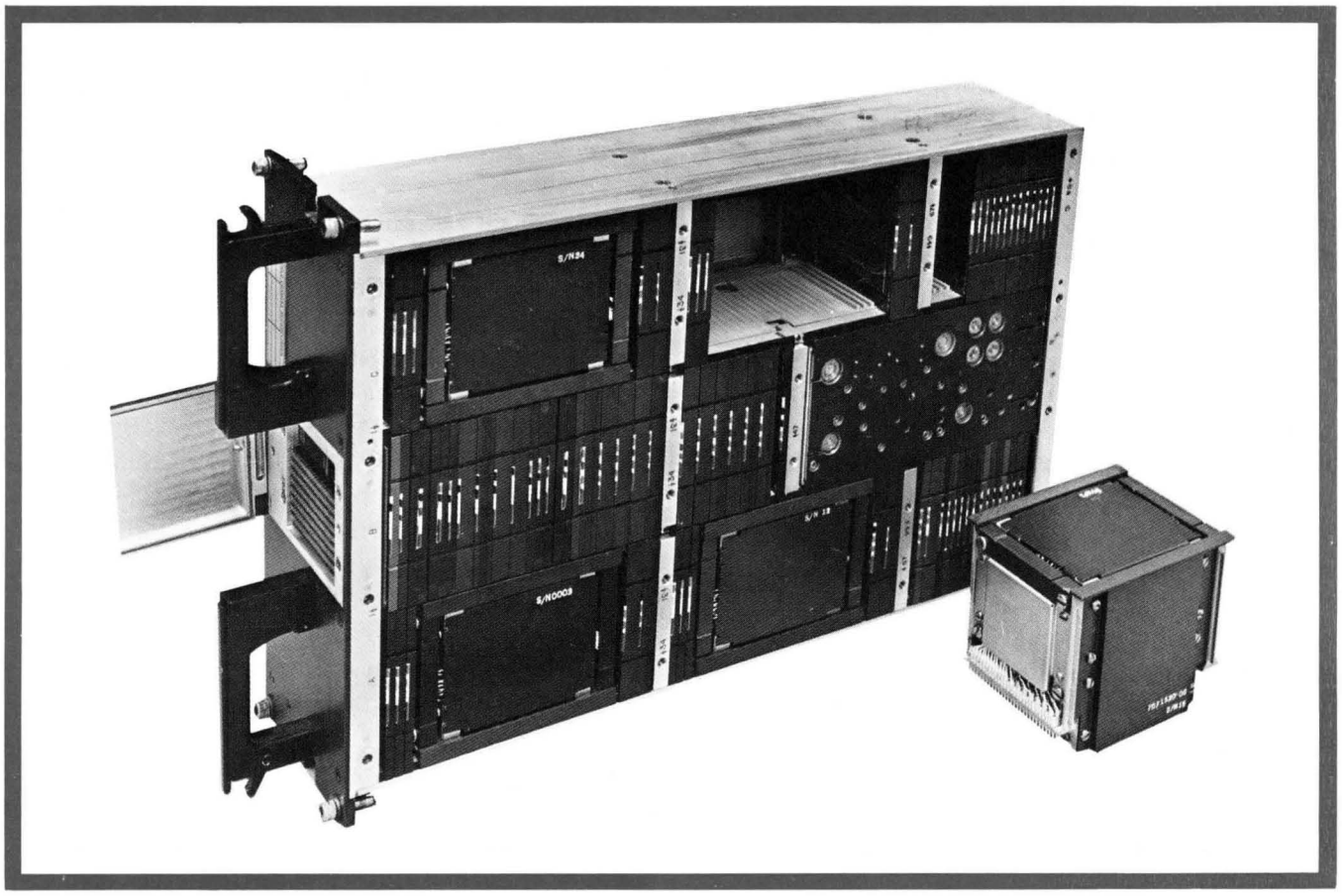


Figure 11. Memory Module

Interleaved Addressing

The UYK-7 memory system can be modified (an option) for interleaved addressing by module pairs (32K words of memory). In this configuration, the even-numbered addresses in the range 00000 through 77776 are accepted and translated by the even-numbered module of the pair, and the odd-numbered addresses in the range 00001 through 77777 are accepted and translated by the odd-numbered module of the pair.

NON-DESTRUCTIVE READ-OUT (NDRO) MEMORY

Each processor contains a 512-word assembly of NDRO magnetic core rope memory containing the hardware interrupt analysis routine, two initial load or automatic recovery routines (bootstrap), and a diagnostic program. If desired, one long, more sophisticated routine can be located in these addresses; however, the programs must be selected at the time of manufacture. The NDRO memory is separated from the total addressing continuum of main memory; the memory used (NDRO or main) depends upon the operating state and operator's panel switch positions.

Hardware error interrupts, when enabled, force the processor into the hardware interrupt analysis routine for diagnosis and transfer to remedial action or stop. NDRO routines have several entries. According to conditions causing the entries, an NDRO routine may exit to another routine or come to an orderly stop. ("Interrupt Processing" has further details.)

INTEGRATED-CIRCUIT CONTROL MEMORY

Each processor and input-output controller is equipped with a fast integrated-circuit memory

used during the execution of instructions and input or output transfers, to capture, maintain, and provide status information when needed; and to provide various controlling addresses and data as dictated by the operating programs. Cycle time is designed to provide accessibility of stored data at precise clock phases so that no time is lost in executing the operating sequences.

CENTRAL PROCESSOR

The central processor contains all the control, arithmetic and timing circuitry required for processing alpha-numeric data and for executive functions. These include all communication links with other modules under its control or within its sphere of direct influence.

Each central processor can address 262,114 words in 16 memory modules via two busses: one for instructions, and one for operands. In a two- or three-processor system, each processor may (but need not) be connected to all memory modules and input-output controllers; some isolation can be attained by not utilizing all intercommunication capabilities.

Central processors operate in two different modes or states; the Interrupt State executes the executive-type functions, and the Task State processes the worker programs. For convenience and increased response, a separate set of 7 index (B), 8 base (S), and 8 arithmetic accumulator (A) registers is available to the processor in the Interrupt State. This feature precludes the need for storing and restoring register resident data when leaving the Task State and returning. Other control memory registers become functional for the Interrupt State as assigned.

Executive Control Instructions

The multiple-processing and multi-programming capabilities of the UYK-7 system are enhanced by 16 privileged instructions which permit executive control of the operating system. These instructions, set aside for the exclusive use of the processor in the Interrupt State, include input or output transfer initiation, read and control of the monitor clock, and control of the various processing activities in both the Task and Interrupt States. If a program in the Task State (a worker program) tries to execute an executive privileged instruction, the worker routine will be interrupted and control transferred to a pre-assigned interrupt entrance address under executive jurisdiction.

Upon receiving an interrupt, program control, through instruction execution or by computer hardware, activates the Interrupt (also called "Executive") State. Either sequence sets the associated interrupt class lockout (described later), stores vital information in associated control memory locations for orderly return to the interrupted program, resets the Program Address (P) Register to the appropriate subroutine entrance address, and effects entry into the Interrupt State. Operations restricted to the Interrupt State generally involve these:

1. Manipulating the Active Status Register
2. Communicating with the input-output controllers
3. Activating the Breakpoint Register
4. Defining memory lockout functions and protected block sizes for worker programs.
5. Reading and activating the monitor clock
6. Interrupt processing
7. Maintaining inter-processor timing compatibility

Processor Control Memory

Eighty-two integrated-circuit random-access registers of appropriate size serve as the central processor control memory. The various registers are grouped into stacks according to their use, addressing, and relative size.

Access time for each stack varies according to the processor control and arithmetic section timing requirements. Addresses assigned to the various control memory registers are given in Table 4.

Addressable Registers

Addressable and non-addressable registers enter into the logical execution of all instructions and timing operations in the UYK-7. This functional description is concerned only with those addressable registers involved in operation and programming.

TABLE 4. CENTRAL PROCESSOR CONTROL MEMORY ADDRESS ASSIGNMENTS

CMR Address	Task State Assignment	Register Size (bits)
0 - 7	Arithmetic Accumulator Registers 0 - 7	32
10	Unassigned (Addressable)	19
11 - 17	Index Registers 1 - 7 (B)	19
20 - 27	Base Registers 0 - 7 (S) (addressable in interrupt mode only)	18
30 - 57	Unassigned (Not useable)	
CMR Address	**Interrupt State Assignment**	**Register Size (bits)**
6X	Breakpoint (addressable in interrupt mode only)	20
7X	Active Status (addressable in interrupt mode only)	23
100 - 107	Arithmetic Accumulator Registers 0 - 7	32
110	Central Processor Monitor Clock	19 (16-bit clock)
111 - 117	Index Registers 1 - 7 (B)	19
120 - 127	Base Registers 0 - 7 (S)	18
130 - 137	Unassigned (Not useable)	
140 - 157	Designator Storage Words (DSW), Initial Condition Words (ICW)	20
160 - 167	Storage Protection Registers (SPR) 0 - 7	21
170 - 177	Segment Identification Registers (SIR) 0 - 7	21

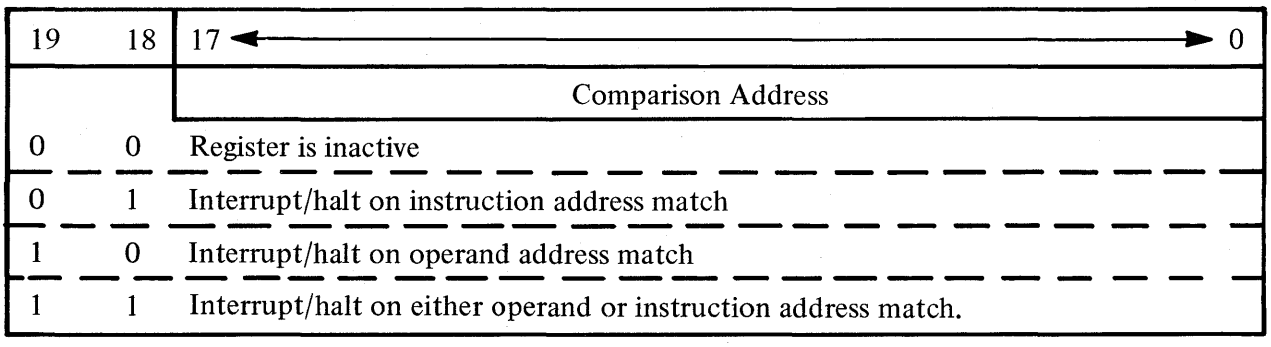


Figure 12. Breakpoint Register Format

Breakpoint Register – The Breakpoint Register contains 20 bits: 18 bits for a comparison address, and two control bits to designate action to follow a matching comparison. The breakpoint function is implemented by loading control memory register address 6X. Then, a comparison is made between the contents of the Breakpoint Register and the address of each instruction and/or operand, as designated by the control bits. If the PROGRAM/MANUAL switch is in the PROGRAM position and a match is made, the computer sets the *Breakpoint Match* interrupt and transfers the program to the Breakpoint Match Interrupt Entrance address stored in Initial Condition Word 2 (ICW2). If the switch is in the MANUAL position and equality is detected, the computer will halt. Breakpoint word format and the interpretation of elements are shown in Figure 12.

Active Status Register (ASR) – The 23-bit status register controls and indicates the status of various operations in a processor. Individual bits and groups of bits are assigned special function as shown in Table 5. If a bit assigned to a function is set, that particular state or function exists and the processor is controlled accordingly. When an interrupt condition is encountered the register contents are stored in control memory before the interrupt state is entered. The interrupt routine can then interrogate and/or change certain bits as required, after which it can return to the interrupted program in a normalized condition. Except for bits 22-15, which are hard-wired, the ASR can be changed by executing the *Load CMR* (Control Memory Register) instructions (codes 54 and 61 with i = 0), and selective bits can be changed by the following:

Instruction	Code
Jump on no overflow	53 0; a = 0
Jump on overflow	53 0; a = 1
Prevent Class III Interrupts	77 4
Allow Class III Interrupts	77 5
Compare (arithmetic)	42 thru 47 and 74 4 - 74 7

Arithmetic Registers (A) – Two sets of eight arithmetic accumulators (32 bits each) are provided in the processor control memory for flexible and efficient execution of arithmetic processes. One set is reserved for the Task State and one for the Interrupt State. Bit 10 of the Active Status Register selects the A-register set (along with the index register set). Each instruction defining an operation involving an arithmetic register selects its own accumulator (or pair of accumulators for double-length operations).

Base Registers (S) – Two sets of eight, 18-bit base registers, one for the Task State and one for the Interrupt State, are contained in the control memory for final operand and instruction address generation. The Task set is also used for initial-word address definition in the Task State memory lockout function.

A relative addressing philosophy is exercised in the central processor. The instruction furnishes a displacement address that may be indexed by one of seven index registers to form a relative address. The final effective *operand* address is formed by adding the contents of a specified base register to this relative address. The final *instruction* address is formed by adding the contents of the base register defined by the upper three bits of the Program Address Register to the relative address in the lower 16 bits.

Index Registers (B) – Two groups of seven 19-bit index registers are provided in control memory for each processor. Bit 10 of the Active Status Register selects the B-index group that operates in the Task or the Interrupt State. One group is assigned to each state so that no interference or delay is encountered during a change from one to the other. For indexing purposes, the lower-order 16 bits of the specified B-register are added to the zero-extended operand address field in the instruction; the upper three bits are not used. Special uses and functional details of the index registers are defined in the particular feature or instruction description.

Program Address Register (P) – The 20-bit Program Address Register has 19 bits which are active and one (bit 16), inactive. The three most-significant bits identify one of the eight base registers from the group with which the computer is operating and the lower 16 bits contain the relative address of the next instruction. The lower field count is increased by one, in preparation for

TABLE 5. ACTIVE STATUS DESIGNATOR

FORMAT	22——20	19	18	17	16	15	14	13	12	11	10	9	8	7	6—— 4	3	2	1	0

Bit No.	Significance
22-20	Central Processor Identification: individual bits set or cleared permanently by fixed wiring option at time of manufacture.
19	State I
18	State II
17	State III
16	State IV
	* Indicators set when corresponding class interrupt occurs; Indicators all cleared in the Task State.
15	Upper/Lower indicator: Set when an upper half-word instruction has been executed; cleared when a whole or lower half-word instruction has been executed.
14	Class I Lockout: Set when entering Class I Interrupt State. Locks out Class I interrupts except Power Tolerance interrupt.
13	Class II Lockout: Set when entering Class I or Class II Interrupt State. Locks out Class II interrupts.
12	Class III Lockout: Set when entering Class I, Class II or Class III Interrupt State. Locks out Class III interrupts.
11	Base register select: Set when entering Interrupt State. Selects base register sets for respective states.
10	Accumulator/Index register select: Set when entering Interrupt State. Selects A & B register sets for respective states.
9	MLO Disable: Set when entering Interrupt State, thereby removing memory lockouts.
8	Load Base Enable: Set by program to allow use of Load Task Register instruction (Code 05 4) in the Task State. Otherwise 05 4 is a privileged instruction.

*Analogous to Interrupt classes, which are defined later.

TABLE 5. (Continued)

7	Bootstrap Mode: Set by interrupt mode (Class II CP illegal instruction) if AUTO REC/MANUAL switch is in AUTO REC (automatic recovery) position. Set in MANUAL position by the operator.
6-4	Programmable spare bits
3	Fixed-point overflow: set when a fixed-point overflow condition occurs from those conditions: a) Addition: Addend and augend have like signs and the sum has a different sign. b) Subtraction: Minuend and subtrahend have different signs and the difference has a sign different from the minuend. c) Division: Attempt to divide by zero or if the magnitude of divisor times 2^{31} is less than the magnitude of the dividend. d) Square Root: Attempt to take square root of a negative number or a number greater than or equal to 2^{62}.
2	Compare designator: 1 = Equal 0 = Unequal
1	Compare designator: 1 = Greater than or equal 0 = Less than
0	Compare designator: 1 = Outside of limits 0 = Within limits

the next instruction address, each time an instruction is executed.

Either or both fields can be changed when a transfer to another routine is effected. The P-register can be entered manually at the maintenance unit or by program control via jump instructions. When an interrupt is processed, control hardware transfers the address from and to the P-register.

Processor Monitor Clock – A 16-bit control memory register can be activated to decrease its count at the rate of 1024 counts per second (±2 counts in 10 seconds) from a positive value loaded by the *Load CMR* instruction. When the count passes through zero, a Class II interrupt (described later) is generated and the decremental process is terminated. Any negative value entered into this Monitor Clock Register will disable the monitor clock. As the interrupt is honored, control is transferred to the program whose entrance address is stored in the control memory register labeled ICW2.

Implementing the external clock option does not change the operation of the monitor clock in the central processor because a separate oscillator module is used in the processor.

Control Section Adders

Two arithmetic adders take part in the operations involved in instruction execution. Various fields from the instruction register and addressable registers must be interpreted and their functional definitions combined to form address and arithmetic operands:

1. A 16-bit one's-complement index adder updates index counters, advances the relative address field in the P-register, forms partial operand addresses and forms literal operands for the arithmetic section when an instruction defines such operands. Results transferred in the latter case are zero-extended to 32 bits in the arithmetic section.
2. An 18-bit two's-complement base adder forms the effective (final) address of both instructions and operands. Values combined in the base adder are supplied by the index adder and specified base registers.

Values supplied to these adders containing fewer bits than the adder are zero-extended to form a full adder word and then combined with the second input value for the final sum.

Memory Read - Write Lockout

Worker programs may be prevented access to certain segments of main memory via the memory lockout feature. Any processor, in the Interrupt State, can lock out from its own non-executive operating programs read and/or write operations in defined areas of any memory module. The lockout feature is disabled when the processor reverts to the Interrupt State, and all memory locations become available to the Executive program. If a program in another processor desires the same lockout capability applied to selected areas, its lockout functions must be similarly defined.

Three groups of associated Control Memory Registers govern memory lockout functions. For any block in memory, which may be any size up to 65K words, there are these control activities:

- A base register holds the beginning address definition
- An associated storage protection register (SPR) defines the lockout function and block size (displacement: number of words, less one)
- A segment identification register (SIR) contains the relative address of the segment identifier (that address in main memory from which the lockout information is transferred)

Table 4 shows control memory assignments and word size. The *Load Base and Memory Protection* instruction (Code 05 4) is programmed to load the base register and its associated SPR from two consecutive memory locations and to save the relative address of the first in the associated SIR.

Memory protection applied to a segment of memory defined by a base register and its associated SPR governs the following operations in the task state:

1. Within the protected area
   - prevent or allow reading operands
   - prevent or allow storing operands
   - prevent or allow executing instructions
   - prevent or allow indirect addressing
2. Outside of the protected area
   - prevent any operand references
   - prevent executing instructions
3. Prevent or allow the use of the Interrupt set of index or base registers for indirect addressing.

If memory protection integrity is to be maintained, each processor must be programmed in terms of the current status of the system. The following programming considerations should be kept in mind:

1. A locked-out area for one processor can be utilized freely by another processor;
2. An active input-output controller can communicate with a locked-out area defined by any processor.

Control Of Input And Output

Input and output transfers are controlled completely by the input-output controller module addressed by a central processor. I-O command chains are stored in a memory module accessible to the controller for execution. The central processor executes an *Initiate I/O* instruction (code 07 4, a

privileged instruction) which identifies the controller (one of four possible) and the address of the first command in the input/output program sequence. The addressed controller receives an absolute address on the memory-processor operand bus, which is also connected to the controller. Subsequent activity and details are directed by the program available to the controller.

Interrupts

The UYK-7 processes data from multiple sources in real-time (as events occur) or as scheduled. This ability to process data "on demand" and continue other processing and scheduled operations is implemented by a well-organized interrupt capability.

Interrupts may originate at some remote external device, or they may originate within the computer. Since more than one may occur at the same time, the processor has a priority network with decision-making qualities so it can select the program routine 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. Thus, real-time problem solution and maximum processing potential of the system are realized, because less-important routines can occupy the processor's surplus time.

The interrupts in the UYK-7 are processed by an Executive-type program when the central processor is in the Interrupt State. Four classes (Table 5) divide all types into an orderly arrangement to permit selective processing according to importance or timing. Those in Class I are fault and hardware interrupts, including the *Power Tolerance* Interrupt which is never disabled. Class II includes program faults and error interrupts. Grouped in Class III are input and output program faults, program-imposed monitors on input and output transfers, and the *IOC Monitor Clock* Interrupt. The Class IV interrupt is a program-initiated entrance (Instruction Code 07 0) to the Interrupt State. When an interrupt is honored in any class, entrance into the Interrupt State disables all others in that class and also the classes having higher numbers, meaning lower priority.

Power Tolerance Interrupt – The power supply contains a feature that protects memory from transferring what may be defective data during primary power interruptions. However, if external sources are attempting data transfers to memory during this protected time, these data may be lost at the input interface. Conditions of both complete power failure, and of power decrease-and-recovery, are resolved without operator intervention if desired.

An interrupt is generated when power falls below a tolerable level. At this time, the computer can still operate for 250 microseconds on residual power stored in the power supply.

When a *Power Tolerance* Interrupt is generated, a program transfer is effected to the address stored in Initial Condition Word 1 (ICW1), and the Class I interrupt lock-out circuitry is set. The routine thus entered has 250 microseconds to store volatile register data. At this point in the routine, a *Manual Jump* Instruction (code 53 3) will have been programmed as a transfer to the restart routine. Design characteristics of this specific jump instruction prevent its execution at below-voltage-tolerance level. Since the voltage level is below tolerance, central processor memory requests and interrupt scans are inhibited.

Depending upon the length and severity of the power problem, two alternatives exist:

1. If power returns to normal before the failure level is reached, the jump is executed and the orderly restart is effected.
2. If the power falls to the failure level, an automatic computer MASTER CLEAR signal is generated by the power supply. Return to normal power, after the computer is cleared, will cause automatic start from the last address in NDRO if the AUTO-START switch is in the AUTO-START position, but will require operator intervention to start if the switch is in the down position.

Interrupt Processing – When honoring an interrupt, the processor will store in the Designator Storage Word (DSW) location these current values:

1. The contents of the Program Address (P) Register;
2. The contents of the Active Status Register (ASR);
3. The processor-formed, and/or input-output-controller-formed interrupt status code.

The current ASR will then be changed to reflect the new status, showing:

- appropriate Interrupt State class
- "interrupt" set of accumulator, base, and index registers
- setting of interrupt class lockouts
- removal of the memory lockout
- clearing of overflow and compare designators
- setting of bootstrap designator (if the interrupt is a Class I hardware fault; or if it is Class II and the AUTO REC/MANUAL switch is in the AUTO REC position and the processor is in the Interrupt State, and there is a *CP Illegal Instruction* Interrupt).

Interrupts occuring synchronously with central processor operations are not held pending if they are locked out. These include all Class II interrupts except *CP Monitor Clock* and *Interprocessor Interrupt*; and all in Class I, except *IOC-Memory Resume* and *Intercomputer Time-Out*.

All others, occurring asynchronously with central processor operations, are held pending so the processor can detect them during an instruction priority scan sequence. After the initialization process is completed, the program address is reset in P from the class-associated Initial Condition Word (ICW) in control memory to either the entrance address of the interrupt subroutine or to a bootstrap entrance address. The contents of the control memory ICW locations are set by the operating programs. The NDRO entrance address is dependent upon the position of the BOOTSTRAP switch (0, 1, or 2). Then, the interrupt routine or bootstrap routine is executed. The interrupt subroutine is terminated by executing the *Interrupt Return* Instruction (code 07 5). Control is returned to the interrupted state by restoring (from control memory) the ASR and P values existing at the moment of interrupt.

Multiple Interrupts – An interrupt that has not been locked out may interrupt an operating program even if the program is itself processing an interrupt. Interrupts may therefore be cascaded from Class IV to Class I or Class I to Class IV, or within a class, by clearing the appropriate lock-out designator in the ASR. Any interrupt analysis subroutine permitting such action within the same Class requires saving the Designator Storage Words for each interrupt.

IOC Interrupts – Two types of interrupts may result from IOC activity. Abnormal internal interrupts are generated by the IOC whenever the:

- inter-computer time-out exceeds the optional 250, 500, 1000, 2000, or 4000 milliseconds allowed.
- IOC detects an illegal function code
- the IOC request to memory exceeds the allotted time

Normal internal interrupts are generated by the IOC whenever:

- a buffer transfer, with imposed monitor, terminates (*Input Data, Output Data, External Function* or *External Interrupt Monitor* Interrupts)
- the IOC monitor clock is loaded with zero (*IOC-Processor* Interrupt)
- the IOC monitor clock counts down through zero (*Monitor Clock* Interrupt)

INPUT-OUTPUT CONTROLLER AND ADAPTER

The UYK-7 input-output controller contains the necessary control and timing circuitry to conduct orderly input and output transfers of data, external commands and external interrupts between accessible memory modules and the external devices on 4, 8, 12 or 16 full-duplex channels; and to update both the real-time clock register and an activated monitor clock register. IOC functions are governed by a chain of commands (input/output programs) initiated by one or more controlling central processors. Input/output programs define buffer areas, channel numbers, and any functions related to word or byte size, imposed monitors, and transfer types.

Interface Adapter

The interface adapter module associated with each IOC contains interface circuitry for 4, 8, 12, or 16 input/output channels. This interface comprises output registers, line drivers, input amplifiers, and acknowledge timing. Options exercised in the available interface voltage levels (N3, N15 or P3.5, which are -3V, -15V, and +3.5V respectively) are supplied in the adapter.

Number and characteristics of channels are provided in groups of four input and four output. Incoming data or control lines terminate in input amplifiers; each outgoing control line is driven by one line driver. However, the four output channels in each group share the 32 data line drivers; only one of the channels receives the associated control signal during the transfer. The channel interface lines for both an input and output parallel channel are shown in Figure 13.

Defined by easily-changed internal wiring, each channel can communicate in the following modes:

- Normal Buffer
- Externally Specified Index (ESI)
- Externally Specified Address (ESA)
- Inter-computer (IC)

A serial interface, with a transmission capability to 1000 feet, is offered as an option on any group of four channels, for the Normal Buffer and Inter-computer communication modes. Data, commands, synchronizing pulses, and control signals are carried on a single 72-ohm coaxial cable for each input or output channel. The serial interface option does not change the IOC or CP functional operations. Control line signals employed in parallel channels are replaced by coded control frames that precede the word transfer on serial channels. A three-bit control frame, consisting of a synchronizing bit and two

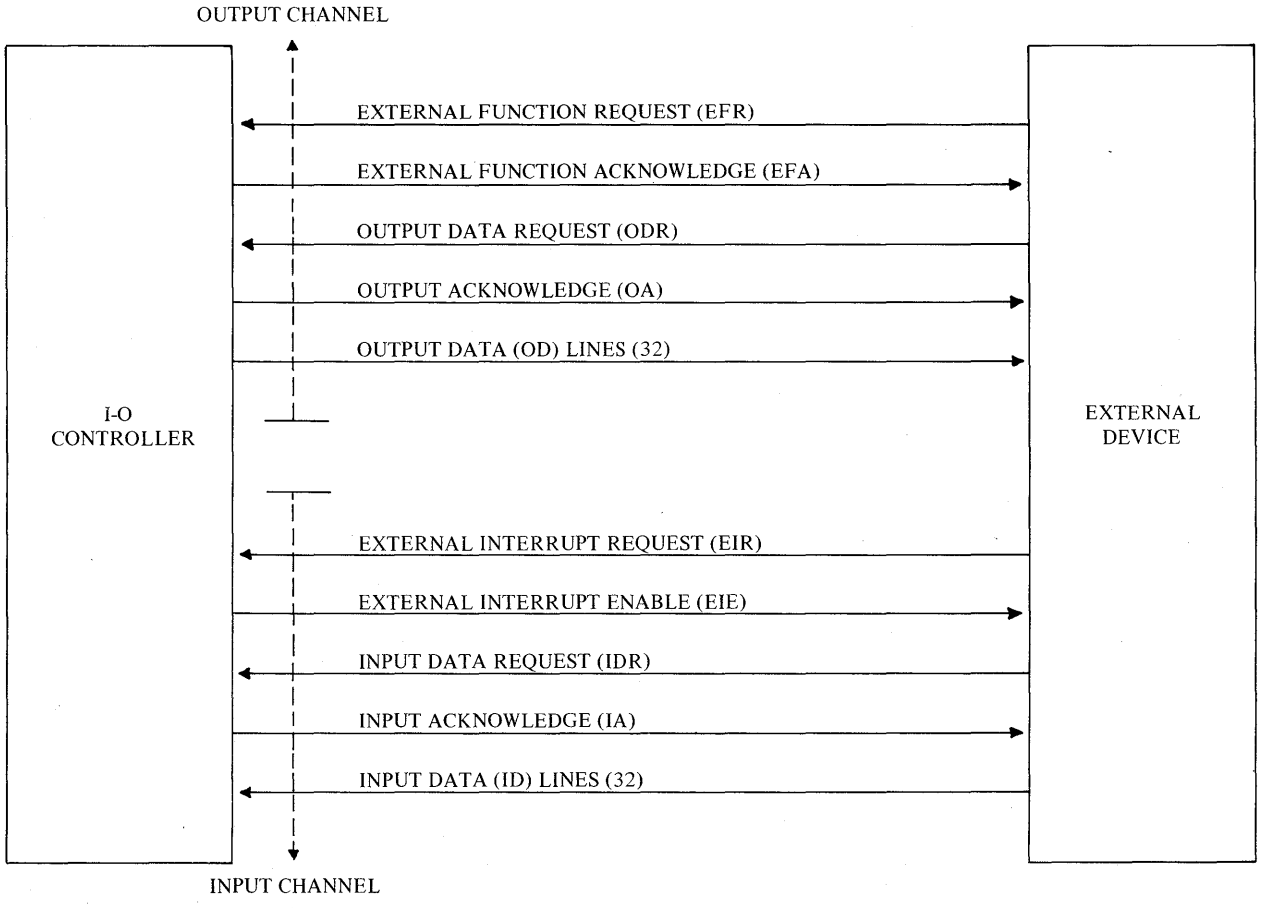


Figure 13. I-O Controller-External Device Communication

definition bits, is transmitted for each of the following:

Peripheral to computer - Input or External Interrupt Request	Input Cable
Computer to peripheral - Input or External Interrupt Enable	
Peripheral to computer - Output or External Function Request	Output Cable
Computer to peripheral - Output or External Function Enable	

The applicable control frames precede the transmission of each word (data input, data output, external function or external interrupt), which consists of a synchronizing bit and a word-identifier bit followed by a 32-bit computer word. All control bits transmitted are generated by the serial interface hardware.

The Continuous Data Mode (CDM), under program control, can be applied to the Normal Buffer and the Inter-computer operating modes.

Because the input-output adapter merely serves as an interface matching device between the IOC and peripheral equipment all discussions on input/output operation will be concentrated on the IOC.

Input-Output Controller Registers

Efficient and fast input and output operations in the input-output controller are enhanced by a fast integrated circuit control memory just as in the central processor (Table 6). One 56-bit control word is provided for each channel for each of the following operations:

- Input Data
- Output Data
- External Function
- External Interrupt

Buffer Control Words define all operations for transferring information on an active channel, and when this process is finished, point to the address of the next instruction in the command chain. Table 11 gives the word format and the field interpretations.

Real-Time Clock

A 32-bit real-time clock register counts up, at a rate of 1024 counts per second (± 2 counts per 10 seconds) when the internal clock oscillator is used, and at an external clock rate when the external clock is implemented.

TABLE 6. IOC CONTROL MEMORY ADDRESS ASSIGNMENTS

Address	Buffer Control Word For
00 - 17	Input Data on Channels $0\text{-}17_8$
20 - 37	Output Data on Channels $0\text{-}17_8$
40 - 57	External Functions on Channels $0\text{-}17_8$
60 - 77	External Interrupts on Channels $0\text{-}17_8$

Monitor Clock

A 16-bit monitor clock register in the IOC may be activated by loading it with a positive value. When it is activated, the count decreases at the rate of the clock employed (internal or external). As the count passes through zero a Class III *IOC Monitor Clock* interrupt is generated in an attempt to interrupt a central processor connected to the IOC. A zero value loaded into the monitor clock register will generate a Class III *IOC-CP* interrupt. Any negative value loaded will disable the monitor clock function.

IOC Commands

A repertoire of 15 basic instructions is provided for programming all I-O transfers and related operations assigned to the IOC. The initiating command from a central processor provides the absolute address of an IOC instruction. This may cause execution of a single instruction or the first in a chain of commands. Subsequent instructions addresses in the chain are dependent on a chain flag in the instruction.

OPERATIONAL DESCRIPTION

The AN/UYK-7 is a self-modifying, single-address processor capable of relative direct or indirect addressing. This means that, although one address value is provided by an instruction, this value can be modified to effect relative direct or indirect addressing during the execution sequence. Instructions are read sequentially from memory storage until a transfer is directed from that sequence to another routine, either conditionally or unconditionally. These conditions may be imposed by the stored program or by hardware as in the case of interrupts.

PROCESSOR

Most instructions executed by the processor are transmitted from memory via the instruction bus to the *U*-Register. The components of the instruction are translated, to direct the control section in executing the operations specified. Partial translation takes place from the *U*-Register, but further translation is required to allow for extended sequences, indirect addressing and instruction overlap. These additional levels of translation are provided by other registers (V & F).

Instruction Word Formats

Instructions for the central processor appear in five different formats according to their operational characteristics. Formats I, II and III occupy a full computer word and Formats IV-A and IV-B each occupy a half-word (see Figure 14). Two half-word instructions can be stored in one memory location. When a half-word instruction in the upper half of the computer word is executed, the processor sets bit position 15 of the Active Status Register; if a whole word instruction or one in the lower half of the computer word is executed, the bit position is cleared.

Function Word Designator Fields – Each field except *y* (constant or address field) has a particular function in controlling the various internal commands and enables for proper execution of the instruction. Some fields define the use, modification or application of the *y* field to secure the desired operand; others expand or modify the *y* field, select an accumulator, index or base register, IOC, are used as a sub-function code, or combined for special interpretation as defined by the particular instruction.

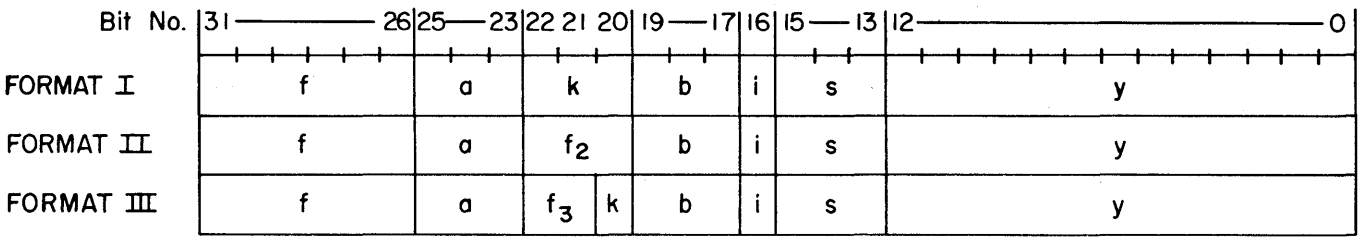
The *f*-designator appears in the most significant six bits of each instruction word. It defines in computer language the operation to be performed. The field may define the operation, or it may be used in conjunction with a sub-function designator or with other fields to define the operation. Subfunction code designators f_2 (3 bits), f_3 (2 bits), and f_4 (3 bits) in Formats II, III, and IV-A respectively, supply further operational definition in conjunction with the *f* code. Instruction codes 00, 02 f_2=1, 04, 05 f_2=5-7, 07 f_2=7, 30, 31, 70 f_4=4-7, 71 f_4=4, 71 f_4=6-7, 72, 73, 75, 76, 77 f_4=2-3 and 77 f_4=7 are "illegal"; when execution is attempted by the processor a Class II interrupt is generated, and control is transferred to the interrupt mode for resolution.

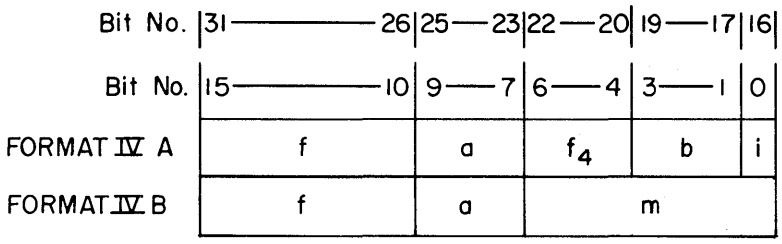
The Operand Designator (*y* field) occupies the lower-order 13 bits of instructions in Formats I, II and III. The *y*-designator furnishes the basis for operand address generation and, in combination with the contents of the *s*-field (*sy*), defines a constant that may be modified (indexed) to form the actual operand, a jump address, indirect address, or a series of identifier bits.

The *i*-designator specifies the mode of addressing, direct or indirect; the *m*-designator defines the shift count or its source. Table 7 lists the most commonly-used interpretations of the *a*, *b*, and *s* designators; Table 8 lists the *k*-designator interpretation. Other special assignments to designators are given in the descriptions of the applicable instructions.

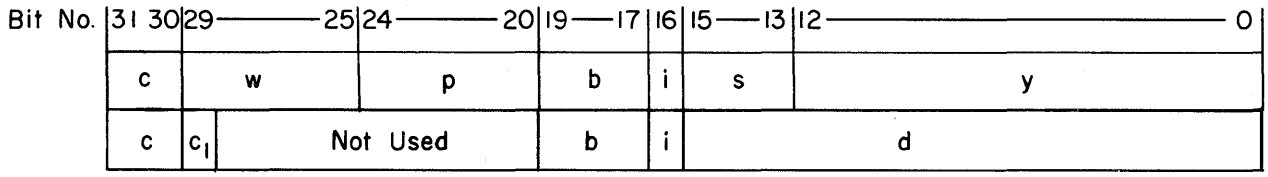
Instruction Address Generation

The program address register (*P*-register) contains the instruction address information in two distinct fields. A relative address, advanced each time an instruction is executed, is held in the *d* field. The *s* field specifies one of eight base registers which modifies the relative address, *d*, to produce the





INDIRECT CONTROL WORD FORMATS (Indirect Addressing)



Elements of the Word are Interpreted as Follows:

field	basic definition
f	6- bit function code
f_2	3- bit subfunction code
f_3	2- bit subfunction code
f_4	3- bit subfunction code
a	3- bit accumulator register designator
k	operand interpretation designator
m	6- bit shift count designator
b	3- bit index register designator
i	indirect addressing designator
s	3- bit base designator
y	13- bit address displacement /operand designator
c	2- bit control designator
c_1	1- bit indirect subfunction designator
w	6- bit character length designator
p	5- bit position indicator for character LSB
d	**16-bit address displacement**

Figure 14. Processor Instruction Word, Indirect Address Word Formats

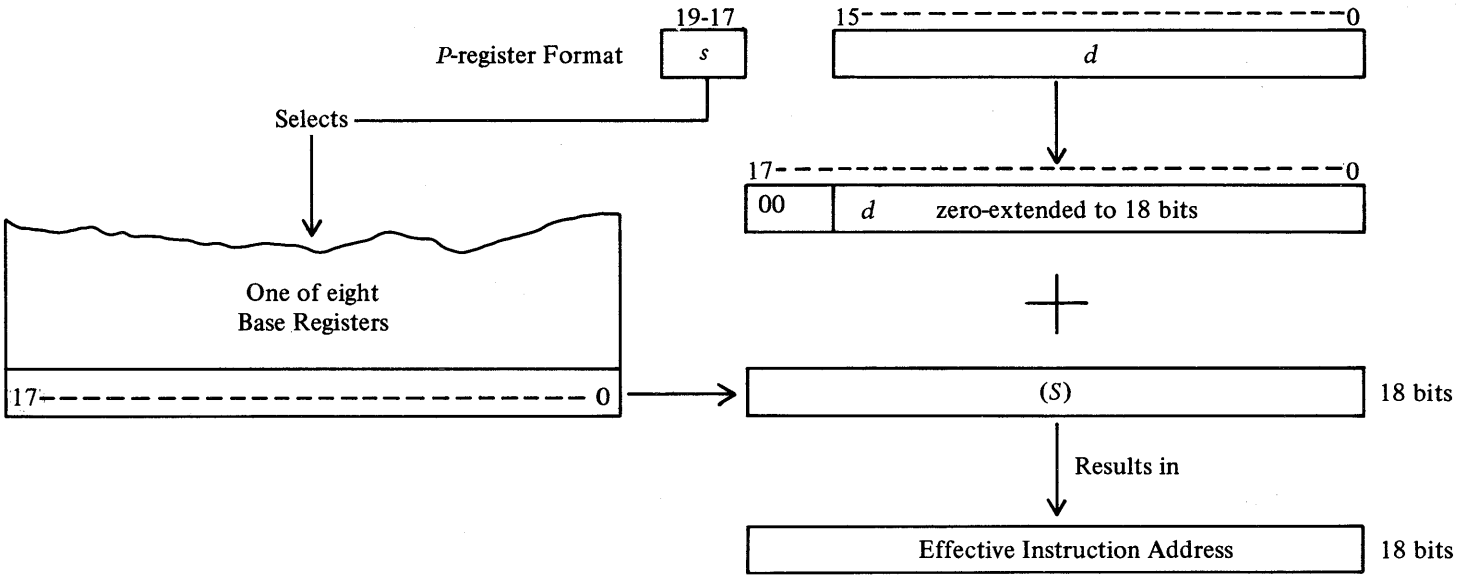


Figure 15 . Instruction Address Generation

effective address. The relationships are shown in Figure 15.

Program sequences not in the "mainstream" sequence are effected in two ways: by the use of jump instructions which change the contents of the *P*-Register, or by changing the contents of the base register specified in P_S. When the program must return to the original sequence at the point of transfer a *Return Jump* or *Load B and Jump* instruction is used in the program. Both store the contents of the *P*-Register as the return address, and then change the *P*-Register to the jump address. In the case of an interrupt, the interrupt hardware saves the contents of the *P*-Register in a control memory location and reloads P with a value from an associated control memory location for the class of interrupt honored.

Indexing

Index registers have the same format as does the *P*-Register (Figure 16). The lower-order 16 bits are used for indexing purposes or to hold the *d*-field from *P*. Bits 19-17 of an index register are used to store the *s* field of the *P*-Register. In both registers bit 16 is not used.

For indexing purposes, the 16-bit *d* value is added to the *y* or *sy* field of the instruction or of the indirect control word. When an index register is used as a counter the count in the *d* field is increased, decreased, or tested.

Control memory addresses 11-17 and 111-117 are assigned to the Task and Interrupt groups, respectively. Direct program access to the absolute

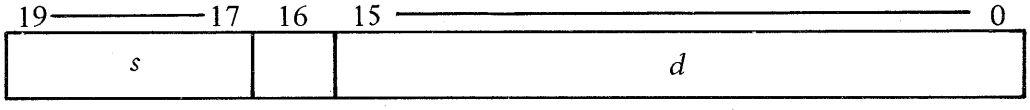


Figure 16 . *P*-Register and Index Register Format

TABLE 7. REGISTERS SPECIFIED BY THE *a*, *b*, and *s*-DESIGNATORS

Designator Value Interpreted	Designator						
	a or *b* specifies		*s* specifies	*a* specifies			
	A-Accumulator register number	*B*-index register number	*S*-Base register number	Storage Protection register number	Segment Identification register number	*S*-Base register number	IOC number
0	0		0	0	0	0	0
1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	
5	5	5	5	5	5	5	
6	6	6	6	6	6	6	
7	7	7	7	7	7	7	

In the Task State, the specified register is taken from the task set of index, accumulator and base registers. In the Interrupt State, the register selected is taken from the interrupt set of index, accumulator and base registers.

TABLE 8. FORMAT I INSTRUCTION *k*-FIELD INTERPRETATION

k	Memory to Arithmetic (Read)	Arithmetic to Memory (Store)
0	$sy_{SE}+(B_b) \longrightarrow A_{15\text{-}0}$ SE	NOT USED
1	$(Y_{15\text{-}0}) \longrightarrow A_{15\text{-}0}$ SE	$(A_{15\text{-}0}) \longrightarrow Y_{15\text{-}0}$; $Y_{31\text{-}16}$ unchg.
2	$(Y_{31\text{-}16}) \longrightarrow A_{15\text{-}0}$ SE	$(A_{15\text{-}0}) \longrightarrow Y_{31\text{-}16}$; $Y_{15\text{-}0}$ unchg.
3	$(Y_{31\text{-}0}) \longrightarrow A_{31\text{-}0}$	$(A_{31\text{-}0}) \longrightarrow Y_{31\text{-}0}$
4	$(Y_{7\text{-}0}) \longrightarrow A_{7\text{-}0}$ ZE	$(A_{7\text{-}0}) \longrightarrow Y_{7\text{-}0}$; $Y_{31\text{-}8}$ unchg.
5	$(Y_{15\text{-}8}) \longrightarrow A_{7\text{-}0}$ ZE	$(A_{7\text{-}0}) \longrightarrow Y_{15\text{-}8}$; $Y_{31\text{-}16}$ unchg. $Y_{7\text{-}0}$ unchg.
6	$(Y_{23\text{-}16}) \longrightarrow A_{7\text{-}0}$ ZE	$(A_{7\text{-}0}) \longrightarrow Y_{23\text{-}16}$; $Y_{31\text{-}24}$ unchg. $Y_{15\text{-}0}$ unchg.
7	$(Y_{31\text{-}24}) \longrightarrow A_{7\text{-}0}$ ZE	$(A_{7\text{-}0}) \longrightarrow Y_{31\text{-}24}$; $Y_{23\text{-}0}$ unchg.

k-Field Interpretation for Replace Instructions;
k=0 Not Used.
Read Cycle--Same as memory to arithmetic.
Store Cycle--Same as arithmetic to memory.
SE--Sign Extended; ZE--Zero Extended.

addresses is provided by the *Load* and *Store CMR* instructions. Otherwise they are addressable as B_1 through B_7, via the *b* and *a* designators. All instructions except *Load B and Jump* (52 0), *Jump sy+ B* (52 2), *Load* and *Store CMR* (54-57) and indirect addressing with indirect control word bits 31,30 and 29 = 001, will use only the *d* field of *B*.

Addressing

The central processor can address up to 262,144 memory locations (with internal address modification) via any specified base register. An index register may be used to modify the displacement address of an instruction, by a value up to 65,536.

Direct – Direct address generation is shown in Figure 17 . When *i* is 0, the effective address $Y=y + (B_b) + (S_s)$ is generated from the *y* field, zero-extended to 16 bits, added to the *d*-field of the index register selected by the *b* field. This quantity is a relative address which is zero-extended to 18 bits and added to the 18-bit base register selected by the *s* field.

Indirect – Indirect addressing replaces a part of the instruction word with part of an indirect word (Figure 14) which may govern execution or may call for still another indirect word. It is permitted in all Format I, II, and III words.

The *i* designator in the instruction controls the use of indirect addressing. When it is cleared, the instruction functions normally with direct addressing. But when it is set, the *b, i, s* and *y* fields of the *U*-Register (program control register) are replaced by the contents of an indirect word from the address defined by the present contents of the *U*-Register. The higher order 12 bits of the indirect word are interpreted in conjunction with the *i*-field.

As each indirect word is called in, its *i*-field governs additional indirect addressing. If each new i in the indirect word is set, replacement in the *U*-Register will continue as cascaded indirect addressing. If *i* in a new indirect word is cleared, indirect addressing terminates, and bits 31, 30, and 29 designate the type of indirect addressing. (Table 9.)

With character addressing, each character is in the main memory word bit positions defined by the *w* and *p* fields. Transfers to the arithmetic register are right-justified. Transfers from the arithmetic register to memory are defined by the *w* low-order bits of the register and stored in memory with the lowest-order character bit at memory word bit *p*. Each time a sequential character instruction is executed, the *p* field of the indirect word is modified to select the character adjacent to the lowest order bit of the one previously selected. If any portion of that character would fall outside the 32-bit word structure for the next execution, the *y* and *p* fields are modified to locate that character instead at the high-order bits of the next sequential memory address. Examples are shown in Figure 18.

Repeat Mode

A *Repeat* instruction (Code 07 6) causes the next sequential instruction to be repeated the number of times specified by (B_7), which must be loaded prior to the execution of the repeat instruction. Conditions for terminating the process, and operand address modification for the repeated instruction are defined in the repertoire description of the *Repeat* instruction.

Instructions having no logical reason to be repeated and those that require double-word-length operands cannot be repeated. If an attempt is made to repeat such an instruction, the repeat mode may clear with the repeated instruction executed once, or the repeat mode may go to completion with unreliable results from the repeated instruction.

Instructions that should not be repeated are:

- All Format III, IV-A and IV-B instructions.
- The following Format II instructions:
  - Double-word operands
  - Floating-point instructions
  - Execute remote instructions
  - Initiate I/O
  - Prevent and Allow interrupts
  - Interprocessor interrupt
  - Load IOC monitor clock
  - Enter executive state
  - Repeat
  - Load task and memory protection
- The following Format I instructions:
  - Load A and Index B
  - Store A and Index B.

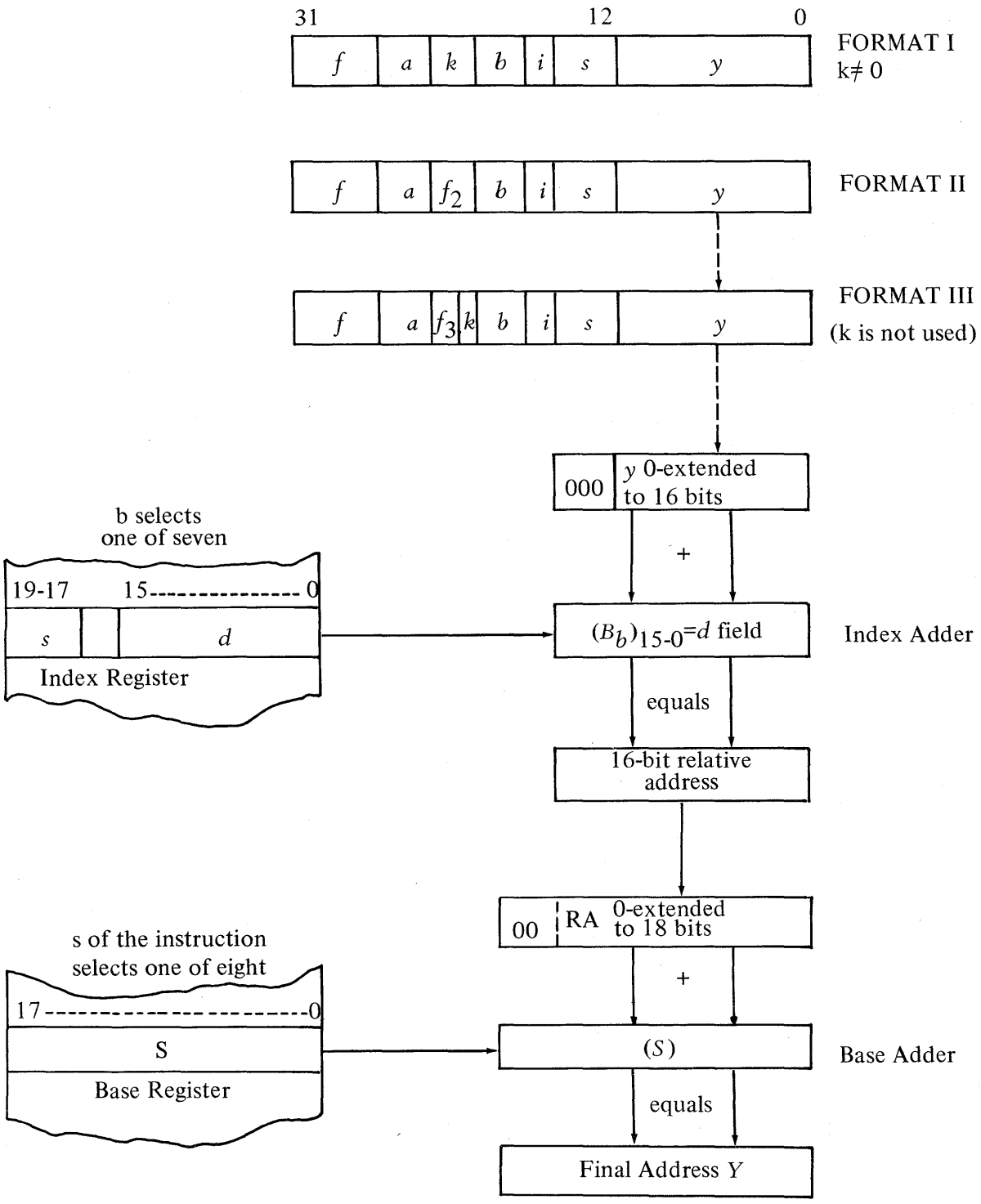


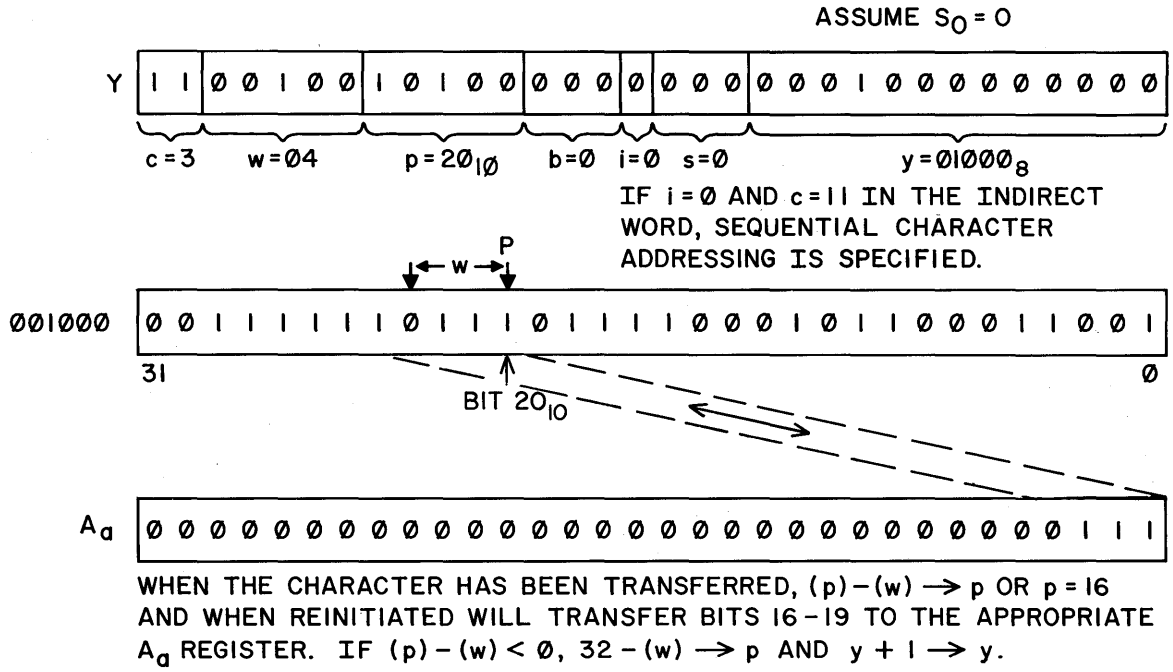
Figure 17. Direct-Address Generation

TABLE 9. INDIRECT WORD ADDRESS GENERATION

If Bits 31, 30 & 29 Equal	and i Equals	Designators in current indirect control word used as follows:
000	1	The next indirect word address $Y = sy + (S_b)$
001	1	The next indirect word address $Y = sy + (B_b) + (S)$ as designated by $(B_b)_{19\text{-}17}$
000	0	The operand* address $Y = sy + (S_b)$
001	0	The operand* address $Y = sy + (B_b) + (S)$ as designated by $(B_b)_{19\text{-}17}$
10X	1	The next indirect word address is $Y = y + (B_b) + (S_s)$
01X	1	The next indirect word address is $Y = y + (B_b) + (S_s)$
11X	1	The next indirect word address is $Y = y + (B_b) + (S_s)$
10X	0	The operand* address $Y = y + (B_b) + (S_s)$
01X	0	The address of the single character operand defined by w and p is $Y = y + (B_b) + (S_s)$
11X	0	The address of the sequential character operand defined by w and p is $Y = y + (B_b) + (S_s)$. Then if $p\text{-}w \geq 0$, $p\text{-}w \longrightarrow p$ and $y \longrightarrow y$ if $p\text{-}w < 0$, $32\text{-}w \longrightarrow p$ and $y + 1 \longrightarrow y$ The updated indirect control word is stored back into main memory for the next execution.

*The operand is defined by the function code and in Format I instructions the k designator.

SEQUENTIAL CHARACTER ADDRESSING



WHEN THE CHARACTER HAS BEEN TRANSFERRED, (p) − (w) → p OR p = 16 AND WHEN REINITIATED WILL TRANSFER BITS 16-19 TO THE APPROPRIATE A_a REGISTER. IF (p) − (w) < 0, 32 − (w) → p AND y + 1 → y.

SINGLE CHARACTER INDIRECT ADDRESSING

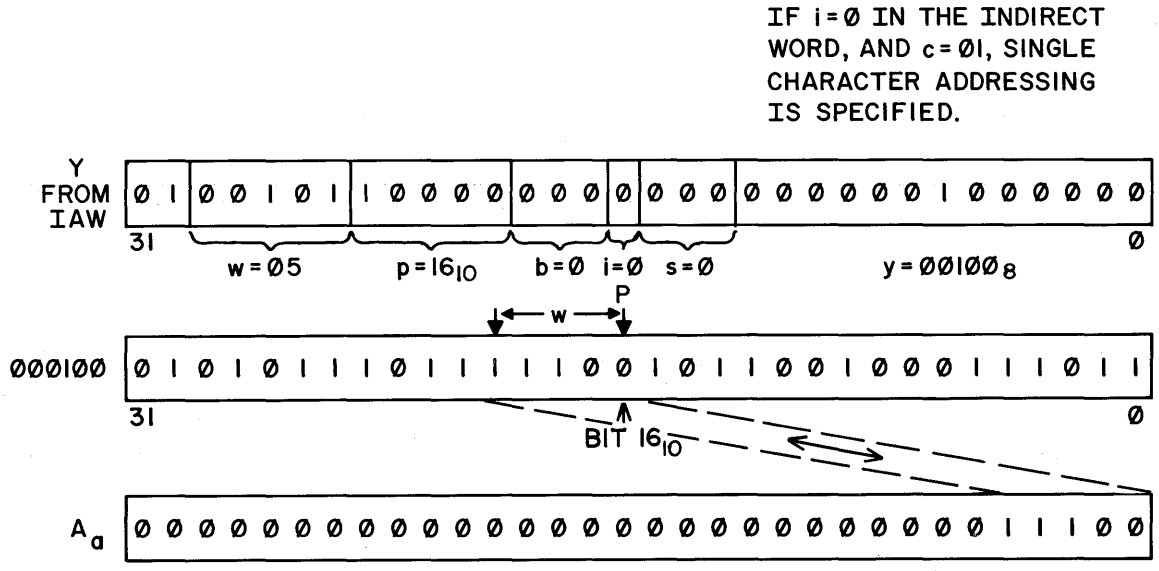


Figure 18. Examples of Indirect Addressing

Indirect Addressing – When an instruction specifying indirect addressing is executed in the repeat mode, the indirect reference (and its subsequent cascading) applies only for the first execution. The resultant final address, from the first execution in the repeat mode, will then be modified (as defined by the repeat instruction) for each subsequent execution. If the repeat mode is interrupted, the indirect addressing (and cascading) will be applied again only to the first execution after returning from the interrupt routine.

Character Addressing – Single-character addressing functions normally in the repeat mode. The character operand is located in the same bit positions in all memory words addressed. If the repeat sequence is interrupted, the indirect addressing and character definition sequence is applied again to the first execution after returning from the interrupt routine.

Sequential character addressing in the repeat mode will operate as single-character addressing throughout the repeat mode if it is allowed to complete without interruption. If it is interrupted, or upon completion of the repeat mode sequence, the p-designator of the indirect control word will be modified by w, y will be advanced if $p\text{-}w < 0$, and the updated indirect control word stored back in memory for the next execution after interrupt return or for the next complete repeat sequence.

INTERRUPT CODE GENERATION

Interrupt codes are generated by both the IOC and the processor. Code words associated with a processor-detected interrupt are formed in the central processor, and those associated with an IOC-detected interrupt are formed in the IOC. The IOC generates an 8-bit interrupt code for each interrupt request; the processor honoring the interrupt then assigns an additional 2-bit code (bits 9 and 8) that identifies the interrupting IOC. Table 10 lists the interrupts, with priorities, codes and their origin.

Interrupt status code requests are sent on the interface lines between the processor and IOC when an interrupt generated by the IOC is honored.

Whenever an IOC generates a Class I or Class III *Illegal Instruction, Monitor Clock,* or *Central Processor* interrupt it sends an interrupt request to all processors. Monitor interrupt (Class III) requests are sent to the central processors that allow Monitor interrupts on the interrupting channel (processor-program controlled). The processor which first responds to the interrupt request is sent the interrupt code having the highest priority within the initiating class presently not locked out. Because this could be other than the interrupt that generated the request, circuits in the IOC hold queued interrupts, pending timely response by a processor. A processor, responding to an IOC interrupt request that has already been acknowledged by another processor, will be sent a NO OPERATION signal. It then continues its operation without changing state.

Interrupts generated by the central processor must be integrated with those generated by an IOC in order to resolve all in an orderly manner. The central processor therefore processes interrupts according to Table 10 .

INPUT-OUTPUT CONTROLLER (IOC)

Priorities

Operations requested of the IOC are honored and executed according to priorities arranged in two groups. Buffered requests are those received on the input/output channels; non-buffered requests are initiated by the processor, or via controls within the IOC such as the RTC and channel-associated command chain. Table 15 lists these two groups.

If requests from both groups exist at the same time, the IOC alternates execution between groups. After a request has been honored, the request priority sequence is reinitiated at its highest state.

Program Chains

A program chain is associated with each channel function (data input, data output, external function transfer, external interrupt). Chains are formed, as required, from the 15 instructions in the IOC repertoire. The sequencing through chains and the transfers of information (data, interrupt codes, and commands to external devices) are

TABLE 10. INTERRUPT STATUS CODES

Class	Priority: Processor, Within Class	Priority: IOC	Interrupt	Interrupt Status Code: Status Code Bits* 9 8 7 6 5 4 3 2 1 0	Interrupt Status Code: Generated By	Class Action Taken When Processed: Store	Class Action Taken When Processed: Load
I	1		CP-Operand Memory Resume	0 0 M M M M 0 0 0 0	CP	(ASR)→141	If Power Tolerance
I	2		CP-IOC Command Resume	K K 0 0 0 0 0 0 0 1	CP	(ISC)→142	Interrupt: (140)→P
I	3		CP-Instruction Memory Resume	0 0 M M M M 0 0 1 0	CP	(P) →143	If not Power Tolerance:
I	4		CP-IOC Interrupt Code Resume	K K 0 0 0 0 0 0 1 1	CP	Set ASR bit positions:	(NDRO O)→P
I	5	1	IOC Memory Resume (Queue)	K K M M M M 1 0 1 0	IOC	19,14-9,7 (bootstrap)	
I	6	2	Intercomputer Timeout (Queue)	K K C C C C 1 0 1 1	IOC	Clear ASR bit pos. 6-0	
I	7		Power Tolerance (Queue)	0 0 0 0 0 0 1 1 1 1	CP		
II	1		Inter Processor Interrupt (Queued)	0 0 0 0	CP	(ASR)→145	If AUTO REC switched,
II	2		Floating Point Error	0 0 0 1	CP	(ISC)→146	if illegal instruction,
II	3		Illegal Instruction	0 0 1 0	CP	(P) →147	and if BOOTSTRAP switch
II	4		Privileged Instruction Error	0 0 1 1	CP		at 0: (NDRO 1)→P
			Not Assigned	0 1 0 0	CP	Set ASR bit positions:	at 1: (NDRO 2)→P
II	5		Operand Breakpoint	0 1 0 1	CP	18,13-9,7 (bootstrap)	at 2: (NDRO 3)→P
II	6		Operand Read or Indirect Addressing	0 1 1 0	CP	Clear ASR bit positions 6-0	
			Not Assigned	0 1 1 1	CP		
			Not Assigned	1 0 0 0	CP		If not as above:
II	7		Operand Write	1 0 0 1	CP		(144)→P
II	8		Operand Limit	1 0 1 0	CP		
II	9		Instruction Breakpoint	1 0 1 1	CP		
			Not Assigned	1 1 0 0	CP		
II	10		Instruction Execute	1 1 0 1	CP		
II	11		Instruction Limit	1 1 1 0	CP		
II	12		Monitor Clock (Queued)	1 1 1 1	CP		
III	1	3	IOC Illegal CAR Instruction	K K 0 0 P P 0 0 0 0	IOC	(ASR)→151	
III	2	4	IOC Illegal Chain Instruction	K K C C C C 0 1 F F	IOC	(ISC)→152	
III	3	5	IOC-Monitor Clock Interrupt	K K 0 0 0 0 1 0 1 0	IOC	(P) →153	
III	4	6	IOC-CP Interrupt	K K 0 0 0 0 1 1 1 1	IOC		(150)→P
III	5	7	IOC External Interrupt Monitor	K K C C C C 1 1 0 0	IOC	Set ASR bit positions: 17,12-9	
III	6	8	IOC External Function Monitor	K K C C C C 1 1 0 1	IOC	Clear ASR bit positions: 6-0	
III	7	9	IOC Output Data Monitor	K K C C C C 1 1 1 0	IOC	No change: ASR bit 7	
III	8	10	IOC Input Data Monitor	K K C C C C 1 1 1 1	IOC		
IV			Executive Return	(16-Bit code assigned through program design)		(ASR)→155	
						(ISC)→156	
						(P) →157	(154)→P
						(Set ASR bit positions: 16,11-9	
						Clear ASR bit positions: 6-0	
						No change: ASR bit 7	

*Definitions:
- MMMM = Memory Bank $(0\text{-}17)_8$
- CCCC = IOC Channel No. $(0\text{-}17)_8$
- KK = IOC No. (0-3) - Generated by CP that honors the interrupt
- PP = CP No. (0-2)
- FF = Function
- 00 = External Interrupt
- 01 = External Function
- 10 = Output
- 11 = Input

TABLE 11 IOC CONTROL MEMORY BUFFER CONTROL WORD FORMAT

FORMAT	55——38	37/36	35	34	33 32	31——18	17——0
	CAP	*k*	*c*	*m*	*byte*	*FA*	*CA*

0 - 17	*CA*	Current Address, the main memory address of the next word transferred, advanced each time a whole computer word is transferred.
31 - 18	*FA*	Final Address, compare bits of the last word in the buffer. When the current address is advanced to equal the final address, the buffer transfer terminates.
33 - 32	*byte*	Byte, identifying the partial-word position in memory involved in the transfer.
34	*m*	Monitor Flag, when set, the buffer transfer termination generates an interrupt.
35	*c*	Chain Flag, when set, indicates that another command follows when this command or buffer transfer terminates.
37 - 36	*k*	Partial-Word Designator, indicating the size of partial-word transfers (whole, half, quarter or suppressed data) and dictating byte updating.
55 - 38	*CAP*	Command Address Pointer, the address of the next command in the chain. When the current command or buffer transfer terminates with the Chain Flag set, the next IOC instruction is read from this address.

controlled by the various fields in associated control words in the IOC control memory (shown in Table 11). These words are selected and loaded according to the commands executed by the IOC. Format and field definitions for the IOC commands words are shown in Table 12.

Processor - IOC Non-Buffered Requests

Monitor Interrupt Control, IOC Monitor Clock Control and *Clock Read* instructions, when executed by the processor, are placed on the operand bus with a code addressing the specified IOC. The IOC accepts the instruction elements via its CIR, translates the functions described, and enters or reads the registers specified (RTC, Mon. clock, Interrupt Lockout Register). RTC and Monitor Clock data, when read, are transferred to the processor on the operand bus.

The processor instructions so handled are these:

07 1 *Allow Enable Interrupt*
07 2 *Prevent Enable Interrupt*
07 3 *Load IOC Monitor Clock*
07 4 *Initiate I/O*
77 0 *Store IOC Monitor Clock in A*
77 1 *Store Real Time Clock in A*

Externally Specified Index (ESI) Buffer Transfers

Any input and output parallel channel pair may operate in the ESI mode for input/output data, external interrupt, and external function transmission. The program must set the channel active, with or without a monitor, when such operation is desired. For output data or external function operation, the Externally Specified Index (address) is placed on the low-order 16 input data

lines and the OUTPUT or EXTERNAL FUNCTION REQUEST (respectively) is set for that channel. For input data and external interrupt operations, the data are placed on the 16 high-order lines, the index placed on the 16 lower order lines, and the INPUT DATA or EXTERNAL INTERRUPT REQUEST is set for that channel. In each case, the index specifies an address in the lower 65K locations of main memory where the buffer control word, as described in Figure 19, is stored.

During the execution of an ESI-controlled transfer, the IOC does not interpret the normal control memory partial-word designator and the byte fields. The terminal address designator, initial address designator and the partial word designator are processed and stored back in main memory. The IOC responds with a word transfer and appropriate acknowledge signal as in a normal buffer transfer.

Bit	31, 30, 29	28 ——— 18	17 ——— 0
		Final Compare Address*	Current Address
Partial Word Transfer Designator			
	0 0 1	Quarter Word (Bits 31-24)	Q4
	0 1 1	Quarter Word (Bits 23-16)	Q3
	1 0 1	Quarter Word (Bits 15- 8)	Q2
	1 1 1	Quarter Word (Bits 7- 0)	Q1
	0 1 0	Half Word (Bits 31-16)	H2
	1 1 0	Half Word (Bits 15- 0)	H1
	1 0 0	Whole Word - Data justified right in memory.	
	0 0 0	Suppress Data - No data transferred.	

*Maximum ESI buffer size is 2048 addresses.

Figure 19. ESI Buffer Control Word Format

Since many index addresses can be utilized by a peripheral device to designate unique buffers for individual subchannels, upon buffer transfer termination the IOC stores the index address in the low-order 16 bits of the memory word specified by y in the instruction establishing the buffer. It is therefore available to the program for subchannel identification.

Externally Specified Address (ESA) Word Transfers

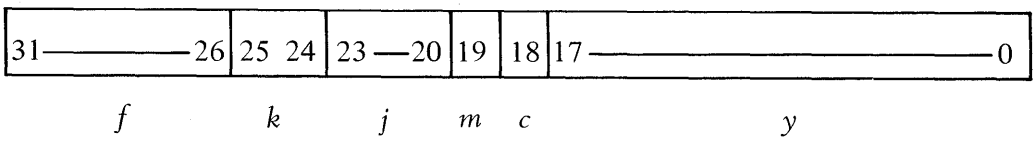
Any input and output channel pair may operate in the ESA mode to retrieve or store data or functions on a word-by-word basis. The peripheral device specifies an address that must be in the lower 65K words of storage for each input or output word requested. An input channel activated by the IOC program will respond to an ESA input-data or external-interrupt request, and an activated output channel will respond to an ESA output-data or external-function request.

For output, the peripheral device places the address on the lower 16 lines of the input cable and signals via the OUTPUT DATA or EXTERNAL FUNCTION REQUEST line on the output cable. The IOC responds according to internal priorities, and reads the memory address specified; it then places the 32-bit contents on the output cable and signals via the OUTPUT DATA or EXTERNAL FUNCTION ACKNOWLEDGE signals, respectively.

For input, the peripheral device places data or interrupt information on the high-order 16 lines, places the address on the lower 16 lines and signals via the INPUT or EXTERNAL INTERRUPT REQUEST line, respectively, of the input cable. The IOC responds according to internal priority, stores the 32 bit contents (data and address) at the address specified, and signals via the INPUT ACKNOWLEDGE line. When responding to an EXTERNAL INTERRUPT REQUEST, the IOC also sets INTERRUPT ENABLE line.

TABLE 12. IOC COMMAND WORD FORMAT

FORMAT



Field	Definition
f	Function code of the command
k	Partial word designator: specifies whole, half, quarter or suppressed word transfer
j	Channel number designator
m	Monitor designator; if set, a monitor interrupt is requested at buffer termination
c	Chain flag; if set, another instruction follows
y	Address designator; specifies an 18-bit address of the operands.

NOTES:

The Instruction Repertoire is detailed in a later section of this book.
IOC command codes 10 through 16 interpret k as indicated by Tables 13 and 14.
IOC command codes 22 and 24 used the combined k and j fields (kj) to address control memory locations.
IOC command codes 17, 25, and 26 use the combined k and j fields (kj) to identify a bit position in main memory address y.
IOC command codes 20, 23, and 27 do not interpret the k, j, and m fields; they are not used.

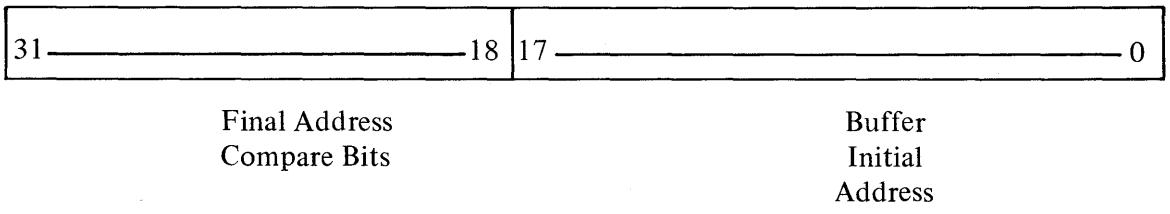


Figure 20. Main Memory Buffer Control Word Format

TABLE 13. *k*-FIELD INTERPRETATION: IOC COMMAND CODES 10, 11 & 13 AND ASSOCIATED CONTROL MEMORY WORDS

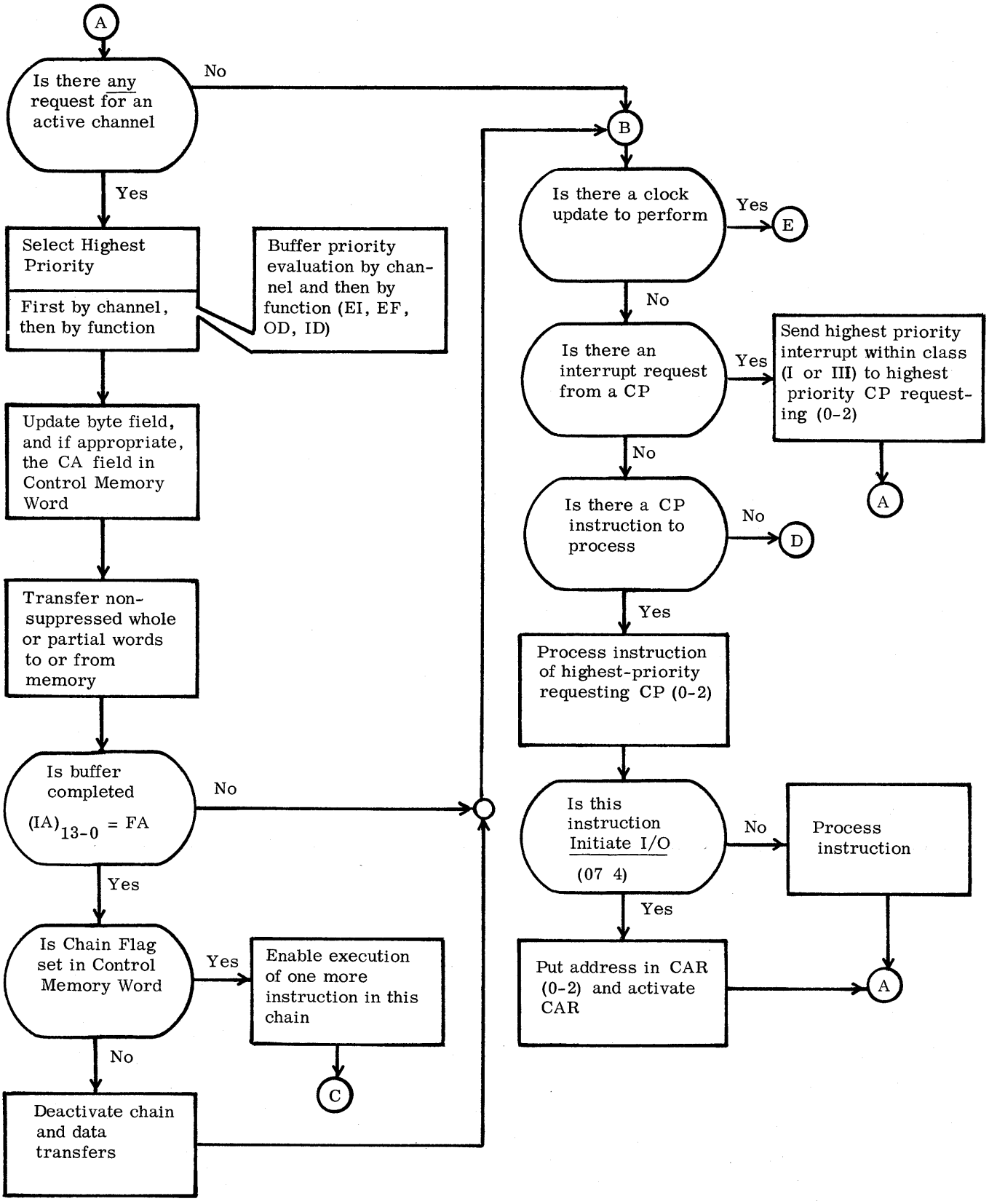
k Value	Transfer Description	Channel Bits Transferred	Memory Word Bits Transferred	Change Current Address	BCW Byte Bits
0	Data Suppression (32 bits)	31-0 Output: all zeros Input: →	31-0 ← no data	once for each transfer	–
1	1st quarter word 2nd quarter word 3rd quarter word 4th quarter word (8 bits)	7-0 ↔ 31-24 7-0 ↔ 23-16 7-0 ↔ 15- 8 7-0 ↔ 7- 0		once for every four transfers	0 1 2 3
2	1st half word 2nd half word (16 bits)	15-0 ↔ 31-16 15-0 ↔ 15- 0		once for every two transfers	0 1
3	whole word (32 bits)	31-0 ↔ 31-0		once for each transfer	–

TABLE 14. *k*-FIELD INTERPRETATION: IOC COMMAND CODES 12, 14, 15 & 16

k Value	Code 12 Transfer External Functions	Code 14 Terminate Buffer Transfer for	Code 15 Request Buffer Termination Monitor Interrupt for	Code 16 Activate Command Chain Using CMW assigned to
0	Force one word	Data input	Data input	Data input
1	Transfer one word when requested	Data output	Data output	Data output
2	Transfer many words as requested	External Function output	External Function output	External Function output
3	Not used	External Interrupt input	External Interrupt input	External Interrupt input

TABLE 15. IOC REQUEST PRIORITIES

NONBUFFERED REQUESTS		
REQUEST PRIORITY	REQUEST TITLE	ACTION WHEN PROCESSED
1	Clock Request	Decreases the IOC monitor clock count by 1 and increases the real-time clock count by 1
2	Processor instruction for IOC and interrupt status code requests	Performs the function as commanded.
2a	Processor No. 1 Request	
2b	Processor No. 2 Request	
2c	Processor No. 3 Request	
3	Processor Command Address request	Performs the function as commanded.
3a	Processor No. 1 Request	
3b	Processor No. 2 Request	
3c	Processor No. 3 Request	
4	Chain commands (channel associated)	Performs the function as commanded according to normal channel priority
BUFFERED REQUESTS Channel dependent buffer request (includes EI, EF, outputs, and input)		
1a	External interrupt request (occurs when an external device sets the EXTERNAL INTERRUPT REQUEST line)	Performs a one-word External Interrupt Code transfer as governed by the relevant CMR Buffer Control Word.
1b	External Function Request (occurs when an external device sets the EXTERNAL FUNCTION REQUEST line)	Performs a one-word External Function Code transfer as governed by the relevant CMR buffer Control Word.
1c	Output Data Request (occurs when an external device sets the OUTPUT DATA REQUEST line)	Performs a one-word data output transfer as governed by the CMR buffer control word.
1d	Input Data Request (occurs when an external device sets the INPUT DATA REQUEST line)	Performs a one-word input data input transfer as governed by the relevant CMR Buffer Control Word.


A
Is there any request for an active channel
No
Yes
Select Highest Priority
First by channel, then by function
Buffer priority evaluation by channel and then by function (EI, EF, OD, ID)
Update byte field, and if appropriate, the CA field in Control Memory Word
Transfer non-suppressed whole or partial words to or from memory
Is buffer completed (IA)13-0 = FA
No
Yes
Is Chain Flag set in Control Memory Word
Yes
Enable execution of one more instruction in this chain
C
No
Deactivate chain and data transfers
B
Is there a clock update to perform
Yes
E
No
Is there an interrupt request from a CP
Yes
Send highest priority interrupt within class (I or III) to highest priority CP requesting (0-2)
A
No
Is there a CP instruction to process
No
D
Yes
Process instruction of highest-priority requesting CP (0-2)
Is this instruction Initiate I/O (07 4)
No
Process instruction
Yes
Put address in CAR (0-2) and activate CAR
A

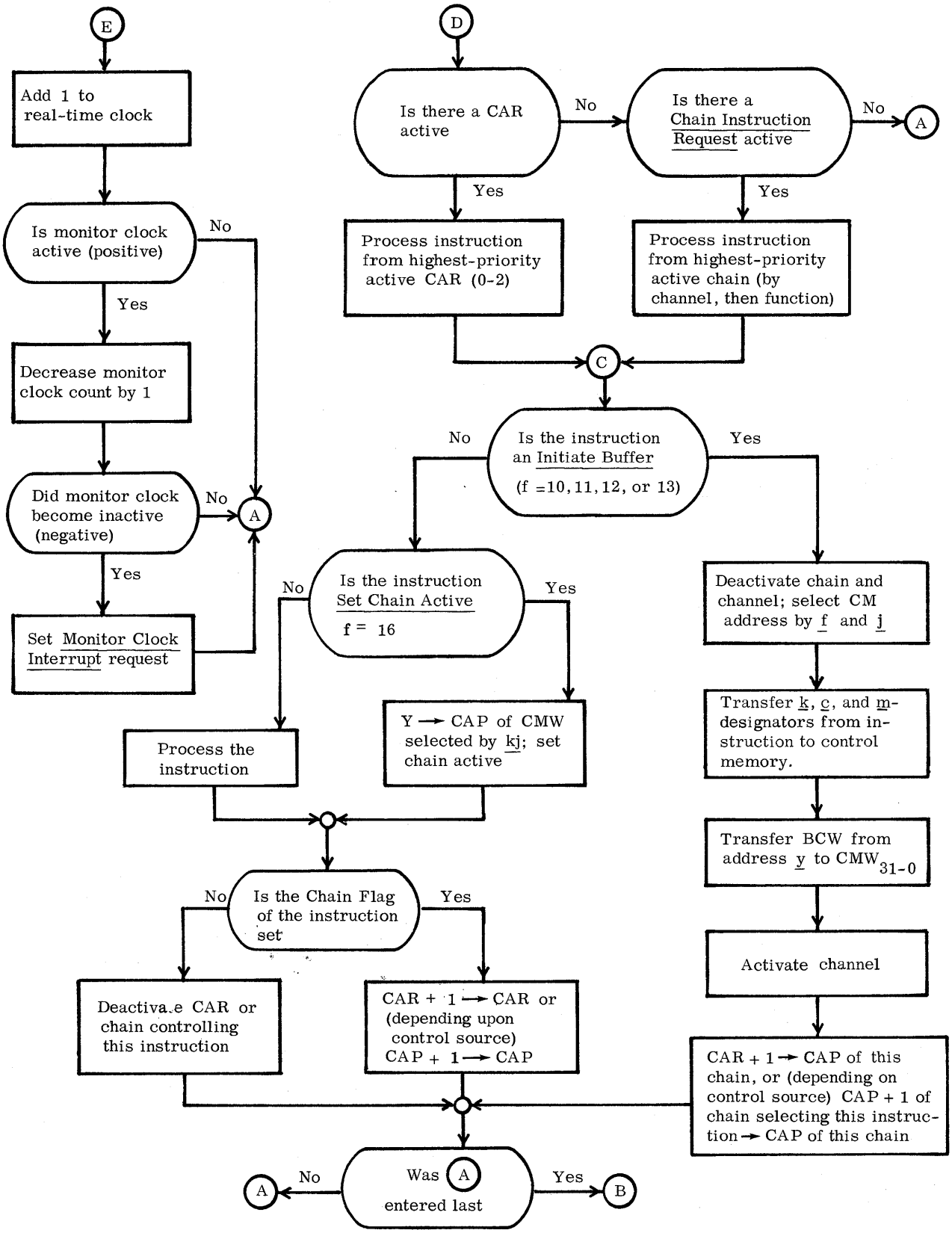


Figure 21. Sequences of IOC Processing

REPERTOIRE OF INSTRUCTIONS

Conventions and Abbreviations

The following symbols and definitions help to describe instructions in the AN/UYK-7 repertoire.

SYMBOL	DEFINITION
( )	Contents of the quantity within parentheses
( )'	Complement of the quantity within parentheses
| |	Absolute value
:	Compare
x or ·	Multiply
÷	Divide
-	Minus
+	Plus
=	Equal
≠	Not equal
>	Greater than
≥	Greater than or equal to
<	Less than
≤	Less than or equal to
⊙	Logical AND or logical product defined as:
⊕	Inclusive OR or logical sum defined as:
⊕̄	Exclusive OR or logical difference defined as:
a	Instruction field usually designating an accumulator or index register.
A_a	Accumulator designated by the *a* field.
A_b	Accumulator designated by the *b* field.
af_4	Instruction field designating the combined *a* and f_4 fields use to specify a control memory register.
ak	Instruction field designating the combined *a* and *k* fields used to specify a bit position or a control memory register.
b	Instruction field designating an index or accumulator register.
B_a	Index register designated by the *a* field.
B_b	Index register designated by the *b* field.

	0	1
0	0	0
1	0	1

	0	1
0	0	1
1	1	1

	0	1
0	0	1
1	1	0

SYMBOL	DEFINITION
c	Instruction field designating indirect addressing word type; or the chain flag in input/output controller commands.
C	Input/output channel.
CA	Capable of indirect word character addressing; Current address in CMR
Cj	Input/output channel designated by the *j* field.
CD	Hardware compare designator.
CMR	Control memory register.
DSW	Designator status words.
EF	External function.
EI	External interrupt.
f	Instruction field designating the major function code.
f_2, f_3, f_4	Instruction fields designating sub-function code.
i	Instruction field designating indirect addressing.
I/O	Input/output.
IOC	Input/output controller.
j	Instruction field designating an I/O channel.
k	Instruction field partial word designator or I/O function.
kj	Instruction field designating the combined *k* and *j* fields used to specify a bit position or control memory register in IOC instructions.
m	Instruction field designating a shift count; monitor flag in an IOC command.
n	Used as a subscript indicating a bit position; for example, $(A_a)_n$.
NI	Next instruction.
OD	Overflow designator.
p	Indirect word bit position designator.
P	Program Address Register.
PI	Privileged instruction executable only when the processor is in the Interrupt (executive) State.
R	Capable of being repeated.
RTC	Real-time clock.
s	Instruction field designating a base register.
S	Base register.
S_a	Base register designated by the *a* field.
S_b	Base register designated by the *b* field.
S_s	Base register designated by the *s* field.
sy	Instruction field representing *s* and *y* fields in combination.
U	*U*-Register (program control register).
UF	ULTRA/32 assembler format (defined on coding card)
w	Indirect word character length designator.
y	Instruction operand field designating an address or value.
Y	Effective address formed by $y + (B_b) + (S_s)$.
$\underline{Y}$	Effective operand as qualified by *k* (or *p* and *w* when applicable).

This section lists the instructions for the processor in the following format.

- Instruction Word format designation, assembler code, octal code, instruction name, symbolic summary, and execution time in microseconds
- Text defining the instruction in detail
- Notes and/or examples, if any

II	OR	01 0	Inclusive OR	$(Y) \oplus (A_a) \longrightarrow A_a$	1.5

Set each bit position in A_a where the corresponding bit position in (Y) is set. Other bits in A_a remain unchanged.

II	SC	01 1	Selective Clear	$(A_a) \odot (Y)' \longrightarrow A_a$	1.5

Clear each bit position in A_a where the corresponding bit position in (Y) is set. Other bits in A_a remain unchanged.

II	MS	01 2	Selective Substitute	$(Y)_n \longrightarrow (A_{a+1})_n$ when $(A_a)_n = 1$	1.5

For each bit position set in A_a substitute the corresponding bit from (Y) for the corresponding bit in A_{a+1}, leaving the remaining bits in A_{a+1}, and all bits in A_a unchanged.

II	XOR	01 3	Exclusive OR	$(Y) \bar{\oplus} (A_a) \longrightarrow A_a$	1.5

Complement each bit in A_a where the corresponding bit position in (Y) is set. Other bits in A_a remain unchanged.

II	ALP	01 4	Add Logical Product	$(A_{a+1}) + (Y) \odot (A_a) \longrightarrow A_{a+1}$	1.5

Add the logical product of (A_a) and (Y) to (A_{a+1}). Store the result in A_{a+1}.

II	LLP	01 5	Load Logical Product	$(Y) \odot (A_a) \longrightarrow A_a$	1.5

Form the logical product of (Y) and (A_a). Store the result in A_a.

II	NLP	01 6	Subtract Logical Product	$(A_{a+1}) - (Y) \odot (A_a) \longrightarrow A_{a+1}$	1.5

Subtract from (A_{a+1}) the logical product of (Y) and (A_a). Store the result in A_{a+1}.

II	LLPN	01 7	Load Logical Product Next	$(Y) \odot (A_a) \longrightarrow A_{a+1}$	1.5

Form the logical product of (Y) and (A_a) and store the result in A_{a+1}. (A_a) remain unchanged.

II	CNT	02 0	Count Ones	Number of 1-bits in (Y) $\longrightarrow A_a$	7.5+

Count the number of 1-bits in (Y) and enter the count in A_a.

+Execution time is independent of overlap operation

II	XR	02 2	Execute Remote	$(Y) \longrightarrow U$	1.5

Execute the whole or upper-half word instruction at address Y without changing the count in the program address register.

II	XRL	02 3	Execute Remote Lower	$(Y)_{15\text{-}0} \longrightarrow U$	1.5

Execute the lower half-word instruction at address Y without changing the count in the program address register.

II	SLP	02 4	Store Logical Product	$(A_a) \odot (A_{a+1}) \longrightarrow Y$	1.5

Form the logical product of (A_a) and (A_{a+1}). Store the result in Y; leave the A registers unchanged.

II	SSUM	02 5	Store Sum	$(A_a) + (A_{a+1}) \longrightarrow Y$ and A_{a+1}	2.0

Add (A_a) and (A_{a+1}). Store the sum in Y and A_{a+1}. (A_a) remains unchanged.

II	SDIF	02 6	Store Difference	$(A_{a+1}) - (A_a) \longrightarrow Y$ and A_{a+1}	2.0

Subtract (A_a) from (A_{a+1}). Store the result in Y and A_{a+1} and leave (A_a) unchanged.

II	DS	02 7	Double Store A	$(A_{a+1},A_a) \longrightarrow Y+1,Y$	3.0

Form a double-word operand with (A_{a+1}) at the high-order half and (A_a) at the low-order half. Store it in Y+1 and Y, with Y+1 containing the high-order half.

II	ROR	03 0	Replace Inclusive OR	$(Y) \oplus (A_a) \longrightarrow Y$ and also A_a	2.5

Set each bit position in A_a where the corresponding bit position in (Y) is set. Other bits in A_a remain unchanged. Store the result in both A_a and Y.

II	RSC	03 1	Replace Selective Clear	$(A_a) \odot (Y)' \longrightarrow Y$ and also A_a	2.5

Clear each bit position in A_a where a corresponding bit position in (Y) is set. Other bits in A_a remain unchanged. Store the result in both A_a and Y.

II	RMS	03 2	Replace Selective Substitute	$(Y)_n \longrightarrow A_{a+1\ n}$ when $(A_a)_n = 1$; then $(A_{a+1}) \longrightarrow Y$	2.5

Substitute bits from (Y) for corresponding bits in (A_{a+1}) where bits in (A_a) contain 1's. Store the results in both A_{a+1} and Y, leaving (A_a) unchanged.

II	RXOR	03 3	Replace Exclusive OR	$(Y) \bar{\oplus} (A_a) \longrightarrow Y$ and also A_a	2.5

Complement each bit in A_a where the corresponding bit position in (Y) is set. Other bits in A_a remain unchanged. Store the result in both Y and A_a.

II	RALP	03 4	Replace A + Logical Product	$(A_{a+1})+(Y) \odot (A_a) \longrightarrow Y$ and also A_{a+1}	2.5

Add the logical product of (Y) and (A_a) to (A_{a+1}). Store the results in both Y and A_{a+1}.

II	RLP	03 5	Replace Logical Product	(Y) $\odot$ (A_a) $\longrightarrow$ Y and also A_{a+1}	2.5

Form the logical product of (Y) and (A_a). Store the result in Y and A_{a+1}.

II	RNLP	03 6	Replace A - Logical Product	(A_{a+1}) - (Y) $\odot$ (A_a) $\longrightarrow$ Y and also A_{a+1}	2.5

Subtract from (A_{a+1}) the logical product of (Y) and (A_a). Store the result in Y and A_{a+1}.

II	TSF	03 7	Test and Set Flag	If $(Y)_{31}$=0. 1 $\longrightarrow$ Y_{31} and set CD EQUAL If $(Y)_{31}$=1 leave it set and set CD UNEQUAL	2.5

Test bit 31 of (Y). If it is 0, set bit 31 of (Y) and set the compare designator to EQUAL (set bit position 2 of ASR). If bit 31 of (Y) is 1, leave it unchanged and set the compare designator to UNEQUAL (clear bit position 2 of ASR).

II	DL	05 0	Double Load A	(Y+1, Y) $\longrightarrow$ A_{a+1}, A_a	3.0

Load the double-length word, from (Y+1) at the high-order half and (Y) at the low-order half, in A_{a+1}, holding the high-order half, and A_a the low-order half.

II	DA	05 1	Double Add A	(Y+1, Y) + (A_{a+1}, A_a) $\longrightarrow$ A_{a+1},A_a	3.0

Add the double-length word, in (A_{a+1}) at the high-order half, and (A_a) at the low-order half, to the double-length word from (Y+1) and (Y), which has (Y+1) as the high-order half. Store the result in A_{a+1} and A_a, with A_{a+1} holding the high-order half.

II	DAN	05 2	Double Subtract A	(A_{a+1},A_a) - (Y+1,Y) $\longrightarrow$ A_{a+1},A_a	3.0

From the double-length word in (A_{a+1}) at the high-order half and (A_a) at the low-order half, subtract the double-length word from (Y+1) and (Y), which has (Y+1) at the high-order half. Store the result in A_{a+1} and A_a, with A_{a+1} holding the high-order half.

II	DC	05 3	Double Compare	(A_{a+1},A_a) : (Y+1,Y); Set CD	3.0

Compare the double-length word in (A_{a+1}) at the high-order half, and (A_a) at the low-order half, with the double-length word from (Y+1) and (Y), which has (Y+1) at the high-order half. Set the comparison designator as follows:

If (A_{a+1},A_a) = (Y+1,Y) set ASR bit 2 (EQUAL)
If (A_{a+1},A_a) $\neq$ (Y+1,Y) clear ASR bit 2 (UNEQUAL)
If (A_{a+1},A_a) $\geq$ (Y+1,Y) set ASR bit 1 (GREATER THAN OR EQUAL TO)
If (A_{a+1},A_a) < (Y+1,Y) clear ASR bit 1 (LESS THAN)

Both ASR bits will reflect comparison results.

II LBMP 05 4 Load Base and Memory Protection $(Y)_{17-0} \longrightarrow S_a$ (Task); $(Y+1)_{20-0} \longrightarrow SPR_a$ $s \longrightarrow SIR_{a19-17}$; $y+(B_b)_{15-0} \longrightarrow SIR_{a15-0}$ 5.75

Load the task base register specified by *a* with the low-order 18 bits of (Y). Load the associated storage protection register (also specified by *a*) with the low-order 21 bits of (Y+1). Load the associated segment identifications register (specified by *a*) with the *s* value in bits 19-17 and the relative address quantity $y+(B_b)_{15-0}$ in bits 15-0.

RESTRICTIONS: The relative address quantity must be an even number or an *Illegal Instruction* Interrupt will be generated. If in the Task State, three conditions must prevail, or a *Privileged Instruction Error* Interrupt will be generated:

1) Bit 8 (Load Base Enable) of the Active Status Register must be set.
2) The s-field must be 7.
3) The a-field must not be 7.

Table 16 illustrates the Storage Protection Register format.

$(Y+1)$ or (A_{a+1})	(Y) or (A_a)	
31 ——— M ——— 0	31 —Sign— 16	15 ——C—— 0
One's-Complement Mantissa (M) $1.0 > \|M\| \geq ½$ (Normalized)	One's-Complement Characteristic (C) $+32767 \geq C \geq -32767$	

Mantissa is normalized so that $(Y+1)_{31} \neq (Y+1)_{30}$, and $(A_{a+1})_{31} \neq (A_{a+1})_{30}$

Figure 22. Normalized Floating-Point Word Format

II FA 06 0 Floating-Point Add 6.25^+

Operands must be normalized before entry. Add the floating-point number in Y+1 and Y to the floating-point number in A_{a+1} and A_a. Leave the normalized sum mantissa in A_{a+1} and the adjusted characteristic in A_a.

II FAN 06 1 Floating-Point Subtract 6.25^+

Operands must be normalized before entry. Subtract the floating-point number in Y+1 and Y from the floating-point number in A_{a+1} and A_a. Leave the normalized difference mantissa in A_{a+1} and the adjusted characteristic in A_a.

$^+$Execution time is independent of overlap operation

TABLE 16. STORAGE PROTECTION FORMAT AND EXAMPLES

20	19	18	17	16	15 — 0
I	*OR*	*OW*	*IA*	*IR*	*R*

R: Maximum allowable displacement; number of words protected minus 1. Final address=$(S) + R$

IR: When set, the interrupt group of B and S registers specified in the indirect control word are used for addressing.

IA: When set, indirect addressing is permitted.

OW: When set, storing operands is permitted.

OR: When set, reading operands is permitted.

I: When set, executing instructions is permitted.

Assume: 1. Task State

2. $(S)_3$ =

0	1	5	0	0	0
000	001	101	000	000	000

3. $(SPR)_3$ =

I	*OR*	*OW*	*IA*	*IR*	*R*					
1	1	0	0	0	0	000	100	000	000	000

4. $(B)_2$ =

0	0	3	0	0	0
0	000	011	000	000	000

Lower Limit = $(S)_3$ = 015000 (octal)
Upper Limit = $(S)_3$ + R=021000 (octal)

Type of Reference	Net Address Referenced	Within Limits?	SPR Enabled?	Ref. Allowed?	Class II Interrupt Generated
Instruction, P_s=3, P_d=001000	016000	Yes	Yes I=1	Yes	None
Instruction, P_s=3, P_d=005000	022000	No	Yes I=1	No	Instruction limit
Operand Read *s*=3, *b*=0, *i*-0, *y*=002000	017000	Yes	Yes OR=1	Yes	None
Operand Read *s*=3, *b*=2, *i*=0, *y*=002000	022000	No	Yes OR=1	No	Operand limit
Operand Read *s*=3, *b*=0, *i*=1, *y*=002000	017000	Yes	No IA=0	No	Operand read or indirect addressing
Operand Write *s*=3, *b*=0, *i*=0, *y*=002000	017000	Yes	No OW=0	No	Operand Write

II FM 06 2 Floating-Point Multiply 10.0^{+}

Operands must be normalized before entry. Multiply the floating-point number in A_{a+1} and A_a by the floating-point number in Y+1 and Y. Leave the normalized product mantissa in A_{a+1} and the adjusted characteristic in A_a.

II FD 06 3 Floating-Point Divide 17.0^{+}

Operands must be normalized before entry. Divide the floating-point number in A_{a+1} and A_a by the floating-point number in Y+1 and Y. Leave the normalized quotient mantissa in A_{a+1} and the adjusted characteristic in A_a.

ROUNDING OF FLOATING POINT RESULTS

Mantissa rounding is performed (A_{a+1}) according to the status of the intermediate double-length result in the arithmetic section for add, subtract and multiply; and according to the value of the remainder in divide operations. The final sum or difference mantissa in (A_{a+1}) is rounded as follows:

1. If bit 31 of the 64 bit intermediate sum or difference equals 1 and (A_{a+1}) are positive, 1 is added to (A_{a+1}).
2. If bit 31 of the 64 bit intermediate sum or difference equals 0 and (A_{a+1}) are negative, 1 is subtracted from (A_{a+1}).
3. If not 1 or 2 above, (A_{a+1}) are not changed.
4. If overflow results in 1 or 2 above (A_{a+1}) are shifted right one place, 1 is added to the characteristic exponent in A_a and the mantissa sign bit in A_{a+1} is restored.

Rounding of a product mantissa is done before final sign correction.

1 is added to (A_{a+1}) if bit 31 of the 64 bit intermediate product equals 1; otherwise (A_{a+1}) are not changed.

Rounding of a quotient mantissa is done before final sign correction.

1. If the remainder is equal to or greater than one-half the divisor and there is no overflow, 1 is added to (A_{a+1}).
2. If bit 31 of the quotient in A_{a+1} equals 1, (A_{a+1}) are shifted right one place, $(A_{a+1})_0$ before shifting, is added to the shifted (A_{a+1}) and 1 is added to the characteristic exponent in A_a.

II FAR 06 4 Floating-Point Add with Round 6.25^{+}

Operands must be normalized before entry. Add the floating-point number in Y+1 and Y to the floating-point number in A_{a+1} and A_a. Leave the rounded and normalized sum mantissa in A_{a+1} and adjusted characteristic in A_a.

II FANR 06 5 Floating-Point Subtract with Round 6.25^{+}

Operands must be normalized before entry. Subtract the floating-point number in Y+1 and Y from the floating-point number in A_{a+1} and A_a. Leave the rounded and normalized difference mantissa in A_{a+1} and the adjusted characteristic in A_a.

$^{+}$Execution time is independent of overlap operation

II FMR 06 6 Floating-Point Multiply with Round 10.0+

Operands must be normalized before entry. Multiply the floating-point number in A_{a+1} and A_a by the floating-point number in Y+1 and Y. Leave the rounded and normalized product mantissa in A_{a+1} and the adjusted characteristic in A_a.

II FDR 06 7 Floating-Point Divide with Round 17.0+

Operands must be normalized before entry. Divide the floating-point number in A_{a+1} and A_a by the floating-point number in Y+1 and Y. Leave the rounded and normalized quotient mantissa in A_{a+1} and the adjusted characteristic in A_a.

II XS 07 0 a=0 Enter Executive State sy + $(B_b)_{15-0}$ ⟶ CMR156 4.0

Generate a Class IV *Executive Return* interrupt and transfer Y = sy +$(B_b)_{15-0}$ to the interrupt status code word location, 156, for the Executive State.

NOTE: When the interrupt is honored: (ASR) ⟶ CMR address 155
(P) ⟶ CMR address 157
ASR interrupt lockout bits remain unchanged
and (CMR154) ⟶ P

II IPI 07 0 a=1 Interprocessor Interrupt (Privileged) 4.0

Generate a Class II interrupt to all processors whose assigned number (7 through 0) corresponds to an individual bit position of Y_{7-0} that is set. Y = sy + $(B_b)_{15-0}$. Ignore a self-interrupt thus generated if bit position 15 of Y is set. Otherwise honor the self-interrupt. Bits 14-8 of Y are not used.

II AEI 07 1 Allow Enable Interrupt (Privileged) 2.0

Allow interrupt requests in the IOC specified by *a* (*a* = 0, 1, 2 or 3; 4-7 are not used) on each channel whose number corresponds to the bit position set in Y_{15-0}. Y = sy + $(B_b)_{15-0}$.

II PEI 07 2 Prevent Enable Interrupt (Privileged) 2.0

Prevent interrupt requests in the IOC specified by *a* (*a* = 0, 1, 2 or 3; 4-7 are not used) on each channel whose number corresponds to the bit position in Y_{15-0} that is set. Y = sy + $(B_b)_{15-0}$.

II LIM 07 3 Load IOC Monitor Clock Y ⟶ IOC_a Monitor Clock 3.0

Load the monitor clock in the IOC specified by *a* (*a*=0, 1, 2, or 3; 4-7 are not used) with Y. The value Y = sy + $(B_b)_{15-0}$ determines the monitor clock action. If Y is negative the clock is disabled; if zero the IOC immediately generates a Class III *IOC-CP* interrupt to all processors connected to that IOC. If Y is positive the clock count-down function is enabled.

+Execution time is independent of overlap operation

II IO 07 4 Initiate I/O $Y \longrightarrow IOC_a$ 3.5

Initiate a chain in the IOC specified by *a* (*a* = 0, 1, 2, or 3; 4-7 are not used) and transmit Y to the command address register of that IOC. Y is the address of an IOC instruction.

II IR 07 5 Interrupt Return 3.0

Return control to the processor state specified by the designator storage words. Restore the ASR and P from the designator storage words assigned to this interrupt state. Control Memory designator storage words transferred when returning from:

Class I interrupt; (141) $\longrightarrow$ ASR; (143) $\longrightarrow$ P
Class II interrupt; (145) $\longrightarrow$ ASR; (147) $\longrightarrow$ P
Class III interrupt; (151) $\longrightarrow$ ASR; (153) $\longrightarrow$ P
Class IV interrupt; (155) $\longrightarrow$ ASR; (157) $\longrightarrow$ P

II RP 07 6 Repeat 1.5

Repeat the next instruction in the program sequence (B_7) times. If $(B_7) = 0$, skip the next instruction. If $(B_7) \neq 0$, repeat the next instruction (B_7) times and decrease (B_7) by one each time the next instruction is executed until either (B_7) becomes zero or the condition specified by the *a*-designator is satisfied as follows:

TERMINATE REPEAT CONDITIONS

Special *a* value in *Repeat* instructions	Terminate repeated non-compare instructions	Terminate repeated compare instructions
0	If (A) $\neq$ 0	If CD set to $\neq$
1	If (A) = 0	If CD set to =
2	If (A) $\geq$ 0	If CD set to >
3	If (A) < 0	If CD set to $\geq$
4	Do not terminate	If CD set to <
5	If (A) is even parity on write into memory	If CD set to $\leq$
6	If (A) is odd parity on write into memory	If CD set to Outside Limits
7	Do not terminate	If CD set to Within Limits

sy is the increment or decrement to the repeated instruction address for each execution. Designators and fields in the repeat and in the repeated instructions, or associated indirect address words, are interpreted as follows:

If repeat instruction Contains:	The repeat instruction designators are interpreted as:	The repeated instruction designators are interpreted as:
b = 0 (Special)	Y=sy, a 16-bit signed constant Δ	The operand address Y=y + (B_b) + (S_s) for all *Read, Store,* and *Replace* instructions. $\Delta + (B_b) \rightarrow B_b$ each execution
b ≠ 0 (Special)	Y=sy, a 16-bit signed constant Δ	The operand address Y=y + (B_b) + (S_s) for *Read* and *Store* instructions and for the read portion of the *Replace* instruction; for the store portion of *Replace* instruction Y = y + (B_b) + (S_6). Then $\Delta + B_b \rightarrow B_b$ each execution.

I LA 10 Load A $\underline{Y} \longrightarrow A_a$ 1.5

Load A_a with $\underline{Y}$

I LXB 11 Load A and Index B $\underline{Y} \longrightarrow A_a; (B_b)_{15\text{-}0} + 1 \longrightarrow B_b$ 1.5

Load A_a with $\underline{Y}$. Add 1 to the lower 16 bits of (B_b) and put the result in B_b.

I LDIF 12 Load Difference $\underline{Y} - (A_a) \longrightarrow A_{a+1}; (A_a)_i = (A_a)_f$ 1.5

Subtract (A_a) from $\underline{Y}$; put the result in A_{a+1}. (A_a) remain unchanged.

I ANA 13 Subtract A $(A_a) - \underline{Y} \longrightarrow A_a$ 1.5

Subtract $\underline{Y}$ from (A_a); put the result in A_a.

I AA 14 Add A $(A_a) + \underline{Y} \longrightarrow A_a$ 1.5

Add $\underline{Y}$ to (A_a); put the result in A_a.

I LSUM 15 Load Sum $\underline{Y} + (A_a) \longrightarrow A_{a+1}; (A_a)_i = (A_a)_f$ 1.5

Add $\underline{Y}$ to (A_a); put the result in A_{a+1}. (A_a) remain unchanged.

I LNA 16 Load Negative $\underline{Y}' \longrightarrow A_a$ 1.5

Load A_a with the one's complement of $\underline{Y}$.

I	LM	17	Load Magnitude	$\lvert \underline{Y} \rvert \longrightarrow A_a$	1.5

Load A_a with the absolute value of $\underline{Y}$. Complement whole-, half- or quarter-words when $\underline{Y}_{31}$ equals 1.

I	LB	20	Load B	$\underline{Y} \longrightarrow B_a$	2.0

Load the lower 16 bits of B_a with $\underline{Y}$. If a=0, no operation results.

I	AB	21	Add B	$(B_a)_{15\text{-}0} + \underline{Y} \longrightarrow B_a$	2.0

Add the lower 16 bits of (B_a), zero-extended, to $\underline{Y}$. Put the result in the lower 16 bits of B_a. If a=0, no operation results.

I	ANB	22	Subtract B	$(B_a)_{15\text{-}0} - \underline{Y} \longrightarrow B_a$	2.0

Subtract $\underline{Y}$ from the lower 16 bits of (B_a), zero-extended. Put the result in the lower 16 bits of B_a. If a = 0, no operation results.

I	SB	23	Store B	$(B_a)_{15\text{-}0} \longrightarrow Y$	1.5

Store the lower 16 bits of (B_a) at memory address $\underline{Y}$. If a = 0, the value zero is stored at $\underline{Y}$.

I	SA	24	Store A	$(A_a) \longrightarrow \underline{Y}$	1.5

Store (A_a) at memory address $\underline{Y}$.

I	SXB	25	Store A and Index B	$(A_a) \longrightarrow \underline{Y}; (B_b)_{15\text{-}0} + 1 \longrightarrow B_b$	1.5

Store (A_a) at memory address $\underline{Y}$. Add 1 to the lower 16 bits of (B_b); put the result in the lower 16 bits of B_b.

I	SNA	26	Store Negative	$(A_a)' \longrightarrow \underline{Y}$	1.5

Store the one's complement of (A_a) at memory address $\underline{Y}$.

I	SM	27	Store Magnitude	$\lvert (A_a) \rvert \longrightarrow \underline{Y}$	1.5

Store the absolute value of (A_a) at memory address $\underline{Y}$. Complement whole or partial words when $(A_a)_{31}$ equals 1.

I	BZ	32	Clear Bit	$0 \longrightarrow (Y)_{ak}$	2.5

Clear the bit position in memory address Y that corresponds to the value in the combined *ak* field. Values 0 through 37 octal are permitted for *ak*.

I	BS	33	Set Bit	$1 \longrightarrow (Y)_{ak}$	2.5

Set the bit position in memory address Y that corresponds to the value in the combined *ak* field. Values 0 through 37 octal are permitted for *ak*.

I	RA	34	Replace Add	$\underline{Y} + (A_a) \longrightarrow$ Y and also A_{a+1}	2.5

Add $\underline{Y}$ to (A_a) and store the result in both A_{a+1} and $\underline{Y}$. (A_a) remain unchanged.

I	RI	35	Replace Increment	$\underline{Y} \longrightarrow A_a$; $(A_a) + 1 \longrightarrow A_a$ and also Y	2.5

Load A_a with $\underline{Y}$. Add 1 to (A_a) and store the result in both A_a and $\underline{Y}$.

I	RAN	36	Replace Subtract	$\underline{Y} - (A_a) \longrightarrow$ Y and also A_{a+1}	2.5

Subtract (A_a) from $\underline{Y}$ and store the result in both A_{a+1} and Y. (A_a) remain unchanged.

I	RD	37	Replace Decrement	$\underline{Y} \longrightarrow A_a$; $(A_a) - 1 \longrightarrow A_a$ and also Y	2.5

Load A_a with $\underline{Y}$. Subtract 1 from (A_a) and store the result in both A_a and $\underline{Y}$.

I	M	40	Multiply A	$(A_a) \cdot \underline{Y} \longrightarrow A_{a+1}, A_a$	7.5+

Multiply (A_a) by $\underline{Y}$. Put the result in the double-length register formed by A_{a+1} and A_a, with the high-order half in A_{a+1}.

I	D	41	Divide A	$(A_{a+1}, A_a) \div \underline{Y} \longrightarrow A_a$, Rem. $\longrightarrow A_{a+1}$	14.5+

Divide the contents of the double-length register formed by A_{a+1} and A_a, by $\underline{Y}$. Put the quotient in A_a and the remainder in A_{a+1}. If the quotient exceeds 31 data bits and one sign bit, indicate as fixed-point overflow in the ASR (divide overflow).

I	BC	42	Compare Bit to Zero	$(Y)_{ak} : 0$ Set CD	1.5

Compare to zero the bit in (Y) that corresponds to the value in the combined *ak* field and set the compare designator to EQUAL or UNEQUAL as the result indicates. Values 0 through 37 octal are permitted for *ak*.

I	CXI	43	Compare Index Increment	$(B_a)_{15-0} : \underline{Y}$ If $\geq \underline{Y}$, O $\longrightarrow B_a$, CD ind. OL If $< \underline{Y}$, $(B_a) + 1 \longrightarrow B_a$, CD ind. WL	2.0

Compare (B_a) to the lower-order 16 bits of $\underline{Y}$. If (B_a) are equal to or greater than $\underline{Y}$, set the compare designator to OUTSIDE LIMITS and clear B_a. If (B_a) are less than $\underline{Y}$, increase (B_a) by 1 and set the compare designator to WITHIN LIMITS.

I	C	44	Compare	$(A_a) : \underline{Y}$ Set CD	1.5

Compare (A_a) with $\underline{Y}$. Set CD as follows:

If $(A_a) = \underline{Y}$, set ASR bit 2, (EQUAL)

If $(A_a) \neq \underline{Y}$, clear ASR bit 2, (UNEQUAL)

If $(A_a) \geq \underline{Y}$, set ASR bit 1, (GREATER THAN OR EQUAL TO)

If $(A_a) < \underline{Y}$, clear ASR bit 1, (LESS THAN)

Both ASR Bits will reflect comparison results.

+Execution time is independent of overlap operation

I	CL	45	Compare Limits	$(A_{a+1}) : \underline{Y}, (A_a) : \underline{Y}$ If $(A_{a+1}) > \underline{Y} \geq (A_a)$, CD ind. WL If $(A_{a+1}) \leq \underline{Y}$, or $\underline{Y} < (A_a)$, CD ind. OL	1.5

Compare (A_{a+1}) and (A_a) with $\underline{Y}$ and indicate the comparison result in the ASR as follows:

If $\underline{Y}$ lies between the two A values or is equal to (A_a) indicate WITHIN LIMITS; otherwise, indicate OUTSIDE LIMITS.

I	CM	46	Compare Masked	$(A_a) \odot \underline{Y} : (A_{a+1})$ Set CD	1.5

Compare the logical product of (A_a) and $\underline{Y}$ with (A_{a+1}). Indicate the result in the ASR comparison designator as follows:

If $(A_{a+1}) = (A_a) \odot \underline{Y}$, set ASR bit 2; (EQUAL)
If $(A_{a+1}) \neq (A_a) \odot \underline{Y}$, clear ASR bit 2; (UNEQUAL)
If $(A_{a+1}) \geq (A_a) \odot \underline{Y}$, set ASR bit 1; (GREATER THAN OR EQUAL TO)
If $(A_{a+1}) < (A_a) \odot \underline{Y}$, clear ASR bit 1; (LESS THAN)

(A_a) remain unchanged. Both ASR bits will reflect comparison results.

I	CG	47	Compare Gated	$\lvert \underline{Y} - (A_a) \rvert : (A_{a+1})$ Set CD	1.5

Compare the absolute value of the difference between $\underline{Y}$ and (A_a) with (A_{a+1}). Indicate the comparison result in the ASR as follows:

If $\lvert \underline{Y} - (A_a) \rvert = (A_{a+1})$, set ASR bit 2; (EQUAL)
If $\lvert \underline{Y} - (A_a) \rvert \neq (A_{a+1})$, clear ASR bit 2; (UNEQUAL)
If $\lvert \underline{Y} - (A_a) \rvert \geq (A_{a+1})$, set ASR bit 1; (GREATER THAN OR EQUAL TO)
If $\lvert \underline{Y} - (A_a) \rvert < (A_{a+1})$, clear ASR bit 1; (LESS THAN)

(A_a) remain unchanged. Both ASR bits will reflect comparison results.

OPERAND INTERPRETATIONS FOR JUMP INSTRUCTIONS (FORMAT III)

k is not used
When i = 0 the jump address $Y = y + (B_b) + (S_s)$
When i = 1 the indirect control address $Y = y + (B_b) + (S_s)$.

Indirect addressing continues through all indirect control words until i = 0 is encountered. Depending on the c-field in the indirect control word the jump address will be $Y = y + (B_b) + (S_s)$, $Y = sy + (S_b)$ or $Y = sy + (B_b)_{15\text{-}0} + (S)$ as specified by $(B_b)_{19\text{-}17}$ as designated by those respective fields in the indirect control word. A request for character addressing in the indirect control word for a Format III instruction is not allowed. These are jump instructions.

III	JEP	50 0	Jump on Even Parity	$(A_a) \odot (A_{a+1})$; If ODD parity, do NI; If EVEN parity, Y $\longrightarrow$ P	2.0

Form the logical product of (A_a) and (A_{a+1}). If the count of one bits in the result is ODD, execute the next instruction; otherwise, transfer Y to the P-register for a program jump. (See box, page 57.)

III	JOP	50 1	Jump on Odd Parity	$(A_a) \odot (A_{a+1})$; If EVEN parity, do NI; If ODD parity, Y $\longrightarrow$ P	2.0

Form the logical product of (A_a) and (A_{a+1}). If the count of ones in the result is EVEN, execute the next instruction; otherwise, transfer Y to the P-register for a program jump. (See box, page 57.)

III	DJZ	50 2	Jump on Double Precision Zero	If $(A_{a+1},A_a) \neq 0$, do NI If $(A_{a+1},A_a) = 0$, Y $\longrightarrow$ P	2.0

Form the double-length operand with (A_{a+1}) and (A_a). If the operand is not zero, execute the next instruction; otherwise, transfer Y to the P-register for a program jump. (See box, page 57.)

III	DJNZ	50 3	Jump on Double Precision Not Zero	If $(A_{a+1},A_a) \neq 0$, Y $\longrightarrow$ P If $(A_{a+1},A_a) = 0$, do NI	2.0

Form the double-length operand with (A_{a+1}) and (A_a). If the operand is zero, execute the next instruction; otherwise, transfer Y to the P-register for a program jump. (See box, page 57.)

III	JP	51 0	Jump A Positive	If $(A_a) \geq 0$, Y $\longrightarrow$ P, otherwise do NI	1.5

Test $(A_a)_{31}$ for algebraic sign. If (A_a) are negative, execute the next instruction; otherwise, transfer Y to the P-register for a program jump. (See box, page 57.)

III	JN	51 1	Jump A Negative	If $(A_a) < 0$, Y $\longrightarrow$ P, otherwise do NI	1.5

Test $(A_a)_{31}$ for algebraic sign. If (A_a) are positive (zero included), execute the next instruction; otherwise, transfer Y to the P-register for a program jump. (See box, page 57.)

III	JZ	51 2	Jump A Zero	If $(A_a) = 0$, Y $\longrightarrow$ P, otherwise, do NI	1.5

Test (A_a) for zero. If it is not zero, execute the next instruction; otherwise, transfer Y to the P-register for a program jump. (See box, page 57.)

III	JNZ	51 3	Jump A Not Zero	If $(A_a) \neq 0$, Y $\longrightarrow$ P, otherwise, do NI	1.5

Test (A_a) for zero. If it is zero, execute the next instruction; otherwise, transfer Y to the P-register for a program jump. (See box, page 57.)

III	LBJ	52 0	Load B and Jump	(P) + 1 $\longrightarrow$ B_a; Y $\longrightarrow$ P	1.8

Save the address of the next instruction by loading it into B_a; transfer Y to the P-register to effect a program jump, (i.e., transfer $(P)_{15-0}$ to $B_{a_{15-0}}$ and $(P)_{19-17}$ to $B_{a_{19-17}}$). (See box, page 57.)

III	JBNZ	52 1	Index Jump B	If $(B_a)_{15-0} = 0$, do NI, otherwise $(B_a)_{15-0} - 1 \longrightarrow B_a$; Y $\longrightarrow$ P	1.8

Test the lower 16 bits of (B_a) for zero. If (B_a) are zero, execute the next instruction; otherwise, decrease (B_a) by 1 and transfer Y to the P-register for a program jump. (See box, page 57.)

III	JS	52 2	Jump sy+B	[sy + $(B_b)_{15-0}$] zero ext. and $(B_b)_{19-17}$ $\longrightarrow$ P	1.5

Transfer Y to the P-Register for a program jump as follows:

Transfer sy + $(B_b)_{15-0}$ to P_{15-0}, the d-field

Transfer $(B_b)_{19-17}$ to P_{19-17}, the s-field

(The resultant jump address is sy + $(B_b)_{15-0}$ + (S) as designated by $(B_b)_{19-17}$.)

III	JL	52 3	Unconditional Jump Lower	Y $\longrightarrow$ P; $(Y)_l \longrightarrow U_u$	1.5

Transfer Y to the P-Register for a program jump and execute the instruction stored in the lower half of that address. (See box, page 57.)

III	JNF	53 0 a=0	Jump On No Overflow	If OD $\neq$ 1, Y $\longrightarrow$ P If OD = 1, clear it and do NI	1.5

If the fixed-point overflow designator is set, clear it and execute the next instruction. Otherwise, transfer Y to the P-Register for a program jump. (See box, page 57.)

III	JOF	53 0 a=1	Jump On Overflow	If OD = 1, clear it and Y $\longrightarrow$ P If OD $\neq$ 1, do NI	1.5

If the fixed-point overflow designator is not set, execute the next instructruction. Otherwise clear the designator and transfer Y to the P-Register for a program jump. (See box, page 57.)

III	JNE	53 1 a=0	Jump On Not Equal	If CD =, do NI If CD $\neq$, Y $\longrightarrow$ P	1.5

If the compare designator indicates EQUAL (ASR bit position 2 is set), execute the next instruction. If the CD indicates UNEQUAL, transfer Y to the P-Register for a program jump. (See box, page 57.)

III	JE	53 1 a=1	Jump On Equal	If CD $\neq$, do NI If CD =, Y $\longrightarrow$ P	1.5

If the compare designator indicates UNEQUAL (ASR bit position 2 is cleared), execute the next instruction. If the CD indicates EQUAL, transfer Y to the P-Register for a program jump. (See box, page 57.)

III	JG	53 1 a=2	Jump On Greater Than	If CD's ≥ and ≠, Y ⟶ P Otherwise, do NI	1.5

If the compare designators do not indicate GREATER THAN (ASR bit position 1 cleared OR position 2 set), execute the next instruction. If the CD indicates GREATER THAN OR EQUAL AND UNEQUAL (ASR bit position 1 set AND position 2 cleared), transfer Y to the P-Register for a program jump. (See box, page 57.)

III	JGE	53 1 a=3	Jump On Greater Than Or Equal	If CD's ≥ or = or both, Y ⟶ P Otherwise, do NI	1.5

If the compare designators do not indicate GREATER THAN OR EQUAL, execute the next instruction. If the CD's indicate either GREATER THAN OR EQUAL OR EQUAL (ASR bit position 1, OR bit positions 1 and 2 are set), transfer Y to the P-Register for a program jump. (See box, page 57.)

III	JLT	53 1 a=4	Jump on Less Than	If CD indicates < , Y ⟶ P; Otherwise, do NI	1.5

If the compare designator does not indicate LESS THAN, execute the next instruction. If the CD indicates LESS THAN (ASR bit position 1 cleared), transfer Y to the P-Register for a program jump. (See box,page57.)

III	JLE	53 1 a=5	Jump on Less Than Or Equal	If CD indicates < or = or both, Y ⟶ P; otherwise, do NI	1.5

If the compare designators do not indicate either LESS THAN OR EQUAL, execute the next instruction. If the CD indicates either LESS THAN OR EQUAL (ASR bit position 2 set OR bit position 1 cleared), transfer Y to the P-Register for a program jump. (See box, page 57.)

III	JNW	53 1 a=6	Jump Outside Limits	If CD is OL, Y ⟶ P; Otherwise, do NI	1.5

If the compare designator does not indicate OUTSIDE LIMITS, execute the next instruction. If the CD indicates OUTSIDE LIMITS (ASR bit position 0 cleared), transfer Y to the P-Register for a program jump. (See box, page 57.)

III	JW	53 1 a=7	Jump Within Limits	If CD is WL, Y ⟶ P; otherwise, do NI	1.5

If the compare designator does not indicate WITHIN LIMITS, execute the next instruction. If the CD indicates WITHIN LIMITS (ASR bit position 0 set), transfer Y to the P-Register for a program jump. (See box, page 57.)

III	RJ	53 2 a=0	Return Jump	(P) + 1 ⟶ Y; Y +1 ⟶ P	3.0

Store the contents of P +1 in address Y. Transfer Y+1 to the P-Register to effect a program jump. (See box, page 57.)

III	RJC	53 2 a=1, 2, or 3	Return Jump	(P) + 1 ⟶ Y; Y + 1 ⟶ P, per JUMP switch selected otherwise do NI	3.0

If JUMP switch *a* is selected, store the contents of P +1 in Y. Transfer Y+1 to the P-Register to effect a program jump. If switch *a* is not selected, execute the next instruction. (See box, page 57.)

III	RJSC	53 2 a=4, 5, 6, or 7 (Privileged)	Return Jump	(P) + 1 ⟶ Y; Y+1 ⟶ P Stop per STOP switch selected; otherwise, execute jump without stop	

Store the contents of P, +1, in Y; transfer Y+1 to the P-Register; and, if a=4 or STOP switch *a* is selected, stop the computer. When the computer is started again, execute the jump. (See box, page 57.)

III	J	53 3 a=0	Manual Jump	Y ⟶ P	1.5

Transfer Y to the P-Register and execute the whole or upper-half-word instruction stored at that address. (See box, page 57.)

III	JC	53 3 a=1, 2, or 3	Manual Jump	Y ⟶ P per JUMP switch selected otherwise, do NI	1.5

If JUMP switch *a* is selected, transfer Y to the P-Register for a program jump. Execute the whole or upper-half-word instruction stored at that address. (See box, page 57.)

III	JSC	53 3 a=4, 5, 6, or 7 (Privileged)	Manual Jump	Y ⟶ P Stop per STOP switch selected; otherwise, execute jump without stop	

Transfer Y to the P-Register and, if a=4 or STOP switch *a* is selected, stop the computer. When the computer is started again, execute the jump. (See box, page 57.)

I	LCT	54	Load CMR Task	(Y) ⟶ CMR_{ak}	1.5

Load the contents of memory address Y into the control memory register specified by the combined *ak* field. Values for *ak* from 00 through 77 are permitted as follows:

00-17 for Task State
20-27 for Interrupt State only (privileged)
30-57 not available
60-77 for Interrupt State only (privileged)

NOTE: If this instruction is executed in the repeat mode, the instruction is privileged and cannot be interrupted; the *ak* value is increased by 1 with each execution.

I	LCI	55	Load CMR Interrupt (Privileged)	(Y) ⟶ CMR_{ak+100}	1.5

Load the contents of memory address Y into the control memory register specified by the combined *ak* field plus 100. Values of *ak* from 00 through 27 and from 40 through 77 are permitted. (Registers 130-137 are not available.)

NOTE: Changing the base register designated by P_s will cause a program jump. Also see note under 54.

I	SCT	56	Store CMR Task	$(CMR_{ak}) \longrightarrow Y$	1.5

Store the contents of the control memory address specified by the combined *ak* field in memory address Y. Values of *ak* from 00 through 77 are permitted as follows:

00-17 for Task State
20-27 for Interrupt State only (privileged)
30-57 not available
60-77 for Interrupt State only (privileged)

(See note under code 54.)

I	SCI	57	Store CMR Interrupt (Privileged)	$(CMR_{ak+100}) \longrightarrow Y$	1.5

Store the contents of control memory address, specified by the combined *ak* field plus 100, in memory address Y. Values of *ak* from 00-27 and 40-77 are permitted. (Register addresses 130-137 are not available). (See note under code 54.)

IV-A	HSCT	60 i=0	Store CMR in A (Task)	$(CMR_{af_4}) \longrightarrow A_b$	1.75

Store the contents of the control memory address specified by the combined af_4 field in the A-register specified by the *b* field. Values of af_4 from 00 through 77 are permitted as follows:

00-17 for Task State
20-27 for Interrupt State only (privileged)
30-57 not available
60-77 for Interrupt State only (privileged)

For addresses 11-17 (index registers) only the low-order 16 bits are transferred.

IV-A	HSCI	60 i=1	Store CMR in A (Interrupt) (Privileged)	$(CMR_{af_4+100}) \longrightarrow A_b$	1.75

Transfer the contents of the control memory register specified by the combined af_4 field plus 100 to the A-register specified by the *b* field. For addresses 11-17 (index registers) only the low-order 16 bits are transferred. Values for af_4 from 00 through 27 and 40-77 are permitted. (Register addresses 130-137 are not available.)

IV-A	HLCT	61 i=0	Load CMR from A (Task)	$(A_b) \longrightarrow CMR_{af_4}$	1.75

Load the control memory register specified by the combined af_4 field with the contents of the A-Register specified by the *b* field. Values of af_4 from 00 to 77 are permitted as follows:

00-17 for Task State
20-27 for Interrupt State only (privileged)
30-57 not available
60-77 for Interrupt State only (privileged)

For addresses 11 through 17 (index registers) only the low-order 16 bits are transferred.

IV-A HLCI 61 i=1 Load CMR from A (Interrupt) (Privileged) $(A_b) \longrightarrow CMR_{af_4+100}$ 1.75

Load the control memory register specified by the combined af_4 field plus 100 with the contents of the A-register specified by the b field. Values for af_4 from 00-27 and 40-77 are permitted. For addresses 111 through 117 (index registers) only the low-order 16 bits are transferred. (Register addresses 130-137 are not available.)

IV-B HLC 62 Shift Left Circular 1.75

Shift (A_a) circularly to the left according to the m field shift count definition in Figure 23 . The maximum shift is 63 places.

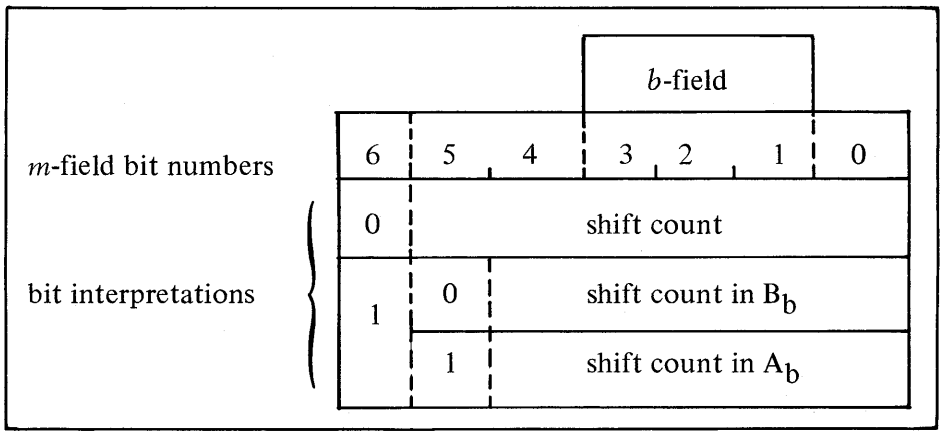


Figure 23 . m-Field Interpretation for Shift Instructions (codes 62-67)

IV-B HDLC 63 Shift Double Left Circular 1.75

Shift the double-length word with (A_{a+1}) at the high-order half and (A_a) at the low-order half circularly to the left, according to the m-field shift count definition in Figure 23. The maximum shift is 63 places.

IV-B HRZ 64 Shift Right Fill Zeros 1.75

Shift (A_a) to the right according to the m-field shift count definition in Figure 23. Drop bits at low-order end and fill with zeros at the high-order end. The maximum shift is 63 places.

IV-B HDRZ 65 Shift Right Double, Fill Zeros 1.75

Shift the double word with (A_{a+1}) at the high-order half and (A_a) at the low-order half to the right according to the m-field shift count definition in Figure 23. Drop bits at the low-order end and fill with zeros from the high-order end. The maximum shift is 63 places.

IV-B1 HRS 66 Shift Right Sign Fill 1.75

Shift (A_a) to the right according to the m-field shift count definition in Figure 23. Drop bits at the low-order end and fill the high-order end with the sign bit from $(A_a)_{31}$. The maximum shift is 63 places.

IV-B HDRS 67 Shift Right Double, Sign Fill 1.75

Shift the double word with (A_{a+1}) at the high-order half and (A_a) at the low-order half to the right according to the *m*-field shift count definition in Figure 23. Drop bits at the low-order end and fill the high-order end with the sign bit from $(A_{a+1})_{31}$. The maximum shift count is 63 places.

IV-A HSF 70 0 Scale Factor 2.25

Shift (A_a) circularly to the left until $(A_a)_{31} \neq (A_a)_{30}$. Count the number of places shifted and store the count in A_b. If $a=b$, the word is normalized but no shift count is stored. If (A_a) are all ones or all zeros, the maximum shift count, 37_8, is stored.

IV-A HDSF 70 1 Double Scale Factor 2.25

Shift the double-length word with (A_{a+1}) as the high-order half and (A_a) as the low-order half circularly to the left until $(A_{a+1})_{31} \neq (A_{a+1})_{30}$. Count the number of places shifted and store the count in A_b. If a or $a+1 = b$, the word is normalized but no shift count is stored. If (A_{a+1}, A_a) are all ones or all zeros, the maximum shift count, 77_8 is stored.

IV-A HCP 70 2 Complement A $(A_a)' \longrightarrow A_a$ 1.1

Complement (A_a) and leave the results in A_a.

IV-A HDCP 70 3 Double Complement A $(A_{a+1},A_a)' \longrightarrow A_{a+1},A_a$ 1.1

Complement the double-length word with (A_{a+1}) at the high-order end and (A_a) at the low-order end; store the result in the double-length register A_{a+1}, A_a.

IV-A HOR 71 0 Logical Sum $(A_a) \oplus (A_b) \longrightarrow A_a$ 1.0

Set each bit position in A_a where the corresponding bit in A_b equals 1 (Inclusive OR). Other bits in A_a and all bits in A_b remain unchanged.

IV-A HA 71 1 Sum $(A_a) + (A_b) \longrightarrow A_a$ 1.0

Add (A_a) to (A_b) and store the result in A_a. If $a=b$, (A_b) are changed by the addition; otherwise, they are not.

IV-A HAN 71 2 Difference $(A_a) - (A_b) \longrightarrow A_a$ 1.0

Subtract (A_b) from (A_a) and store the result in A_a. If $a=b$, $(A_a)_f=0$. se,

IV-A HXOR 71 3 Logical Difference $(A_a) \bar{\oplus} (A_b) \longrightarrow A_a$ 1.0

Complement each bit in A_a where the corresponding bit in A_b equals 1 (Exclusive OR). Other bits in A_a and (if $a \neq b$) all bits in A_b remain unchanged. If $a=b$, $(A_a)_f = 0$.

IV-A HAND 71 5 And $(A_a) \odot (A_b) \longrightarrow A_a$ 1.0

Form the logical product of (A_a) and (A_b). Store the result in A_a, leaving (A_b) unchanged.

IV-A HM 74 0 Multiply Register $(A_a) \cdot (A_b) \longrightarrow A_{a+1}, A_a$ 7.75^+

Multiply (A_a) by (A_b); store the double-length product in A_{a+1} and A_a, with A_{a+1} holding the high-order half. (A_b) are not changed unless $b=a$ or $a+1$.

IV-A HD 74 1 Divide Register $(A_{a+1}, A_a) \div (A_b) \longrightarrow A_a$; Rem. $\longrightarrow A_{a+1}$ 15.0^+

Divide the double-length word with (A_{a+1}) at the high-order half and (A_a) at the low-order half by (A_b). Store the quotient in A_a and the remainder in A_{a+1}. (A_b) are not changed unless $b=a$ or $a+1$.

IV-A HRT 74 2 Square Root $\sqrt{(A_{a+1}, A_a)} \longrightarrow A_b$; Residue $\longrightarrow A_{b+1}$ 15.0

Take the square root of the double-length word with (A_{a+1}) at the high-order half and (A_a) at the low-order half. Store the root in A_b and the residue in A_{b+1}. (A_a) remain unchanged unless $a=b$ or $b+1$; and (A_{a+1}) remain unchanged unless $b=a$ or $a+1$.

IV-A HLB 74 3 Load B_a with B_b $(B_b)_{15\text{-}0} \longrightarrow B_a$ 1.75

Load the low-order 16 bits of B_b into the corresponding bit positions in B_a. If $a = 0$, no operation results.

IV-A HC 74 4 Compare, Register $(A_a):(A_b)$; Set CD 1.1

Compare (A_a) with (A_b). Indicate the result in the ASR comparison designator as follows:

If $(A_a) = (A_b)$, set ASR bit position 2; (EQUAL)
If $(A_a) = (A_b)$, clear ASR bit position 2; (UNEQUAL)
If $(A_a) \geq (A_b)$, set ASR bit position 1; (GREATER THAN OR EQUAL TO)
If $(A_a) < (A_b)$, clear ASR bit position 1; (LESS THAN)

Both ASR bits will reflect comparison results.

IV-A HCL 74 5 Compare Limits, Register $(A_b):(A_{a+1})$ & (A_a); Set CD 1.75

Compare (A_b) with (A_{a+1}) and (A_a). Indicate the comparison result in the ASR as follows:

If $(A_{a+1}) \geq (A_b) \geq (A_a)$, clear ASR bit position 0; (WITHIN LIMITS)
If $(A_b) \geq (A_{a+1})$ or $(A_a) > (A_b)$, set ASR bit position 0; (OUTSIDE LIMITS)

$^+$Execution time is independent of overlap operation.

IV-A HCM 74 6 Compare Masked, Register $(A_{a+1}) \odot (A_a):(A_b)$; Set CD 1.1

Compare the logical product of (A_{a+1}) and (A_a) with (A_b). Indicate the comparison result in the ASR as follows:

If $(A_{a+1}) \odot (A_a) = (A_b)$, set ASR bit position 2; (EQUAL)
If $(A_{a+1}) \odot (A_a) \neq (A_b)$, clear ASR bit position 2; (UNEQUAL)
If $(A_{a+1}) \odot (A_a) \geq (A_b)$, set ASR bit position 1; (GREATER THAN OR EQUAL TO)
If $(A_{a+1}) \odot (A_a) < (A_b)$, clear ASR bit position 1; (LESS THAN)

Both ASR bits will reflect comparison results.

IV-A HCB 74 7 Compare B_b with B_a $(B_b)_{15-0}:(B_a)_{15-0}$; Set CD 2.0

Compare the 16-bit indexing fields of (B_b) and (B_a). Indicate the comparison result in the ASR as follows:

If $(B_b) = (B_a)$, set ASR bit position 2; (EQUAL)
If $(B_b) \neq (B_a)$, clear ASR bit position 2; (UNEQUAL)
If $(B_b) \geq (B_a)$, set ASR bit position 1; (GREATER THAN OR EQUAL TO)
If $(B_b) < (B_a)$, clear ASR bit position 1; (LESS THAN)

Both ASR bits will reflect comparison results.

IV-A HSIM 77 0 Store IOC Monitor Clock in A $(\text{Monitor Clock})_{IOC_a} \longrightarrow A_b$ 3.0

Load A_b with the contents of the monitor clock in the IOC designated by *a*. Values for *a*=0-3 are permitted and 4-7 are not used.

IV-A HSTC 77 1 Store Real-Time Clock in A $(\text{Real-Time Clock})_{IOC_a} \longrightarrow A_b$ 3.5

Load A_b with the contents of the real-time clock in the IOC designated by *a*. Values of *a*=0-3 are permitted and 4-7 are not used.

IV-A HPI 77 4 Prevent Class III Interrupts 2.25
(Privileged)

Set the Class III interrupt lockout and hold all pending interrupts in that class.

IV-A HAI 77 5 Allow Class III Interrupts 2.25
(Privileged)

Clear the Class III Interrupt Lockout but not the individual channel interrupt enable logic.

IV-A HALT 77 6 STOP Processor 2.25
a=0 (Privileged)

Stop processor operation and light the STOP 4 indicator.

IV-A	HWFI	77 6 a=1	Wait for Interrupt	2.25

Stop processor-memory references until an interrupt is received from any class not currently locked out. Process the interrupt. Resume normal operations after returning from the interrupt routine with the instruction following this HWFI.

Input-Output Controller

This section lists the instructions for the input-output controller in the following format.

- Assembler code, octal code, instruction name, symbolic summary, and execution time in microseconds
- Text defining the instruction in detail
- Notes

IB	10	Initiate Input Buffer On C_j	(y) $\longrightarrow$ CMR_{0+j}; Activate Input	3.25

Transfer (y) and the *m*, *c* and *k* fields to the input buffer control word at control memory address 0+*j* and activate the input buffer on channel *j*.

OB	11	Initiate Output Buffer On C_j	(y) $\longrightarrow$ CMR_{20+j}; Activate Output	3.25

Transfer (y) and the *m*, *c* and *k* fields to the output buffer control word at control memory address 20+*j* and activate the output buffer on channel *j*.

FB	12	Initiate EF Buffer On C_j	(y) $\longrightarrow$ CMR_{40+j}; Activate EF	3.25

Transfer (y) and the *m*, *c* and *k* fields to the external function buffer control word at control memory address 40+*j* and activate the external function buffer mode on channel *j*.

XB	13	Initiate EI Buffer On C_j	(y) $\longrightarrow$ CMR_{60+j}; Activate EI	3.25

Transfer (y) and the *m*, *c* and *k* fields to the external interrupt buffer control word at control memory address 60+*j* and activate the external interrupt buffer mode on channel *j*.

TIB	14 *k*=0	Terminate Input Buffer On C_j	3.0
TOB	14 *k*=1	Terminate Output Buffer On C_j	3.0
TFB	14 *k*=2	Terminate External Function Buffer On C_j	3.0
TXB	14 *k*=3	Terminate External Interrupt Buffer On C_j	3.0

Terminate the transfer operation designated by *k* on the channel designated by *j*, suppress the related monitor interrupt, and terminate the associated chain if the buffer was active. If the buffer was not active, allow the chain to continue; and, in addition, if *m*=0, suppress a presently-queued monitor interrupt, but if *m*=1, allow a presently-queued monitor interrupt.

IMIR	15 k=0	Set Input Monitor Interrupt On C_j		2.5
OMIR	15 k=1	Set Output Monitor Interrupt On C_j		2.5
FMIR	15 k=2	Set External Function Monitor Interrupt On C_j		2.5
XMIR	15 k=3	Set External Interrupt Monitor Interrupt On C_j		2.5

Set the monitor interrupt request (to all processors not locked out) for the input/output transfer type designated by k and for the channel designated by j. y is not used.

AIC	16 k=0	Activate Input Chain On C_j		2.5
AOC	16 k=1	Activate Output Chain On C_j	$y \longrightarrow CMR_{20k+j}$(bits 55-38); Activate Chain	2.5
AFC	16 k=2	Activate External Function Chain On C_j		2.5
AXC	16 k=3	Activate External Interrupt Chain On C_j		2.5

Transfer y to the command address pointer (bits 55-38) at CMR_{20k+j} and activate the corresponding chain. (Set the chain active on channel j for the operation specified by k.)

TBZ	17 m=0	Test Bit Zero	$(y)_{kj}$:m If =, SKIP; If ≠, do NI	4.0
TBS	17 m=1	Test Bit Set		4.0

Compare the bit in (y) specified by the combined k and j fields to the bit in the m field. If the two are equal, skip the next instruction (the pointer field is indexed by 2); if they are unequal, do the next instruction (the pointer field is indexed by 1). The combined kj field is interpreted as a counter whose value numerically corresponds to the bit in (y) in the range 00 through 37_8 (k values of 2 and 3 are not permitted).

JIO	20	Jump (Input/Output)	(y) $\longrightarrow$ CMR_{55-38} or CAR	2.5

If the address of this instruction was in the Command Address Register, transfer y to that CAR; if the address was in the command address pointer (bits 55-38) of a control memory word, transfer y to that CAP. The c-field of this instruction must be 1, to execute the jump. The next instruction in this I/O sequence is read from (y).

LICM	22	Load IOC Control Memory	(y) $\longrightarrow$ $CMR_{kj(bits\ 31-0)}$	3.25

Transfer (y) to the 32 low-order bits of the control memory register specified by the combined k and j fields. These fields are interpreted as a combined counter, whose value numerically corresponds to the control memory address in the range of 00 through 77_8.

ILTC	23	Load Real-Time Clock	(y) ⟶ RTC	4.0

Load the real-time clock with (*y*).

SICM	24	Store IOC Control Memory	$(CMR_{kj})_{31-0} \longrightarrow y$	2.75

Store in address *y* the contents of the 32 low-order bits of the control memory register specified by the combined *k* and *j* fields. These fields are interpreted as a combined counter, whose value numerically corresponds to the control memory address in the range of 00 through 77_8.

IBS	25	Set Bit	$1 \longrightarrow (y)_{kj}$	3.25

Set the bit in (*y*) specified by the combined *k* and *j* fields. These fields are interpreted as a combined counter, whose value numerically corresponds to the bit of (*y*), in the range 00 through 37_8 (*k*=2 and 3 are not permitted).

IBZ	26	Clear Bit	$0 \longrightarrow (y)_{kj}$	3.25

Clear the bit in (*y*) specified by the combined *k* and *j* fields. These fields are interpreted as a combined counter, whose value numerically corresponds to the bit of (*y*), in the range 00 through 37_8 (*k*=2 and 3 are not permitted).

ITSF	27	Test and Set Flag	$(y)_{31}$:1 If =, CAR+1 or CAP+1 ⟶ CAR or CAP If ≠, CAR+2 or CAP+2 ⟶ CAR or CAP and $1 \longrightarrow (y)_{31}$	3.25

Test bit 31 of (*y*). If it is 1, execute the next instruction (current pointer address plus 1 to pointer). If it is 0, set the bit position and skip the next instruction (current pointer address plus 2 to pointer).

AN/UYK-7 COMPUTER
REPERTOIRE OF INSTRUCTIONS

Code	Mnemonic	NAME	DESCRIPTION	F	CA	R	UF	Time‡ μs
01 0	OR	Inclusive OR (Selective Set A)	$(Y) \oplus (A_a) \rightarrow A_a$	II	Y	Y	2	1.5
01 1	SC	Selective Clear A	$(A_a) \odot (Y)' \rightarrow A_a$	II	Y	Y	2	1.5
01 2	MS	Selective Substitute	$(Y)_n \rightarrow (A_{a+1})_n$ for all $(A_a)_n=1$; $(A_a)_i=(A_a)_f$	II	Y	Y	2	1.5
01 3	XOR	Exclusive OR (Sel. Comp. A)	$(Y) \bar{\oplus} (A_a) \rightarrow A_a$; $(A_a)_n' \rightarrow (A_a)_n$ for $(Y)_n=1$	II	Y	Y	2	1.5
01 4	ALP	Add Logical Product	$(A_{a+1}) + (Y) \odot (A_a) \rightarrow A_{a+1}$; $(A_a)_i=(A_a)_f$	II	Y	Y	2	1.5
01 5	LLP	Load Logical Product	$(Y) \odot (A_a) \rightarrow A_a$	II	Y	Y	2	1.5
01 6	NLP	Subtract Logical Product	$(A_{a+1}) - (Y) \odot (A_a) \rightarrow A_{a+1}$; $(A_a)_i=(A_a)_f$	II	Y	Y	2	1.5
01 7	LLPN	Load Logical Product Next	$(Y) \odot (A_a) \rightarrow A_{a+1}$; $(A_a)_i=(A_a)_f$	II	Y	Y	2	1.5
02 0	CNT	Count Ones	No. of Bits Set in $(Y) \rightarrow A_a$	II	Y	Y	2	7.5†
02 2	XR	Execute Remote	$(Y) \rightarrow U$	II	N	N	8	1.5
02 3	XRL	Execute Remote Lower	$(Y)_L \rightarrow U$	II	N	N	8	1.5
02 4	SLP	Store Logical Product	$(A_{a+1}) \odot (A_a) \rightarrow Y$; $(A_a)_i=(A_a)_f$; $(A_{a+1})_i=(A_{a+1})_f$	II	Y	Y	2	1.5
02 5	SSUM	Store Sum	$(A_a)+(A_{a+1}) \rightarrow A_{a+1}$ & Y; $(A_a)_i=(A_a)_f$	II	Y	Y	2	2.0
02 6	SDIF	Store Difference	$(A_{a+1})-(A_a) \rightarrow A_{a+1}$ & Y; $(A_a)_i=(A_a)_f$	II	Y	Y	2	2.0
02 7	DS	Double Store A	$(A_{a+1}, A_a) \rightarrow Y+1, Y$	II	N	N	2	3.0
03 0	ROR	Replace Inclusive OR	$(Y) \oplus (A_a) \rightarrow A_a$ & Y	II	Y	Y	2	2.5
03 1	RSC	Replace Selective Clear	$(A_a) \odot (Y)' \rightarrow A_a$ & Y	II	Y	Y	2	2.5
03 2	RMS	Replace Selective Substitute	$(Y)_n \rightarrow (A_{a+1})_n$ for all $(A_a)_n=1$; Then $(A_{a+1}) \rightarrow Y$; $(A_a)_i=(A_a)_f$	II	Y	Y	2	2.5
03 3	RXOR	Replace Exclusive OR	$(Y) \bar{\oplus} (A_a) \rightarrow A_a$ & Y; $(A_a)_n' \rightarrow A_a$ & Y for $Y_n=1$	II	N	Y	2	2.5
03 4	RALP	Replace A+Logical Product	$(A_{a+1})+(Y) \odot (A_a) \rightarrow A_{a+1}$ & Y; $(A_a)_i=(A_a)_f$	II	Y	Y	2	2.5
03 5	RLP	Replace Logical Product	$(Y) \odot (A_a) \rightarrow Y$ & A_{a+1}; $(A_a)_i=(A_a)_f$	II	Y	Y	2	2.5
03 6	RNLP	Replace A—Logical Product	$(A_{a+1})-(Y) \odot (A_a) \rightarrow A_{a+1}$ & Y; $(A_a)_i=(A_a)_f$	II	Y	Y	2	2.5
03 7	TSF	Test and Set Flag	If $(Y)_{31}=0$, CD Set EQUAL. $1 \rightarrow Y_{31}$ If $(Y)_{31}=1$, CD Set UNEQUAL.	II	N	Y	8	2.5
05 0	DL	Double Load A	$(Y+1, Y) \rightarrow A_{a+1}, A_a$	II	N	N	2	3.0
05 1	DA	Double Add A	$(A_{a+1}, A_a)+(Y+1, Y) \rightarrow A_{a+1}, A_a$	II	N	N	2	3.0
05 2	DAN	Double Subtract A	$(A_{a+1}, A_a)-(Y+1, Y) \rightarrow A_{a+1}, A_a$	II	N	N	2	3.0
05 3	DC	Double Compare	Compare (A_{a+1}, A_a) to $(Y+1, Y)$, Set CD	II	N	N	2	3.0
05 4	LBMP	Load Base and Memory Protection	$(Y)_{17-0} \rightarrow S_a$; $(Y+1)_{20-0} \rightarrow SPR_a$; $Y \rightarrow SIR_a$ Privileged if; ASR bit 8 =0, s ≠ 7 or a=7	II	N	N	2	5.75
06 0	FA	Floating-point Add	Shift (A_{a+1}) or $(Y+1)$ Right such that $(A_a)=(Y)$ $(A_{a+1})+(Y+1) \rightarrow A_{a+1}$; Normalize	II	N	N	2	6.25†
06 1	FAN	Floating-point Subtract	Shift (A_{a+1}) or $(Y+1)$ Right such that $(A_a)=(Y)$ $(A_{a+1})-(Y+1) \rightarrow A_{a+1}$; Normalize	II	N	N	2	6.25†
06 2	FM	Floating-point Multiply	$(A_a)+(Y) \rightarrow (A_a)$ $(A_{a+1}) \bullet (Y+1) \rightarrow A_{a+1}$; Normalize	II	N	N	2	10.0†
06 3	FD	Floating-point Divide	$(A_a)-(Y) \rightarrow (A_a)$ $(A_{a+1}) \div (Y+1) \rightarrow A_{a+1}$; Normalize	II	N	N	2	17.0†
06 4	FAR	Floating-point Add with Round	Same as FA with (A_{a+1}) rounded	II	N	N	2	6.25†
06 5	FANR	Floating-point Subtract w/Rd.	Same as FAN with (A_{a+1}) rounded	II	N	N	2	6.25†
06 6	FMR	Floating-point Multiply w/Rd.	Same as FM with (A_{a+1}) rounded	II	N	N	2	10.0†
06 7	FDR	Floating-point Divide w/Rd.	Same as FD with (A_{a+1}) rounded	II	N	N	2	17.0†
07 0 a=0	XS	Enter Executive State	$sy+(B_b) \rightarrow$ CMR 156; Enter class IV (Executive)	II	N	N	11	4.0
07 0* a=1	IPI	Interprocessor Interrupt	Send Class II interrupt to processors n (0-7) IF bit n of $sy+(B_b)=1$	II	N	N	11	4.0
07 1**	AEI	Allow Enable Interrupt	Allow Monitor interrupts from IOCa on Channels n; IF bit n of $sy+(B_b)=1$	II	N	N	6	2.0
07 2**	PEI	Prevent Enable Interrupt	Prevent Monitor interrupts from IOCa on Channels n; IF bit n of $sy+(B_b)=1$	II	N	N	6	2.0
07 3**	LIM	Load IOC Monitor Clock	$sy+(B_b) \rightarrow$ IOCa MON CLK	II	N	N	6	3.0
07 4**	IO	Initiate I/O	Initiate IOCa at address Y	II	N	N	2	3.5
07 5*	IR	Interrupt Return	Return to State Specified by ASR storage DSW	II	N	N	9	3.0
07 6	RP	Repeat	Repeat N.I.B_7 Times; sy of Repeat added to B_b of N.I. after each cycle. See Repeat Conditions	II	N	N	6	1.5
10	LA	Load A	$\underline{Y} \rightarrow A_a$	I	Y	Y	1	1.5
11	LXB	Load A and Index B	$\underline{Y} \rightarrow A_a$; $(B_b)+1 \rightarrow B_b$	I	Y	N	1	1.5
12	LDIF	Load Difference	$\underline{Y}-(A_a) \rightarrow A_{a+1}$; $(A_a)_i=(A_a)_f$	I	Y	Y	1	1.5
13	ANA	Subtract A	$(A_a)-\underline{Y} \rightarrow A_a$	I	Y	Y	1	1.5

REPERTOIRE OF INSTRUCTIONS (CONT.)

Code	Mnemonic	NAME	DESCRIPTION	F	CA	R	UF	Time‡ μs
14	AA	Add A	$(A_a)+\underline{Y}\rightarrow A_a$	I	Y	Y	1	1.5
15	LSUM	Load Sum	$(A_a)+\underline{Y}\rightarrow A_{a+1}$; $(A_a)_i=(A_a)_f$	I	Y	Y	1	1.5
16	LNA	Load Negative	$\underline{Y}'\rightarrow A_a$	I	Y	Y	1	1.5
17	LM	Load Magnitude	$\lvert\underline{Y}\rvert\rightarrow A_a$	I	Y	Y	1	1.5
20	LB	Load B	$\underline{Y}\rightarrow B_a$	I	Y	Y	1	2.0
21	AB	Add B	$(B_a)+\underline{Y}\rightarrow B_a$; B_a zero extended	I	Y	Y	1	2.0
22	ANB	Subtract B	$(B_a)-\underline{Y}\rightarrow B_a$; B_a zero extended	I	Y	Y	1	2.0
23	SB	Store B	$(B_a)\rightarrow\underline{Y}$	I	Y	Y	1	1.5
24	SA	Store A	$(A_a)\rightarrow\underline{Y}$	I	Y	Y	1	1.5
25	SXB	Store A and Index B	$(A_a)\rightarrow\underline{Y}$; $(B_b)+1\rightarrow B_b$	I	Y	N	1	1.5
26	SNA	Store Negative	$(A_a)'\rightarrow\underline{Y}$	I	Y	Y	1	1.5
27	SM	Store Magnitude	$\lvert(A_a)\rvert\rightarrow\underline{Y}$	I	Y	Y	1	1.5
32	BZ	Clear Bit	$0\rightarrow Y_{ak}$	I	N	Y	3	2.5
33	BS	Set Bit	$1\rightarrow Y_{ak}$	I	N	Y	3	2.5
34	RA	Replace Add	$(A_a)+\underline{Y}\rightarrow A_{a+1}$ & $\underline{Y}$; $(A_a)_i=(A_a)_f$	I	Y	Y	1	2.5
35	RI	Replace Increment	$\underline{Y}+1\rightarrow A_a$ & $\underline{Y}$	I	Y	Y	1	2.5
36	RAN	Replace Subtract	$\underline{Y}-(A_a)\rightarrow A_{a+1}$ & $\underline{Y}$; $(A_a)_i=(A_a)_f$	I	Y	Y	1	2.5
37	RD	Replace Decrement	$\underline{Y}-1\rightarrow A_a$ & $\underline{Y}$	I	Y	Y	1	2.5
40	M	Multiply A	$(A_a)\cdot\underline{Y}\rightarrow A_{a+1}, A_a$	I	Y	Y	1	7.5†
41	D	Divide A	$(A_{a+1}, A_a)\div\underline{Y}\rightarrow A_a$; remainder$\rightarrow A_{a+1}$	I	Y	Y	1	14.5†
42	BC	Compare Bit to Zero	If $(Y)_{ak}=0$, CD Set EQUAL If $(Y)_{ak}=1$, CD Set UNEQUAL	I	N	Y	3	1.5
43	CXI	Compare Index Increment	If $(B_a)\geq\underline{Y}$, CD Set OUTSIDE, $0\rightarrow B_a$ If $(B_a)<\underline{Y}$, CD Set WITHIN, $(B_a)+1\rightarrow B_a$	I	Y	Y	1	2.0
44	C	Compare	Compare (A_a) to $\underline{Y}$, Set the CD	I	Y	Y	1	1.5
45	CL	Compare Limits	If $(A_{a+1})>\underline{Y}\geq(A_a)$, Set CD within	I	Y	Y	1	1.5
46	CM	Compare Masked	Compare (A_{a+1}) to $(A_a)\odot\underline{Y}$, Set the CD	I	Y	Y	1	1.5
47	CG	Compare Gated	Compare $\lvert\underline{Y}-(A_a)\rvert$ to (A_{a+1}), Set the CD	I	Y	Y	1	1.5
50 0	JEP	Jump on Even Parity	If $(A_{a+1})\odot(A_a)$ is Even Parity, jump to Y	III	N	N	1	2.0
50 1	JOP	Jump on Odd Parity	If $(A_{a+1})\odot(A_a)$ is Odd Parity, jump to Y	III	N	N	1	2.0
50 2	DJZ	Jump Double Precision Zero	If $(A_{a+1}, A_a)=0$, jump to Y	III	N	N	1	2.0
50 3	DJNZ	Jump Double Precision Not Zero	If $(A_{a+1}, A_a)\neq0$, jump to Y	III	N	N	1	2.0
51 0	JP	Jump A Positive	If $(A_a)\geq0$, jump to Y	III	N	N	1	1.5
51 1	JN	Jump A Negative	If $(A_a)<0$, jump to Y	III	N	N	1	1.5
51 2	JZ	Jump A Zero	If $(A_a)=0$, jump to Y	III	N	N	1	1.5
51 3	JNZ	Jump A Not Zero	If $(A_a)\neq0$, jump to Y	III	N	N	1	1.5
52 0	LBJ	Load B and Jump	$P+1\rightarrow B_a$, jump to Y	III	N	N	1	1.8
52 1	JBNZ	Index Jump B	If $(B_a)\neq0$, then $(B_a)-1\rightarrow B_a$, jump to Y	III	N	N	1	1.8
52 2	JS	Jump sy+B	Jump to $sy+(B_b)$	III	N	N	13	1.5
52 3	JL	Unconditional Jump Lower	Jump to the Lower of Y	III	N	N	12	1.5
53 0 a=0	JNF	Jump on No Overflow	If OD is not Set, jump to Y; Clear OD	III	N	N	12	1.5
53 0 a=1	JOF	Jump on Overflow	If OD is Set, jump to Y; Clear OD	III	N	N	12	1.5
53 1 a=0	JNE	Jump on Not Equal	If CD $\neq$, jump to Y	III	N	N	12	1.5
53 1 a=1	JE	Jump on Equal	If CD $=$, jump to Y	III	N	N	12	1.5
53 1 a=2	JG	Jump on Greater Than	If CD $>$, jump to Y	III	N	N	12	1.5
53 1 a=3	JGE	Jump on Greater Than or Equal	If CD $\geq$, jump to Y	III	N	N	12	1.5
53 1 a=4	JLT	Jump on Less Than	If CD $<$, jump to Y	III	N	N	12	1.5
53 1 a=5	JLE	Jump on Less Than or Equal	If CD $\leq$, jump to Y	III	N	N	12	1.5
53 1 a=6	JNW	Jump Outside Limits	If CD Outside Limits, jump to Y	III	N	N	12	1.5
53 1 a=7	JW	Jump Within Limits	If CD Within Limits, jump to Y	III	N	N	12	1.5
53 2	RJ	Return Jump a=o	$P+1\rightarrow Y$, jump to $Y+1$	III	N	N	12	3.0
53 2	RJC	Return Jump a=1, 2, 3	If switch a is Set, $P+1\rightarrow Y$, jump to $Y+1$; otherwise N.I.	III	N	N	1	3.0
53 2*	RJSC	Return Jump a=4, 5, 6, 7	If switch a is Set, Stop; $P+1\rightarrow Y$, jump to $Y+1$ at restart	III	N	N	1	3.75
53 3	J	Manual Jump a=o	Jump to Y	III	N	N	12	1.5
53 3	JC	Manual Jump a=1, 2, 3	If switch a is Set, jump to Y; otherwise N.I.	III	N	N	1	1.5
53 3*	JSC	Manual Jump a=4, 5, 6, 7	If switch a is Set, Stop; Jump to Y at restart	III	N	N	1	2.25
54✓	LCT	Load CMR Task	$(Y)\rightarrow CMR_{ak}$	I	N	Y	3	1.5
55*	LCI	Load CMR Interrupt	$(Y)\rightarrow CMR_{ak+100}$	I	N	Y	3	1.5

REPERTOIRE OF INSTRUCTIONS (CONT.)

Code	Mnemonic	NAME	DESCRIPTION	F	CA	R	UF	Time‡ μs
56√	SCT	Store CMR Task	$(CMR_{ak}) \rightarrow Y$	I	N	Y	3	1.5
57*	SCI	Store CMR Interrupt	$(CMR_{ak+100}) \rightarrow Y$	I	N	Y	3	1.5
60√i=0	HSCT	Store CMR in A	$(CMR_{af4}) \rightarrow A_b$	IV A	N	N	4	1.75
60*i=1	HSCI	Store CMR in A	$(CMR_{af4+100}) \rightarrow A_b$	IV A	N	N	4	1.75
61√i=0	HLCT	Load CMR from A	$(A_b) \rightarrow CMR_{af4}$	IV A	N	N	4	1.75
61*i=1	HLCI	Load CMR from A	$(A_b) \rightarrow CMR_{af4+100}$	IV A	N	N	4	1.75
62	HLC	Shift Left Circularly	(A_a) Left Shifted End Around $\rightarrow A_a$	IV B	N	N	10	1.75
63	HDLC	Shift Left Circularly Double	(A_{a+1}, A_a) Left Shifted End Around $\rightarrow A_{a+1}, A_a$	IV B	N	N	10	1.75
64	HRZ	Shift Right Fill Zeros	(A_a) Right Shifted, Zero Fill $\rightarrow A_a$	IV B	N	N	10	1.75
65	HDRZ	Shift Right Double, Fill Zeros	(A_{a+1}, A_a) Right Shifted, Zero Fill $\rightarrow A_{a+1}, A_a$	IV B	N	N	10	1.75
66	HRS	Shift Right Fill Sign	(A_a) Right Shifted, Sign Fill $\rightarrow A_a$	IV B	N	N	10	1.75
67	HDRS	Shift Right Double, Fill Sign	(A_{a+1}, A_a) Right Shifted Sign Fill $\rightarrow A_{a+1}, A_a$	IV B	N	N	10	1.75
70 0	HSF	Scale Factor	Normalize (A_a) Shift Count $\rightarrow A_b$	IV A	N	N	5	2.25
70 1	HDSF	Double Scale Factor	Normalize (A_{a+1}, A_a) Shift Count $\rightarrow A_b$	IV A	N	N	5	2.25
70 2	HCP	Complement A	$(A_a)' \rightarrow A_a$	IV A	N	N	7	1.1
70 3	HDCP	Double Complement A	$(A_{a+1}, A_a)' \rightarrow A_{a+1}, A_a$	IV A	N	N	7	1.1
71 0	HOR	Logical Sum	$(A_a) \oplus (A_b) \rightarrow A_a$; $(A_b)_i = (A_b)_f$	IV A	N	N	5	1.0
71 1	HA	Sum	$(A_a) + (A_b) \rightarrow A_a$	IV A	N	N	5	1.0
71 2	HAN	Difference	$(A_a) - (A_b) \rightarrow A_a$	IV A	N	N	5	1.0
71 3	HXOR	Logical Difference	$(A_a) \overline{\oplus} (A_b) \rightarrow A_a$	IV A	N	N	5	1.0
71 5	HAND	AND	$(A_a) \odot (A_b) \rightarrow A_a$; $(A_b)_i = (A_b)_f$	IV A	N	N	5	1.0
74 0	HM	Multiply Register	$(A_a) \bullet (A_b) \rightarrow A_{a+1}, A_a$	IV A	N	N	5	7.75†
74 1	HD	Divide Register	$(A_{a+1}, A_a) \div (A_b) \rightarrow A_a$; Remainder $\rightarrow A_{a+1}$	IV A	N	N	5	15.0†
74 2	HRT	Square Root	$\sqrt{(A_{a+1}, A_a)} \rightarrow A_b$; Residue $\rightarrow A_{b+1}$	IV A	N	N	5	15.0†
74 3	HLB	Load B_a with B_b	$(B_b) \rightarrow B_a$	IV A	N	N	5	1.75
74 4	HC	Compare, Register	Compare (A_a) to (A_b), Set CD	IV A	N	N	5	1.1
74 5	HCL	Compare Limits, Register	If $(A_{a+1}) > (A_b) \geq (A_a)$, Set CD in Limit	IV A	N	N	5	1.75
74 6	HCM	Compare Masked, Register	Compare $(A_{a+1}) \odot (A_a)$ to (A_b), Set the CD	IV A	N	N	5	1.1
74 7	HCB	Compare B_b with B_a	Compare (B_b) to (B_a), Set the CD	IV A	N	N	5	2.0
77 0**	HSIM	Store IOC Monitor Clock in A	$(IOC_a$ MON CLK$) \rightarrow A_b$	IV A	N	N	5	3.0
77 1	HSTC	Store Real-Time Clock in A	$(IOC_a$ RTC$) \rightarrow A_b$	IV A	N	N	5	3.5
77 4*	HPI	Prevent Class III Interrupts	Set Class III Interrupt Lockout	IV A	N	N	9	2.25
77 5*	HAI	Allow Class III Interrupts	Clear Class III Interrupt Lockout	IV A	N	N	9	2.25
77 6*i=0	HALT	Stop Processor	Stop CPU (4-Stop); Continue at Restart	IV A	N	N	9	2.25
77 6*i=1	HWFI	Wait for Interrupt	Cease Memory References until Interrupted	IV A	N	N	9	2.25

ULTRA/32 PSEUDO INSTRUCTIONS								
10	ZA	Clear A	$0 \rightarrow A_a$	I	N	Y	7	1.5
20	ZB	Clear B	$0 \rightarrow B_a$	I	N	Y	7	2.0
20	NOOP	No Operation	$0 \rightarrow B_0$	I	N	Y	9	2.0
23	SZ	Store Zeros	$0 \rightarrow \underline{Y}$	I	Y	Y	12	1.5
74 3	HNO	Half Word No Operation	$(B_0) \rightarrow B_0$	IV A	N	N	9	1.75
ULTRA/32 FORMATING MNEMONICS								
—	HK	Half Word Constant (Variable field becomes next halfword)		—	—	—	16	—
—	IW	Indirect Word (c=10)		—	—	—	8	—
—	IWS	Indirect Word, Special Base (c=00, c_1=0)		—	—	—	11	—
—	IWB	Indirect Word, Special Index (c=00, c_1=1)		—	—	—	11	—
—	IWC	Indirect Word, Character (c=01)		—	—	—	14	—
—	IWCI	Indirect Word, Character Increment (c=11)		—	—	—	14	—
—	MP	Memory Protection (see SPR format)		—	—	—	15	—

ULTRA/32 CODING FORMATS (UF) (An Asterisk (*) Preceding y Indicates Indirect Addressing)

No.	Variable Field	No.	Variable Field	No.	Variable Field	No.	Variable Field	No.	Variable Field	No.	Variable Field
1	a, y, k, b, s	4	af_4, b	7	a	10	a, m (shift by m)	11	sy, b	14	y, w, p, b, s
2	a, y, b, s	5	a, b	8	y, b, s		a, b, 1 (shift by B_b)	12	y, k, b, s	15	r, i, or, ow, ia, ir
3	ak, y, b, s	6	a, sy, b	9	None		a, b, 2 (shift by A_b)	13	sy, k, b	16	e

*Privileged **CPU→IOC Instr.—Privileged √Privileged when ak=2X, 6X or 7X

†Execution time independent of overlap operation

‡Times shown assume 1.5 μs memory with operands not in same bank as instructions (overlapped).

REV. 5.71

I/O CONTROLLER COMMANDS

Code	Mnemonic	NAME	DESCRIPTION	UF**	Time μs
10	IB	Initiate Input Buffer on Cj	(y)→CMA* 0+j; Activate Input	1	3.25
11	OB	Initiate Output Buffer on Cj	(y)→CMA* 20+j; Activate Output	1	3.25
12	FB	Initiate External Function Buffer on Cj	(y)→CMA* 40+j; Activate EF	1	3.25
13	XB	Initiate External Interrupt Buffer on Cj	(y)→CMA* 60+j; Activate EI	1	3.25
14 k=0	TIB	Terminate Input Buffer on Cj	Terminate Input { m=0 Suppress	2	3.0
14 k=1	TOB	Terminate Output Buffer on Cj	Terminate Output { Queued Interrupt;	2	3.0
14 k=2	TFB	Terminate External Function Buffer on Cj	Terminate EF { m=1 Allow Queued	2	3.0
14 k=3	TXB	Terminate External Interrupt Buffer on Cj	Terminate EI { Interrupt	2	3.0
15 k=0	IMIR	Set Input Monitor Interrupt Request on Cj	Set Input Monitor Interrupt on Chan j	3	2.5
15 k=1	OMIR	Set Output Monitor Interrupt Request on Cj	Set Output Monitor Interrupt on Chan j	3	2.5
15 k=2	FMIR	Set EF Monitor Interrupt Request on Cj	Set EF Monitor Interrupt on Chan j	3	2.5
15 k=3	XMIR	Set EI Monitor Interrupt Request on Cj	Set EI Monitor Interrupt on Chan j	3	2.5
16 k=0	AIC	Set Input Chain Active on Cj	{ y→Command Address Pointer Field	4	2.5
16 k=1	AOC	Set Output Chain Active on Cj	{ (bits 55-38) of CMA* 20k+j;	4	2.5
16 k=2	AFC	Set External Function Chain Active on Cj	{ Activate Chain	4	2.5
16 k=3	AXC	Set External Interrupt Chain Active on Cj	{	4	2.5
17 m=0	TBZ	Test Bit Zero	If $(y)_{kj}=0$, SKIP; Else NI	7	4.0
17 m=1	TBS	Test Bit Set	If $(y)_{kj} \neq 0$, SKIP; Else NI	7	4.0
20	JIO	Jump to y	y→Command Address Pointer or CAR‡	6	2.5
22	LICM	Load IOC Control Memory	(y)→IOC Control Memory Address kj	5	3.25
23	ILTC	Load Real-Time Clock	(y)→Real Time Clock	6	4.0
24	SICM	Store IOC Control Memory	$(\text{IOC Control Memory})_{kj}$→y	5	2.75
25	IBS	Set Bit	$1 \rightarrow y_{kj}$	5	3.25
26	IBZ	Clear Bit	$0 \rightarrow y_{kj}$	5	3.25
27	ITSF	Test and Set Flag	$1 \rightarrow y_{31}$; If $(y)_{31}$ was Originally Cleared, Skip; Else NI	6	3.25
		FORMATING MNEMONICS			
—	BCW	Buffer Control Word		8	—
—	BCWE	Buffer Control Word ESI		9	—

‡Command Address Register
*Control Memory Address

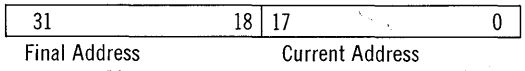
**ULTRA FORMAT

1—j, y, k, c, m	4—j, y, c	7—kj, y	(l=buffer length)
2—j, c, m	5—kj, y, c	8—y, l	
3—j, c	6—y, c	9—y, l, k	

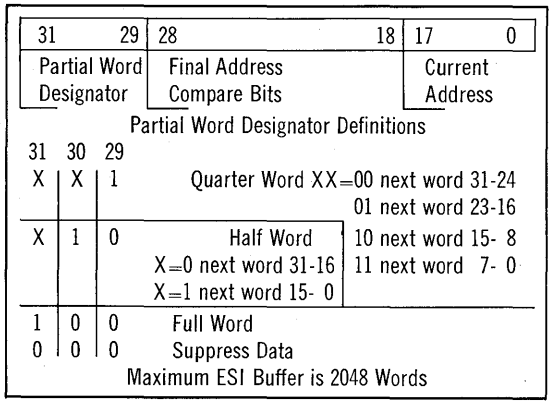
k—DESIGNATOR DEFINITIONS

	k=0	k=1	k=2	k=3
f=10, 11, 13	Suppress data	Pack Quarter word	Pack Half word	Whole word
f=12	Force One Word (y) is EF	One Word Buffer (y) is EF	Multi Word Buffer	Not Used

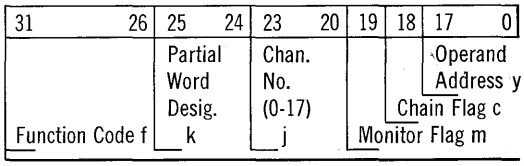
NORMAL BUFFER CONTROL WORD FORMAT



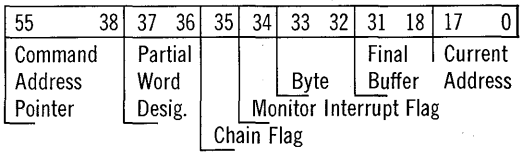
ESI BUFFER CONTROL WORD FORMAT



IOC COMMAND WORD FORMAT



IOC CONTROL MEMORY WORD FORMAT



IOC CONTROL MEMORY ASSIGNMENT

Address	Use
0-17	Input
20-37	Output
40-57	External Function
60-77	External Interrupt

FLOATING POINT FORMAT (each word is one's complement)

Sign Fill ±	14	0

Characteristic (exponent) in A_a or Y

±	30	0

Mantissa in A_{a+1} or $Y+1$

INSTRUCTION WORD FORMATS

Format I

31 26	25 23	22 20	19 17	16	15 13	12 0
f	a	k	b	i	s	y

Format II

31 26	25 23	22 20	19 17	16	15 13	12 0
f	a	f_2	b	i	s	y

Format III

31 26	25 23	22 21	20	19 17	16	15 13	12 0
f	a	f_3	z	b	i	s	y

Format IV A

31 26	25 23	22 20	19 17	16
15 10	9 7	6 4	3 1	0
f	a	f_4	b	i

Format IV B

31 26	25 23	22 16
15 10	9 7	6 0
f	a	m

NORMAL INDIRECT ADDRESS WORD FORMAT

31 30	29 25	24 20	19 17	16	15 13	12 0
c	w	p	b	i	s	y

SPECIAL INDIRECT ADDRESS WORD FORMAT

31 30	29	28 20	19 17	16	15 0
c	c_1	x	b	i	d

f—Function Code
f_2 f_3 f_4—Subfunction Codes
a—Accumulator Register
k—Operand Interpretation
b—Index Register
i—Indirect Bit

c_1—Special Indirect Subfunction
0—$Y=d+(S_b)$
1—$Y=d+(B_b)+(S)$ as specified by $(B_b)_{19-17}$

c—Addressing Designator
00—Indirect Special
10—Indirect Normal
01—Single Character
11—Sequential Character

s—Base Register
w—Field Width
p—Bit Position
y—Operand Address
x—Not Used
d—16 Bit Displacement
z—Not Used—Must be Zero
m—Shift Designator

Bit 2^6	Function
0	Shift by count 2^5—2^0
1	Shift by B_b if $2^5=0$
1	Shift by A_b if $2^5=1$

b is specified by bits 2^3—2^1

REPEAT CONDITIONS

a	Non-Compare Instructions
0	Terminate if A ≠ 0
1	Terminate if A = 0
2	Terminate if A ≥ 0
3	Terminate if A < 0
4	Do not terminate
5	Terminate if (A) is even parity on write into memory
6	Terminate if (A) is odd parity on write into memory
7	Do not terminate

a	Compare Instructions
0	Terminate if CD set to ≠
1	Terminate if CD set to =
2	Terminate if CD set to >
3	Terminate if CD set to ≥
4	Terminate if CD set to <
5	Terminate if CD set to ≤
6	Terminate if CD set to outside limit
7	Terminate if CD set to within limit

FORMAT I INSTRUCTION k—FIELD INTERPRETATION

k	Memory to Arithmetic (Read)	Arithmetic to Memory (Store)
0	sy SE+(B_b)→A 15-0 SE	Not Used
1	(Y_{15-0})→A_{15-0} SE	(A_{15-0})→Y_{15-0}; Y_{31-16}—Un
2	(Y_{31-16})→A_{15-0} SE	(A_{15-0})→Y_{31-16}; Y_{15-0}—Un
3	(Y_{31-0})→A_{31-0}	(A_{31-0})→Y_{31-0}
4	(Y_{7-0})→A_{7-0} ZE	(A_{7-0})→Y_{7-0}; Y_{31-8}—Un
5	(Y_{15-8})→A_{7-0} ZE	(A_{7-0})→Y_{15-8}; Y_{31-16}—Un Y_{7-0}—Un
6	(Y_{23-16})→A_{7-0} ZE	(A_{7-0})→Y_{23-16}; Y_{31-24}—Un Y_{15-0}—Un
7	(Y_{31-24})→A_{7-0} ZE	(A_{7-0})→Y_{31-24}; Y_{23-0}—Un

k—Field Interpretation for Replace Instructions:
Read Cycle—Same as memory to arithmetic.
Store Cycle—Same as arithmetic to memory. For Repeat, with b of repeat instruction not zero, Y will be modified by S_6 and not S_s for store cycle.

SE—Sign Extended; ZE—Zero Extended; Un—Unchanged

SYMBOL DEFINITIONS

CMR—Control Memory Register
F—Format
CA—Character Addressable
R—Repeatable
DSW—Designator Storage Word
UF—Ultra Format
$(A)_n$—Contents of A, bit n
CD—Compare Designator
Y—Address formed by $y+(B_b)+(S_s)$
ICW—Initial Condition Word
$\underline{Y}$—Operand (Y) (Whole word or partial word) or Y, depending on k
⊙—Logical product (AND)
⊕—Logical sum (Inclusive OR)
⊕̄—Logical difference (Exclusive OR)

INTERRUPT STATUS CODES

Class	INTERRUPT	Status Code Bits** 9 8 7 6 5 4 3 2 1 0
I	CP—Operand Memory Resume	0 0 M M M M 0 0 0 0
I	CP—IOC Command Resume	K K 0 0 0 0 0 0 0 1
I	CP—Instruction Memory Resume	0 0 M M M M 0 0 1 0
I	CP—IOC Interrupt Code Resume	K K 0 0 0 0 0 0 1 1
I*	IOC Memory Resume	K K M M M M 1 0 1 0
I*	Intercomputer Timeout	K K C C C C 1 0 1 1
I*	Power Tolerance (never locked out)	0 0 0 0 0 0 1 1 1 1
II*	Interprocessor Interrupt	0 0 0 0
II	Floating Point Error	0 0 0 1
II	CP Illegal Instruction Error	0 0 1 0
II	Privileged Instruction Error	0 0 1 1
	Not Assigned	0 1 0 0
II	Operand Breakpoint Match	0 1 0 1
II	Operand Read or Indirect Addressing	0 1 1 0
	Not Assigned	0 1 1 1
	Not Assigned	1 0 0 0
II	Operand Write	1 0 0 1
II	Operand Limit	1 0 1 0
II	Instruction Breakpoint Match	1 0 1 1
	Not Assigned	1 1 0 0
II	Instruction Execute	1 1 0 1
II	Instruction Limit	1 1 1 0
II*	CP Monitor Clock	1 1 1 1
III*	IOC Illegal CAR Instruction	K K 0 0 P P 0 0 0 0
III*	IOC Illegal Chain Instruction	K K C C C C 0 1 F F
III*	IOC Monitor Clock	K K 0 0 0 0 1 0 1 0
III*	IOC CP Interrupt	K K 0 0 0 0 1 0 1 1
III*	IOC External Interrupt Monitor	K K C C C C 1 1 0 0
III*	IOC External Function Monitor	K K C C C C 1 1 0 1
III*	IOC Output Data Monitor	K K C C C C 1 1 1 0
III*	IOC Input Data Monitor	K K C C C C 1 1 1 1
IV	Executive Return	16 bit code assigned thru program

*Queued

**Definitions: PP—CPU NO. (0-2)
MMMM—Memory Bank (0-17)
CCCC—IOC Channel (0-17)
KK—IOC NO. (0-3)
FF=00—EXT. INT.
01—EXT. FCT.
10—OUTPUT
11—INPUT

MEMORY PROTECTION REGISTERS

Storage Protection Register (SPR)

20	19	18	17	16	15 0
I	OR	OW	IA	IR	R
					Displacement Value
					Use Interrupt B&S Registers during Indirect Addressing*
					Allow Indirect Addressing*
					Allow Operand Writing*
					Allow Operand Reading*
					Allow Instruction Execution*

*Operation Allowed if Bit is Set

Segment Identification Register (SIR)

20	19 17	16	15 0
	SIR_s		SIR_d
			16 Bit Displacement
	Base Register Designator		

BREAKPOINT REGISTER

19	18	17 Comparison Address Bits 0
0	0	—Disabled
0	1	—Instruction address
1	0	—Operand address
1	1	—Instruction and operand addresses

CENTRAL PROCESSOR CONTROL MEMORY ADDRESS ASSIGNMENT

Task Mode		
Address	Use	Bits
0-7	Accumulator (A) registers 0-7	32
10	Unassigned	19
11-17	Index (B) registers 1-7	19†
20-27	Base (S) registers 0-7**	18
30-57	Unassigned (not usable)	—
6x	Breakpoint register**	20
7x	Active status register**	23

Interrupt Mode		
Address	Use	Bits
100-107	Accumulator (A) registers 0-7	32
110	CP monitor clock register	19*
111-117	Index (B) registers 1-7	19†
120-127	Base (S) registers 0-7	18
130-137	Unassigned (not usable)	—
140	ICW—Class I	20
141	DSW—Class I ASR storage	20
142	DSW—Class I interrupt status code	20
143	DSW—Class I P—storage	20
144	ICW—Class II	20
145	DSW—Class II ASR storage	20
146	DSW—Class II interrupt status code	20
147	DSW—Class II P—storage	20
150	ICW—Class III	20
151	DSW—Class III ASR storage	20
152	DSW—Class III interrupt status code	20
153	DSW—Class III P—storage	20
154	ICW—Class IV	20
155	DSW—Class IV ASR storage	20
156	DSW—Class IV interrupt status code	20
157	DSW—Class IV P—storage	20
160-167	Storage Protection Registers (SPR) 0-7	21
170-177	Segment Identification Registers (SIR) 0-7	21

*Clock is Low order 16 bits.

**Not Addressable in the Task Mode. (Privileged instruction error will occur)

†Lower 16 bits used for index and arithmetic functions. Upper three bits used **only** as a base-register designation.

ACTIVE STATUS REGISTER

Bit	Designator	
22-20	Central Processor Identifier	Hardwired
19	State I	
18	State II	
17	State III	
16	State IV	
15	Upper—lower	
14	Class I lockout	
13	Class II lockout	
12	Class III lockout	
11	Base (s) register selector	
10	Accumulator/B register selector	
9	Memory lockout inhibit	
8	Load base enable	
7	Bootstrap mode	
6-4	Programmable spare bits	
3	Fixed point overflow indicator	
2	0—Not equal 1—Equal	
1	0—Less than 1—G.T. or equal	
0	0—Within limits 1—Outside limits	

Bits 9-11 1—Interrupt mode
0—Task mode

SPERRY UNIVAC

PX 5652E
2.73