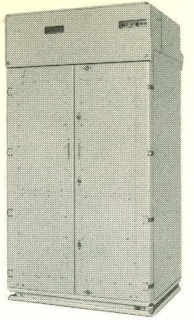


UNIVAC 1206

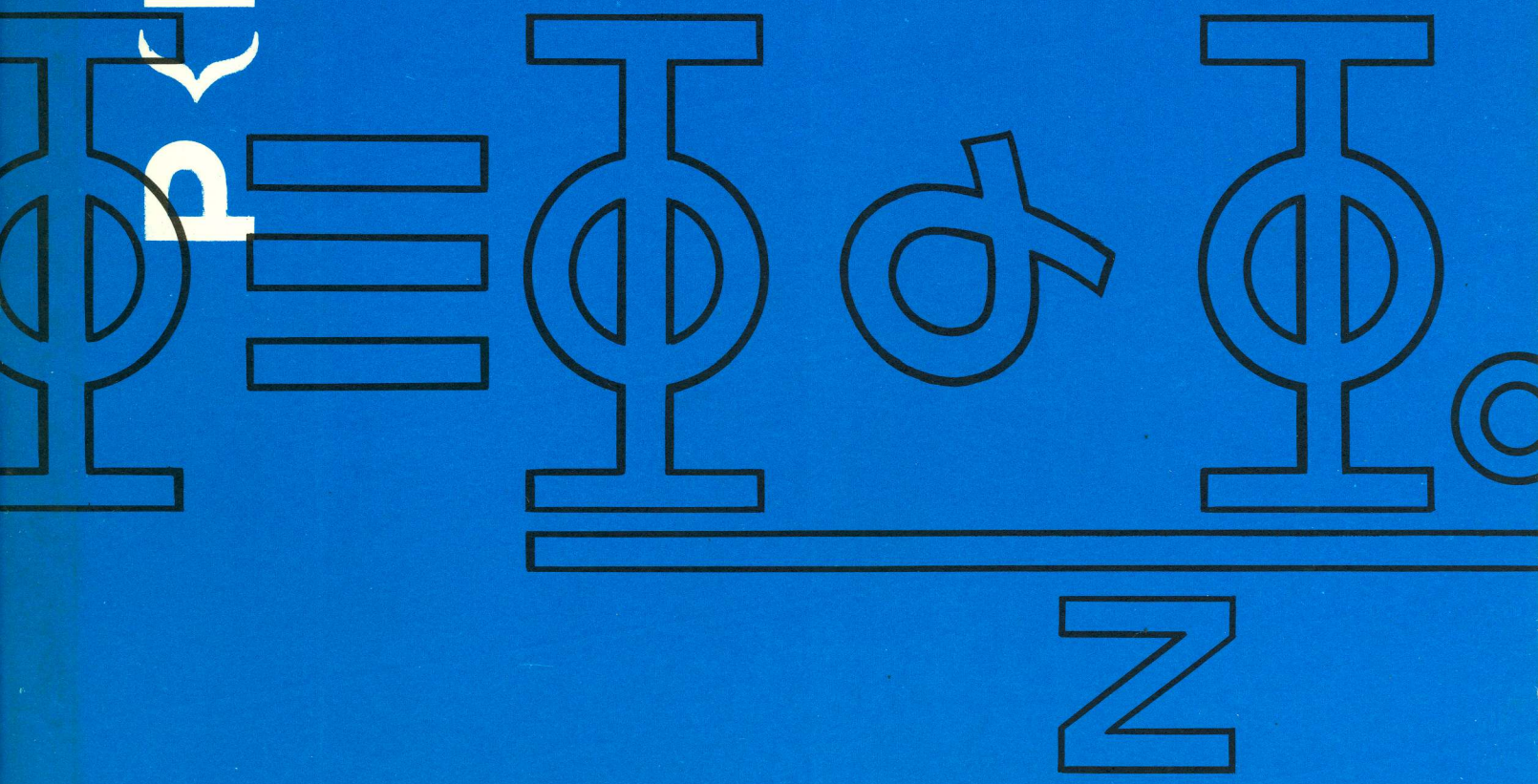
UNIVAC<sup>®</sup>

1206

MILITARY  
COMPUTER



GENERAL DESCRIPTION



**UNIVAC<sup>®</sup>**

**DIVISION OF SPERRY RAND CORPORATION  
MILITARY OPERATIONS · UNIVAC PARK · ST. PAUL, MINN.**

The UNIVAC 1206 Military Real Time Computer and the AN/USQ-20 NTDS Unit Computer are terms that have been used interchangeably.

The designation UNIVAC 1206 has been used by Remington Rand UNIVAC to indicate uses of the computer outside of the Naval Tactical Data System. The actual Military designation of this computer is CP 642A/USQ-20(V) indicating that it is one of a number of components included under the designation AN/USQ-20(V); others include such items as the power supply. USQ-20 is a general term used informally to identify this computer.

The technical specifications included in this report are applicable identically with this computer whether it is called UNIVAC 1206, CP 642A/USQ-20(V), or, loosely, USQ-20. The content of these specifications is nearly identical to that of the original technical Note No. 244, dated 10 October, 1960, titled AN/USQ-20 Unit Computer Characteristics.

**UNIVAC**  
**1206**  
**MILITARY**  
**COMPUTER**

**GENERAL DESCRIPTION AND  
INPUT/OUTPUT SPECIFICATIONS**

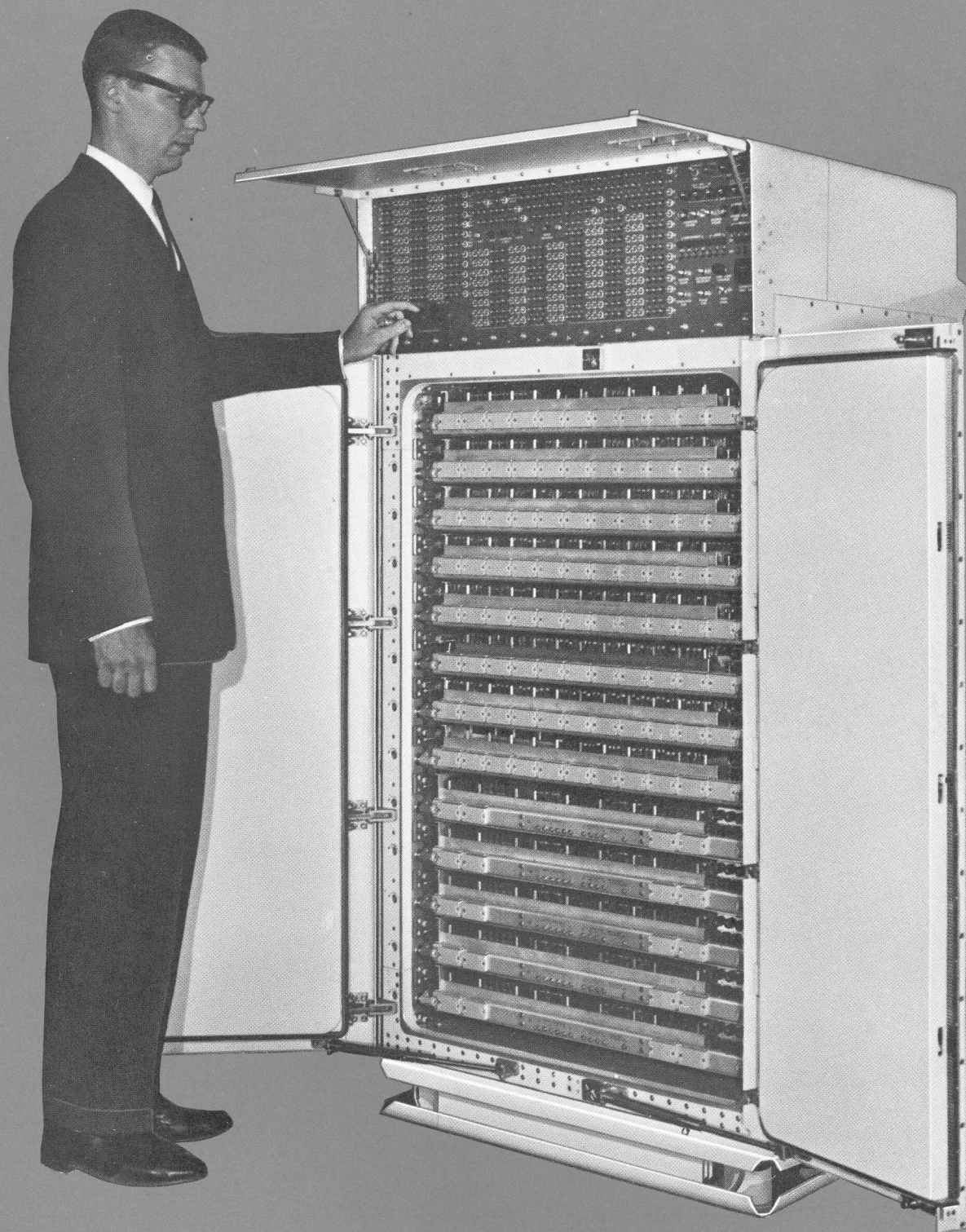
**1206**

# **CONTENTS**

	Page
INTRODUCTION . . . . .	1
GENERAL COMPUTER ORGANIZATION . . . . .	2
OPERATION . . . . .	5
CONSOLE CONTROL . . . . .	8
OTHER FEATURES . . . . .	10
INPUT-OUTPUT SPECIFICATIONS . . . . .	13
PERIPHERAL EQUIPMENT . . . . .	18
INSTALLATION REQUIREMENTS . . . . .	24

## **APPENDIX A – REPERTOIRE OF INSTRUCTIONS AND PROGRAM TIMING**

INTRODUCTION . . . . .	A-1
LIST OF INSTRUCTIONS . . . . .	A-7
TYPICAL INSTRUCTION WORDS . . . . .	A-19



### UNIVAC 1206 SPECIFICATIONS

HEIGHT—72.0 inches—182.8 centimeters. DEPTH—36.9 inches—93.7 centimeters.  
WIDTH—38.1 inches—96.8 centimeters. WEIGHT—2320 pounds—1052 kilograms.

# ILLUSTRATIONS

Figure No.		Page
1	UNIVAC 1206 Military Computer . . . . .	3
2	Computer Cabinet Interior . . . . .	3
3	Logic Module . . . . .	4
4	Tray Assembly . . . . .	4
5	UNIVAC 1206—Simplified Block Diagram . . . . .	6
6	Maintenance and Control Panel . . . . .	9
7	Operating Console . . . . .	9
8	Input-Output Cable Connections . . . . .	13
9	Connections from Computer A to Computer B . . . . .	15
10	Timing for Input Signals . . . . .	16
11	Timing for Normal Output Signals. . . . .	17
12	Timing for External Function Output . . . . .	17

# TABLES

Table No.		Page
1	Summary of Equipment Physical Characteristics . . . . .	23
A-1	UNIVAC 1206—Repertoire of Instructions . . . . .	A-2
A-2	UNIVAC 1206—Interpretation of Instruction Word Designators . . . . .	A-3
A-3	UNIVAC 1206—Core Memory Address Assignments . . . . .	A-6
A-4	CS-1 Compiler—Phase Three . . . . .	A-18

# INTRODUCTION

The UNIVAC 1206 Military Computer is a general-purpose stored-program machine capable of processing very rapidly a large quantity of complex data. Major features of the UNIVAC 1206 Military Computer include the following:

- 1) *Internal high-speed storage with a cycle time of 8 microseconds and a capacity of 32,768 words (16,384 optional);*
- 2) *Repertoire of 62 instructions, most of which provide for conditional program branches;*
- 3) *30-bit word length;*
- 4) *Optional operation with 15-bit half-words;*
- 5) *Internally stored program;*
- 6) *Programmed checking of data parity;*
- 7) *Parallel, ones' complement, subtractive arithmetic;*
- 8) *Single-address instructions with provision for address modification via seven index registers;*
- 9) *Internal 7-day real-time clock for initiating operations at desired times;*
- 10) *12 input and 12 output channels for rapid data exchanges with external equipment without program attention;*
- 11) *2 input and 2 output channels for intercomputer data transfer;*
- 12) *16-word wired auxiliary memory, for storage of critical instructions and constants, which provides facility for Automatic Recovery in event of program failure and for automatic initial loading of programs.*

The following specifications were used as the basis for the design and construction of the UNIVAC 1206.

General Electronic Equipment . . . . .	MIL-E-16400
(Reliability, Simplicity, Materials, Workmanship, Production and Control Inspection, Ease of Operation and Maintenance)	
Temperature . . . . .	MIL-E-16400 Class 4
Test Points and Test Features . . . . .	MIL-T-21322
Repair Parts . . . . .	MIL-E-17362
Enclosure . . . . .	MIL STD 108
Technical Manuals . . . . .	MIL-M-16616
Drawings . . . . .	MIL-D-16415
Preparation for Delivery . . . . .	MIL-P-17555



# GENERAL COMPUTER ORGANIZATION

The UNIVAC 1206 Military Computer is a stored-program computer intended for rapid handling of large quantities of complex data. The computer is especially suitable for real time applications such as missile guidance, range safety, process monitoring, and tactical control and display. Relative to other general-purpose systems, the UNIVAC 1206 Computer emphasizes rapid communication with external devices and large, randomly accessible, internal storage.

Single-address instructions are employed and have an average execution time of 16 microseconds. Instruction words are 30 bits; data words can be either 15 or 30 bits.

Internal storage of the UNIVAC 1206 Military Computer consists of a 32,768-word ferrite core memory. Each word may be interpreted as a single 30-bit word, or as two 15-bit words individually addressed. Control, Arithmetic, and Input-Output sections of the computer each have access to the Storage section. A complete cycle for storage of a 30-bit word received from one of the other sections requires eight microseconds.

Arithmetic and logical operations are performed in the parallel binary mode. In most instances, the result of an operation appears in a 30-bit accumulator register. Arithmetic is ones' complement subtractive with a modulus ( $2^{30} - 1$ ).

Computer operation is controlled by a stored program capable of self-modification. Each program instruction contains a function code (6 bits), instruction operand designator (15 bits), and three execution modifiers (3 bits each). Execution modifiers provide for address incrementation, operand interpretation, and branch-point designation. The operand may be increased by the amount contained in any one of seven index registers. The operand specified by the execution address may be interpreted as a 30-bit quantity, or as a 15-bit half-word with or without sign extension. The next sequential program step may be skipped; it is under control of the content of the Accumulator or the Q-register.

Communication between UNIVAC 1206 Military Computer and its associated external equipment is normally handled by a block transfer of data, with timing under control of the external device. Operating asynchronously with the main computer program, such transfers of data have independent access to storage.

A communication path is established by a sequence of request and response signals between external equipment and computer. Such signals may originate in either the computer or the external device. The main computer program is interrupted by external request signals and a communications channel is established. Once the link has been created, the computer returns to the main program sequence. Block transfer of input or output data then proceeds without program reference until completed.

A total of 14 input and 14 output channels is provided in the computer; each channel consists of 30 parallel lines. Two input and two output (special) channels are reserved for communication with other computers. The maximum possible transfer rate of input or output data over a given channel is greater than 30,000 words per second.

Output channels carry *External Function Words* as well as data words to external equipment. These specify the function desired of the external device. An External Function Word to a tape control unit, for example, may specify *Rewind Tape Unit 2*.

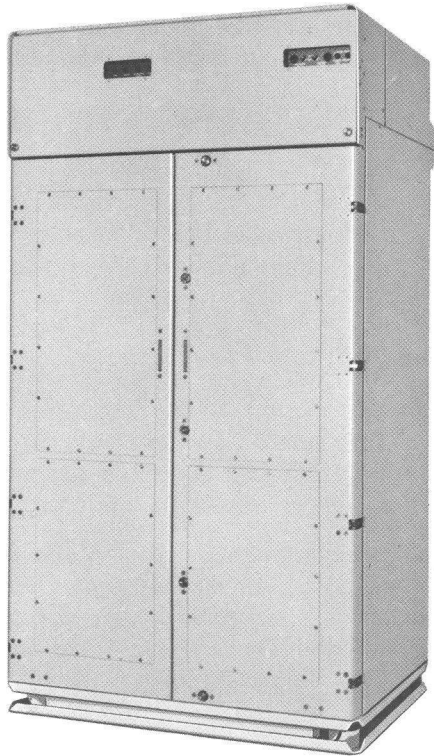


Figure 1. UNIVAC 1206 Military Computer

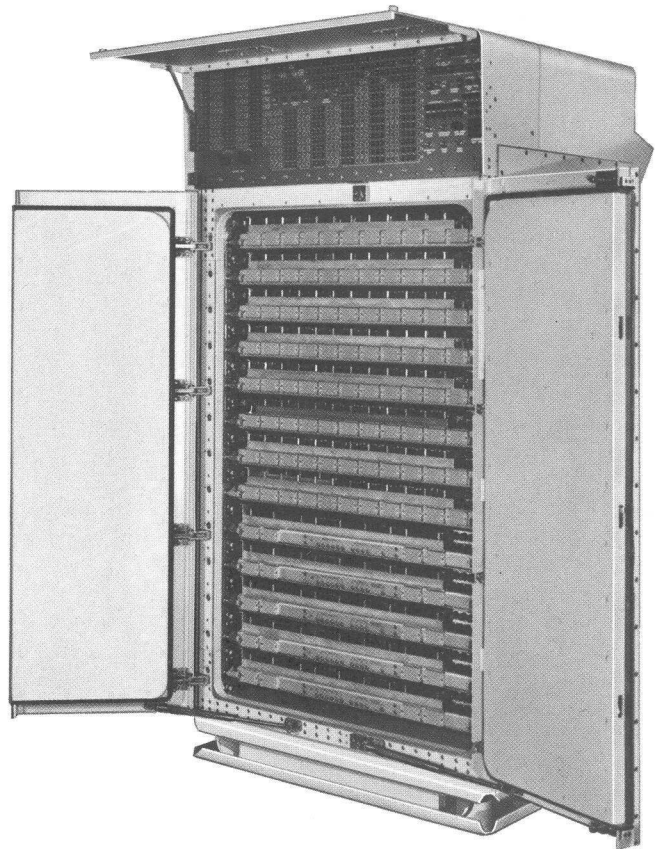


Figure 2. Computer Cabinet Interior

The computer (Figure 1) is housed in a single cabinet, 37 inches deep, 38 inches wide, and 72 inches high. Thirteen trays, eight of logic and five of memory, are horizontally arrayed within the cabinet (Figure 2). Logic modules (Figure 3) are encapsulated printed-circuit cards which plug into the trays (Figure 4). Maintenance test points are readily accessible at the front of the trays.

Computer cabinet doors, closed during normal operation, can be opened for maintenance. The maintenance and control panel, which forms the upper part of the cabinet, contains register indicators, set and clear pushbuttons, and operating switches. A separate operating and maintenance console can be supplied for applications where a remote console is desirable.

Primary power is provided to the computer from a 60-cycle input, 400-cycle output motor-alternator which, in addition to converting frequency, serves to isolate the computer from the main power source. The standard UNIVAC 1206 is equipped with a water-cooled heat exchanger. A plenum which uses ambient air as a coolant is available as an option. Inter-equipment cabling enters the computer at the top of the cabinet and may be run either overhead or through floor ducts.

The computer is designed and constructed to withstand severe shock and vibration. It may be installed aboard ship or in a trailer without modification.

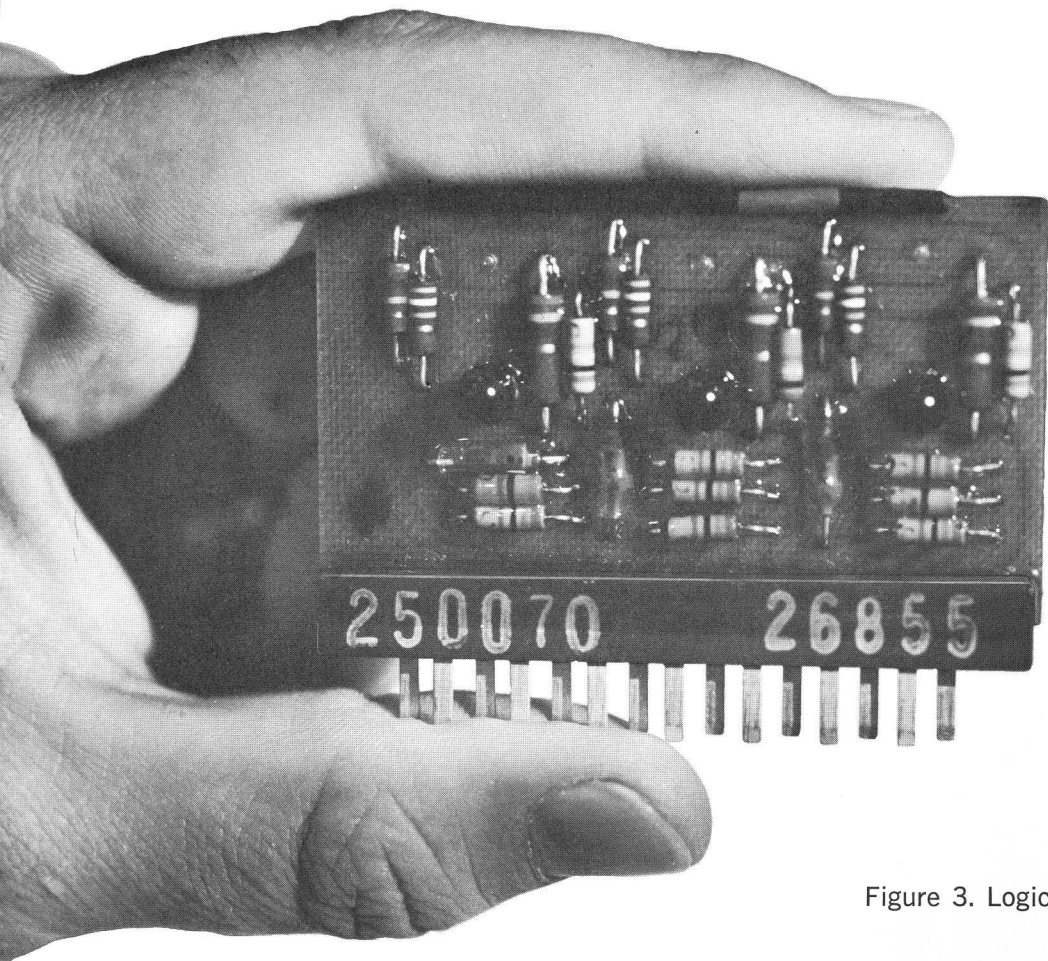


Figure 3. Logic Module

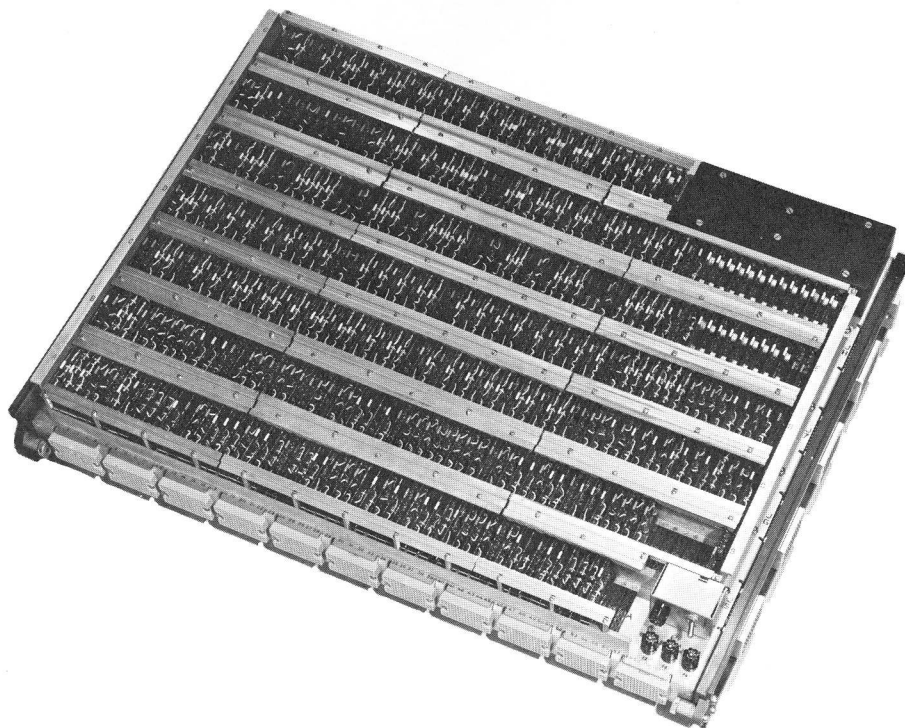


Figure 4. Tray Assembly

# OPERATION

Figure 5 is a simplified block diagram of the UNIVAC 1206 Military Computer. As indicated, the computer consists of four major sections: Input-output, Storage, Arithmetic, and Control. Symbols on the diagram are explained as operation of the various sections is discussed.

## *INPUT-OUTPUT SECTION*

The *Input-output section* includes those data paths and control circuits used by the computer for communicating with external equipment. Main parts of the Input-Output section are 1) two output registers (C0 and C1) and their associated line drivers, 2) fourteen sets of gated input amplifiers, and 3) priority and access control circuits.

### *Output Registers*

The *C0-register* is used for transmissions to all external devices except other computers. As illustrated in Figure 5, C0 receives its input directly from the Storage section via gates controlled by the priority and access circuits. Three sets of 30 line drivers branch from the output of C0; each set drives four output channels. Gated registers located in the external devices determine which channel is active during any particular transmission.

The *C1-register* handles transmissions over the two special output channels—those used to transmit data to other computers. Operation of C1 is similar to that of C0: Words enter C1 from storage via gates controlled by the priority and access circuits and are transmitted over the active channel by a set of 30 line drivers.

The fourteen output channels are numbered from 0 to 15; the intercomputer channels are 0 and 1. If two or more output transmissions are simultaneously requested, the channel with the highest number is granted priority; others follow in order.

### *Input Amplifiers*

A set of 30 *gated amplifiers* is provided for each of the fourteen input channels. Gates are controlled by the priority and access circuits, with the channel having the highest number being given priority if two or more inputs are simultaneously requested. As in the case of the output channels, Channels 0 and 1 are used for intercomputer communication.

This method of treating input data eliminates the need for input buffer registers and gives external equipment direct access to the computer's internal memory.

### *Priority and Access Control*

The Priority and Access Control circuitry implements the assignment of access to the core memory. In the event of two or more simultaneous requests for access to memory, the priority and access circuitry evaluates the nature of the requests and assigns priority on the basis of the following established sequence.

1. Advance Real Time Clock.
2. External Interrupt.
3. Internal Output Monitor Interrupt.
4. Internal Input Monitor Interrupt.
5. Output Request on a normal channel.
6. Input Request.
7. Output Request on an intercomputer channel.

If the same type of request occurs on two or more channels simultaneously, the priority and access circuitry assigns priority in descending order of channel numbers. For simultaneous requests on the intercomputer channels, priority assignment alternates between the two.

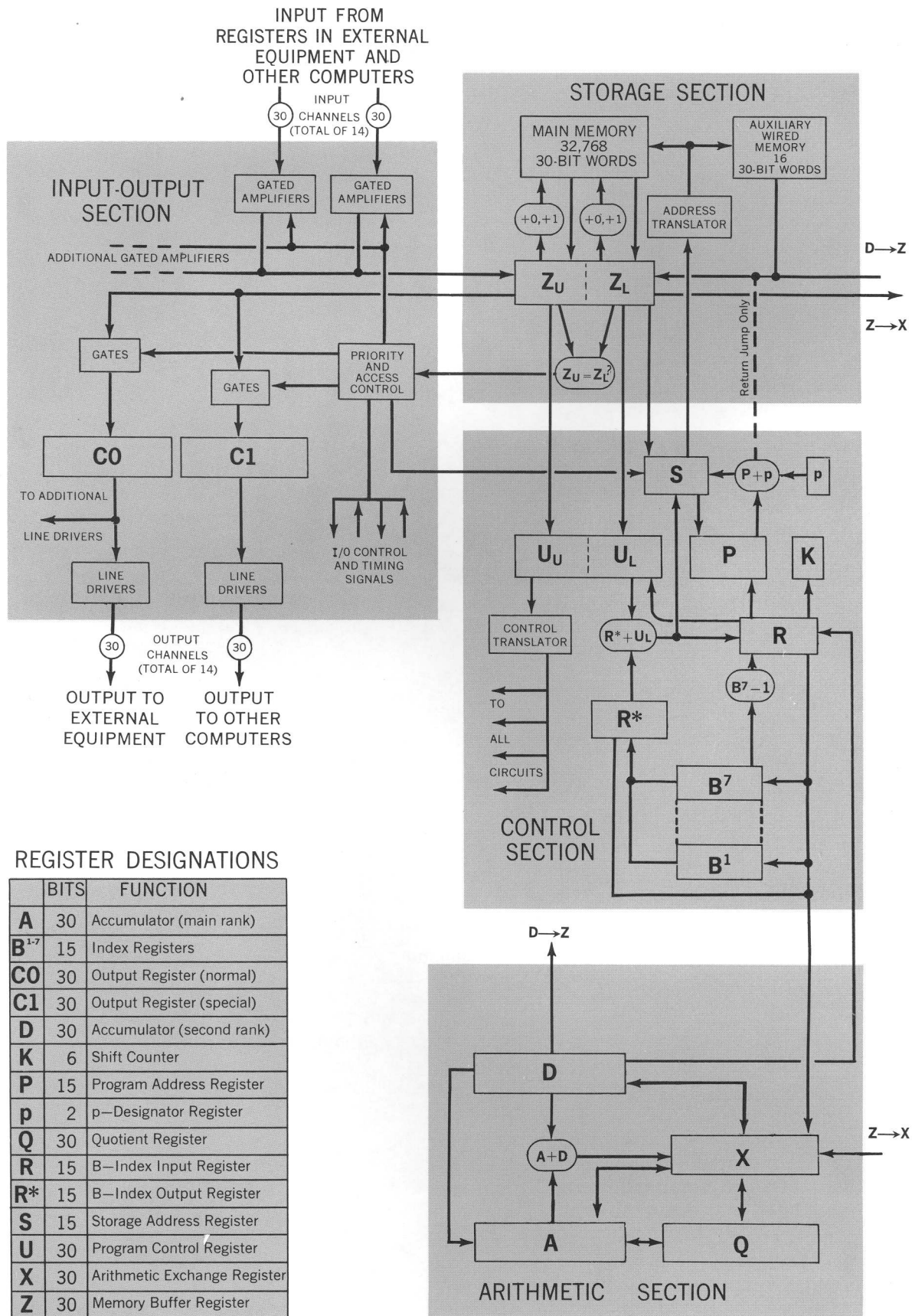


Figure 5. UNIVAC 1206 Simplified Block Diagram.

## STORAGE SECTION

The *Storage section* consists of main memory, wired auxiliary memory, and associated address, transfer, and control circuits.

The main memory, constructed of modular arrays of ferrite cores, has a capacity of 32,768 words of 30 bits each, is coincident-current driven, and is addressed via the address translator. Contents of the location specified by the address is read into the 30-bit Z-register. Because of optional use of 15-bit half-words, Z is split into two 15-bit sections termed Z-upper ( $Z_U$ ) and Z-lower ( $Z_L$ ).

The memory operates in the destructive read-out mode. Time required for the read/restore cycle is eight microseconds.

During the restore portion of the cycle, the contents of  $Z_L$  or  $Z_U$  may be increased by one, as indicated by the +0, +1 modifier boxes. This provision allows for automatically increasing the I/O control words, with the result that memory locations interrogated during a block transfer of data are automatically advanced.

The comparator,  $Z_U = Z_L?$ , is used to detect coincidence between the two halves of the I/O control word. When coincidence occurs, a signal is generated to terminate the I/O transfer.

## ARITHMETIC SECTION

The *Arithmetic section* is that part of the computer which performs numeric and logical calculations. Though greatly simplified, Figure 5 shows the important components of the Arithmetic section: the A-, D-, Q-, and X-registers and the add network.

The *A-register* (30 bits) may be thought of for programming purposes as a conventional accumulator. Because of the logic employed, however, the A-register is actually only the main rank of the accumulator; the *D-register* serves as a second rank. This configuration, while different from former and usual arrangements, permits the use of a much more reliable building-block circuit.

The Add operation is typical of the relationship between the A- and D-registers: The augend and addend are initially contained in A and D. As addition is performed, the sum is formed in parallel by the add network and placed in the *X-register*. From X, the sum is transmitted to A.

The *Q-register* (30 bits) is used principally during multiply and divide operations. The contents of both A and Q may be shifted left or right, individually or as one double-length 60-bit word.

## CONTROL SECTION

The *Control section* consists of those registers and circuits necessary to procure, modify, and execute instructions of the program.

The *U-register* (30 bits) is the program-control register. It holds the instruction word during execution of an instruction. The function code and the various execution modifiers are translated from appropriate sections of the register. The lower-order 15 bits of the U-register have addition properties, modulus  $2^{15} - 1$ .

The 15-bit *B-registers*,  $B^1$  thru  $B^7$ , store the quantities used for address modification. In addition,  $B^7$  stores the count for the repeat mode of operation.

The *R-register* (15 bits) functions as input register for the B-registers. It also has counting properties to increase the contents of the B-registers. The  $R^*$ -register is the B-index output register, it holds the quantity added to the lower-order 15 bits of the U-register during address modification.

The *K-register* (6 bits) functions as a shift counter for all arithmetic operations that involve shifts. The maximum shift count is 63 octal. Multiply and divide operations are controlled by pre-setting the K-register to ZERO and counting operational steps.

The *S-register* (15 bits) holds the storage address during memory references. At the beginning of a storage access period, the address is transferred to the S-register. The contents of the S-register are then translated to activate the storage selection system.

The *P-register* (15 bits) holds the *memory address* of a computer instruction word—that of the current instruction or of a new instruction (e.g., as a result of a jump condition).

The *p-register* (2 bits) adds an increment of +0, +1, or +2 to the contents of the P-register. Output of the adder is the memory address of the *next* instruction and is transferred to the S-register. In the case of a Return Jump instruction *only*, output of the adder is transferred to the Z-register and stored in memory.

## CONSOLE CONTROL

Both the self-contained maintenance and control panel (Figure 6) and the separate (optional) operating console (Figure 7) include indicator lamps that display a detailed report of the internal status of the computer, and controls that allow varied manually governed operations. It is not necessary to monitor the consoles during normal operation.

### REGISTERS

Each register is represented on a console by a row of display lamps, each of which indicates the content of a corresponding register bit-position; a row of SET buttons, each of which can be used to manually enter a *one* into the corresponding bit-position; and a CLEAR button, which can be used to enter zeros manually into all bit-positions of the register. Many of the registers are involved in the mechanics of executing instructions, and are not directly accessible to the program. (These registers are not discussed in this publication.)

### SPECIAL MODES

Both the maintenance and control panel and the separate console are provided with manual controls that permit the following special modes of operation:

- 1) Execution of one program instruction of a sequence for each depression of a switch.
- 2) Execution of consecutive program instructions at a low rate governed by a console frequency control.
- 3) Execution of one master clock phase for each depression of a switch.
- 4) Execution of consecutive master clock phases at a low rate governed by a console frequency control.
- 5) Operation that is normal except that the computer does not stop when it executes a programmed *Stop* instruction. (Such operation is called abnormal high-speed operation.)

The consoles are also provided with a manual control that may be used to disable the real-time clock. This option enables the operator to suspend normal operation temporarily without affecting such operation when it is subsequently resumed. Such suspensions could include stopping the computer or operating temporarily in one of the special modes listed previously.

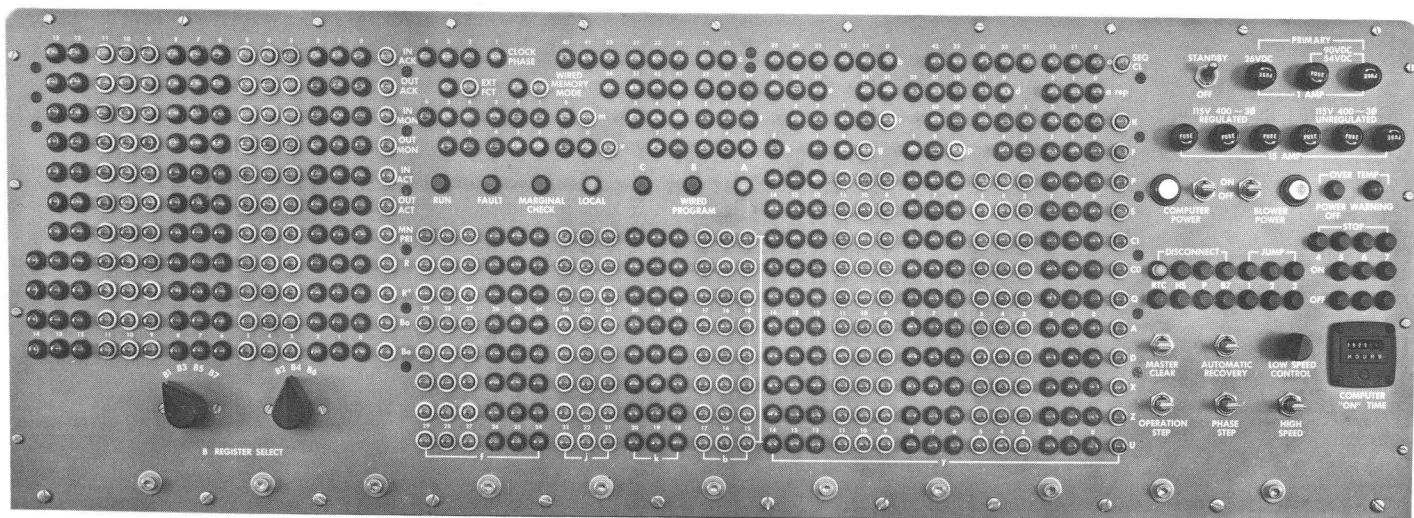


Figure 6. Maintenance and Control Panel

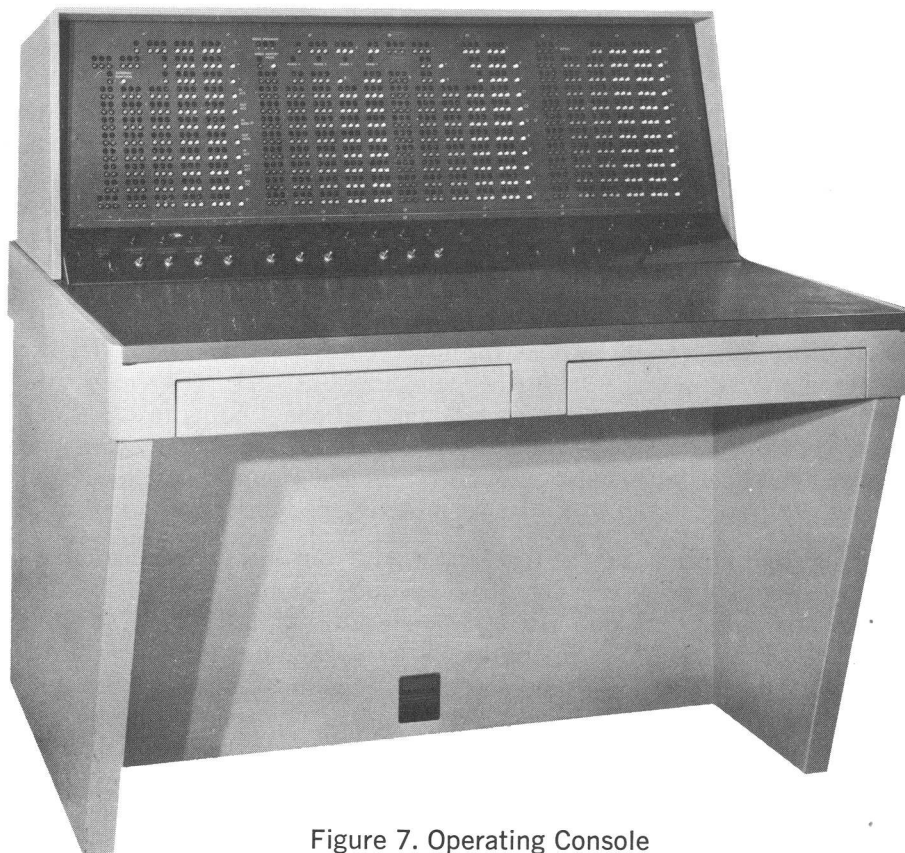


Figure 7. Operating Console



## OTHER FEATURES

The UNIVAC 1206 Military Computer can be used in a wide range of applications, including scientific problems that have previously been limited to very large general-purpose computers and other problems that have ordinarily been solved by business machines. It is especially well suited to real-time control, data processing, and data reduction. Most of the basic features of the computer are described in previous sections. A detailed account of how these features may be used is beyond the scope of this document. This section, however, suggests uses of certain computer features in a few areas.

Features considered here are: 1) automatic programming, 2) floating-point arithmetic, 3) the real-time clock, 4) masked comparison, 5) inclusion of operands in instruction words, 6) detection of special faults, 7) program branching, 8) search operations, 9) half-word and full-word logic, 10) external and internal program interrupts, and 11) 16-word wired auxiliary memory.

### *AUTOMATIC PROGRAMMING*

Automatic Programming for the UNIVAC 1206 computer is in the form of an extensive program processing system that produces, from information coded in an English-algebraic shorthand, a running program complete with memory allocation. There is provision for the detection and correction of errors, editing the program, and debugging the program once it has been placed in operation. There are two distinct coding levels: the AS-1 assembly language, which is computer oriented, and the CS-1 compiling language, which is problem oriented as well as computer oriented. In addition, CS-1 includes an extensive magnetic tape subroutine library.

Programs prepared using AS-1 closely parallel machine coded programs, however, they are readable without reference to charts or tables. As a further advantage, AS-1 language can specify, by a single code, certain operations that require several machine instructions. Coding in AS-1, of course, requires a thorough knowledge of UNIVAC 1206 operation. The system requires 32K core memory and peripheral equipment consisting of a paper tape reader and punch unit and an on-line typewriter or teletypewriter unit.

The CS-1 compiling system includes all the features of AS-1 but has a far broader scope. CS-1 is not limited to machine oriented language, nor does its use require extensive knowledge of computer operation. CS-1 features a number of very powerful *operators* that are totally divorced from machine code. Table A-4 lists the CS-1 operators and indicates the CS-1 programming format. From its extensive *dual* magnetic tape subroutine library, CS-1 has access to numerous computational and manipulative processes. One library portion, called Compiler Library, contains a fixed set of subroutines that control various peripheral equipments when incorporated in the generated program. Such equipment control includes the monitoring typewriter paper tape punch, and the high-speed printer. The other library portion stores subroutines selected by individual programming groups for convenient retrieval. These generally consist of computational routines. Computational routines currently available include *square*

*root, sine, cosine, tangent, arcsine, and arccosine.* Routines are also available for coordinate conversion and for conversion from octal-to-decimal and decimal-to-octal notation.

The CS-1 compiling system operates in core memory with three magnetic tape units for additional storage, and one for a User Library. The compiler produces a variety of listings on a high-speed printer such as: side-by-side (input vs. machine code output) listings, sorted allocations, corrected programs, etc.

Input coding is presented to CS-1 either on 80-column cards or on paper tape. Compiled object programs may be placed either on paper tape or magnetic tape.

### *FLOATING-POINT ARITHMETIC PROGRAM PACKAGE*

The floating-point format is based on a two-word information unit: one mantissa word and one characteristic word. The length of the mantissa word is 28 bits; the length of the characteristic word is 15 bits, including sign bit. It is a three-address system in which four B registers are used to designate operand and operation code. The average instruction time for a floating-point operation (*add, subtract, multiply, or divide*) is 500 microseconds.

### *REAL-TIME 7-DAY CLOCK*

Among the features that suit the computer to real-time problems is the 7-day clock which contains an accurate record of time. The clock may be used to log the receipt times of a periodic real-time input. Each message may be recorded together with its receipt time. Another use of the clock is to initiate periodic programmed operations without requiring more than occasional attention of the main program. Since the clock recycles only once in 7 days, it is suitable for use where the computer is used on an around-the-clock basis.

### *MASKED COMPARISON*

Masked Comparison is used to compare all or any part of a word with the contents of the accumulator. It also tests for *equality*, and *non-equality, greater than, or less than* conditions. In all cases, the original content of the Accumulator is left unchanged.

### *INCLUSION OF OPERANDS IN INSTRUCTION WORDS*

The lower half (15 bits) of an instruction word is commonly used as an operand address. Where 15-bit operands provide acceptable precision, the lower half of the instruction word itself may serve as an operand. This option of storing the operand as part of the instruction word is particularly advantageous for certain applications. It reduces computation by eliminating a memory reference and doubles storage potential.

### *DETECTION OF SPECIAL FAULTS*

Various *fault* conditions can arise during the operation of a running program. For example, an attempt to execute an illegal function code (i.e., 00 or 77) constitutes a fault. Two fault conditions can arise in the execution of a *divide* instruction: the division may be zero or the quotient may exceed 30 bits in length. In each case, the computer can automatically detect the fault and take appropriate remedial action.

### *PROGRAM BRANCHING*

The instruction repertoire of the computer includes provision for program branching that is economical both of memory capacity and instruction execution time. By means of designator bits in the instruction word, it is possible to include in any instruction in a running program, provision for altering the established sequence of instructions without an additional memory reference.

Most instructions can include provision for skipping the next instruction in the regular sequence upon

the occurrence of some specified condition. Other instructions make possible the conditional execution of the instruction at a specified storage location—with or without automatic return to the branch point. Still another type of instruction can establish various iterations of a given instruction. By proper combination of instructions, of course, it is possible to set up numerous compound branch conditions.

### *SEARCH OPERATIONS*

The computer can search internally stored files at very high speeds. The search identifier is entered into the Accumulator, and a mask that identifies the key (a *one* at every key bit position) is entered in Q. The search consists of a masked comparison that is performed repeatedly at successive memory addresses until a *find* (or other terminal condition) is detected.

### *HALF-WORD AND FULL-WORD LOGIC*

All instructions that procure their operands from memory may select the upper 15 bits, the lower 15 bits, or the entire 30 bits as the operand. For some purposes, half-word accuracy is generally sufficient, and this feature can be used to reduce computation time and to double the effective memory capacity.

### *EXTERNAL AND INTERNAL PROGRAM INTERRUPTS*

Provision is made for the interruption of a running program by an event which may occur asynchronously with that program.

Any interrupt discontinues the normal execution of instructions by forcing the execution of the instruction located in a permanently assigned “Interrupt Entrance Register”. For each of the 14 input-output channels there are three Interrupt Entrance Registers.

- External Interrupt Register
- Internal Interrupt Register for Input
- Internal Interrupt Register for Output

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

- To ignore the interrupt, a 60000 00000 instruction will remove the interrupt lockout; and the program will continue with the normal execution of instructions, since the P-register is not affected by the interrupt itself.
- To respond to the interrupt requires a Return Jump to the interrupt routine in the interrupt entrance register. The Return Jump instruction saves the contents of the P-register plus one, which is the address of next instruction that would have been executed in the normal sequence if no interrupt had occurred, rather than the address of the Return Jump instruction + 1. This provides a return to the program that was interrupted at the point of interruption.

An external interrupt results from an external device placing a signal on an External Interrupt line. Appropriate action is generally taken by an interrupt program.

Internal interrupts are generated by the input-output section of the computer whenever a buffer, which has been initiated with a monitor imposed, terminates at the end of the transfer. The interrupt program takes cognizance of the buffer termination, and the main program is resumed.

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

### *16-WORD WIRED AUXILIARY MEMORY*

In addition to the large main memory, a 16-word auxiliary memory is also provided. It is a wired memory and operates in the nondestructive read-out mode. The auxiliary memory is used to contain important instructions or constants. For example, a program-load routine may be stored there to facilitate rapid changes in the main program and automatic program recovery.

# INPUT-OUTPUT SPECIFICATIONS

## GENERAL

Communication with the UNIVAC 1206 Military Computer is carried on in a 30-bit parallel mode. The computer is provided with 14 input channels, and 14 output channels. The output channels are divided into 12 normal and 2 special output channels. External Function Codes are carried over the same 30 lines as are used for output data, but the control signals used with External Function Codes are carried on different lines to indicate the nature of the signals on the 30 lines. The two *special* output channels differ from the normal channels only in timing and control of data transfer. All input-output channels maintain the same electrical specifications.

Note that all references to input or output are made from the standpoint of the computer; that is, *input* is input to the computer and *output* is output from the computer.

## CONTROL COMMUNICATION

The UNIVAC 1206 Military Computer is designed to use a d-c level input-output system. Signals are d-c levels which may be changed upon interchange of control information. Signals may exist for microseconds or days, depending on the nature of the particular task.

It should be noted that the control lines are carried in the same cables as the data lines and have the same voltage levels. Hence, delay times, rise and fall times, and storage times are similar.

## DATA AND CONTROL SIGNALS

Each input and each output channel has its own cable associated with it (28 cables in all). Each cable has 30 data lines plus 3 control lines.

Figure 8 shows the UNIVAC 1206 Military Computer receiving input from Equipment I and sending output to Equipment II. Of course in most cases, both input and output cables will be used on the same peripheral equipment. Only normal output channels are used for output to peripheral equipment. Any

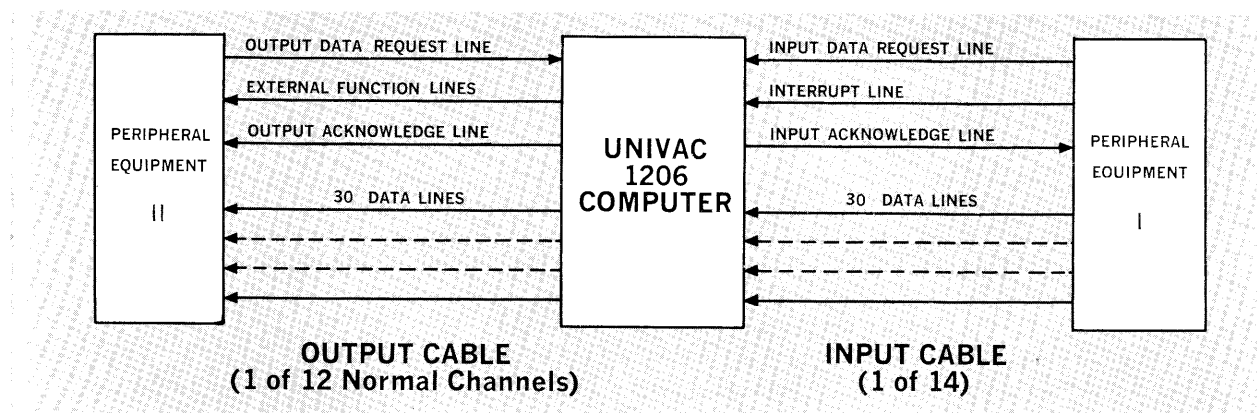


Figure 8. Input-Output Connections

input channel may be used for input from peripheral equipment. Notice the direction of information flow. "Request" and "Interrupt" signals always originate at the peripheral equipment. "Acknowledge" and "External Function" signals always originate at the computer. The Acknowledge signals are always sent from the computer to the peripheral equipment. The third set of control signals, called Interrupt in the input cable and External Function in the output cable is always sent in the same direction as data flow.

### *SEQUENCE OF EVENTS*

The sequence of events for each of four cases of communication between the UNIVAC 1206 Military Computer and peripheral equipment follows:

#### *Normal Input Data Transfer to Computer from Equipment I (Buffer mode).*

- 1) Computer initiates input buffer for given channel.
- 2) Peripheral equipment places data word on 30 data lines.
- 3) Peripheral equipment sets the Input Data Request line to indicate that it has data ready for transmission.
- 4) Computer detects the Input Data Request.
- 5) Computer samples the 30 data lines at its own convenience.
- 6) Computer sets the Input Acknowledge line, indicating that it has sampled the data.
- 7) Peripheral equipment senses the Input Acknowledge line.
- 8) Peripheral equipment drops the data lines and the Input Data Request line.

Steps 2) through 8) of this sequence are repeated for every data word until the number of words specified in the input buffer has been transferred.

#### *Transmission of an Interrupt Code to Computer from Equipment I*

- 1) Peripheral equipment places the Interrupt code on the 30 data lines.
- 2) Peripheral equipment sets the Interrupt line.
- 3) Computer detects the Interrupt.
- 4) Computer samples the 30 data lines.
- 5) Computer sets the Input Acknowledge line, indicating that it has sampled the data.
- 6) Peripheral equipment senses the Input Acknowledge line.
- 7) Peripheral equipment drops the Interrupt code from the data lines and the Interrupt line.

Note that the *Input Acknowledge* is the computer response to either an *Input Data Request* or to an *Interrupt*. To eliminate misinterpretation of the *Input Acknowledge* signal, peripheral equipment must not interrupt until its last *Input Data Request* has been acknowledged by the computer. Under emergency conditions, when data loss is of secondary importance, the *Input Data Request* may be dropped and the *Interrupt* raised a minimum of 100 microseconds later.

When these conditions prevail, an *Input Acknowledge* that occurs after the *Interrupt* is raised will be in answer to the *Interrupt*.

#### *Normal Output Data Transfer from Computer to Equipment II (Buffer Mode).*

- 1) Computer initiates output buffer for given channel.
- 2) Peripheral equipment sets the Output Data Request line indicating that it is in a condition to accept data.
- 3) Computer detects Output Data Request.
- 4) Computer (at its convenience) places information on the 30 data lines.
- 5) Computer sets the Output Acknowledge line, indicating that data are ready for sampling.
- 6) Peripheral equipment detects the Output Acknowledge.
- 7) Peripheral equipment may drop Output Data Request anytime after detecting Output Acknowledge.
- 8) Peripheral equipment samples the 30 data lines.
- 9) Computer drops Output Acknowledge and data lines.

Steps 2) through 9) of this sequence are repeated for every data word until the number of words specified in the output buffer have been transferred.

*Transmission of an External Function Code from Computer to Equipment II.*

- 1) Computer places the External Function Code on the 30 data lines.
- 2) Computer sets the External Function line.
- 3) Peripheral equipment detects the External Function line.
- 4) Peripheral equipment samples the 30 data lines.
- 5) Computer drops External Function Code on the 30 data lines and the External Function line.

**USE OF SPECIAL OUTPUT CHANNELS**

Communication between two computers requires using the two special output channels. Figure 9 illustrates the connections for Computer A to transmit data to Computer B. Another cable using a special output channel of Computer B and an input channel of Computer A would be necessary if Computer B were going to transmit data to Computer A.

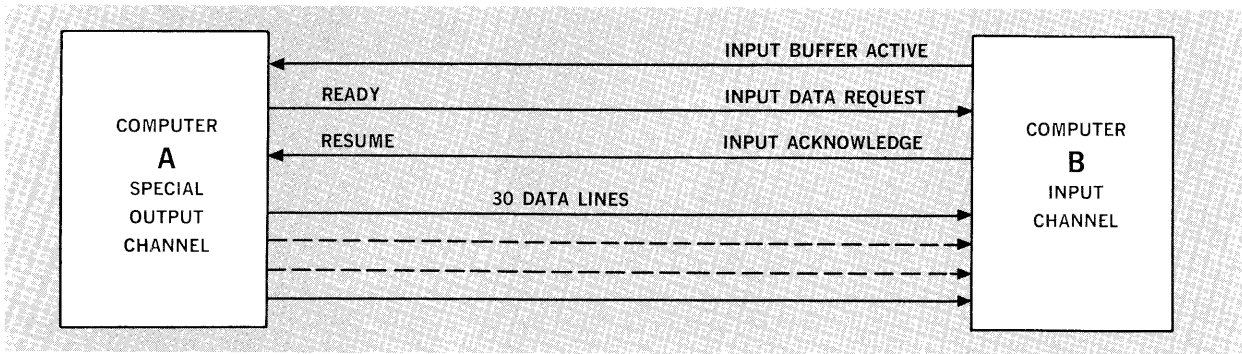


Figure 9. Connections from Computer A to Computer B

Normal transfer of data from computer A to Computer B (Buffer Mode) takes place as follows:

- 1) Computer B sets Input Buffer Active signal.
- 2) Computer A detects Input Buffer Active signal.
- 3) Computer A places data on 30 data lines.
- 4) Computer A sets *Ready* which becomes Input Data Request in Computer B.
- 5) Computer B detects Input Data Request.
- 6) Computer B samples 30 data lines.
- 7) Computer B sets Input Acknowledge line (returned to Computer A as *Resume*).
- 8) Computer A senses *Resume* line.
- 9) Computer A drops data lines and *Ready* line.

Steps 3) through 9) of this sequence are repeated for every data word. Input Buffer Active remains energized during entire transfer of block of words.

**TIMING**

Data lines, when transmitting data from computer to equipment, *must be stable* before being sampled. Hence, a 4.4-microsecond fixed time delay exists between the computer's loading of an output register and energizing of the Output Acknowledge signal or of the External Function signal.

*Input Timing Considerations*

The Input Data Request signal indicates to the computer that data have been placed on the 30 input data lines. To ensure that the data will be accepted, the Input Data Request (or Interrupt) must be

maintained on the lines until an answering Input Acknowledge is received. As shown in Figure 10, there is a 16-microsecond minimum delay between the setting of the Input Data Request and its answering Input Acknowledge. There is no maximum limit stated for the delay, since its value for any particular cycle is determined by the interaction with the computer program and the other input-output channels. The data lines must remain stable as long as the Input Data Request is up.

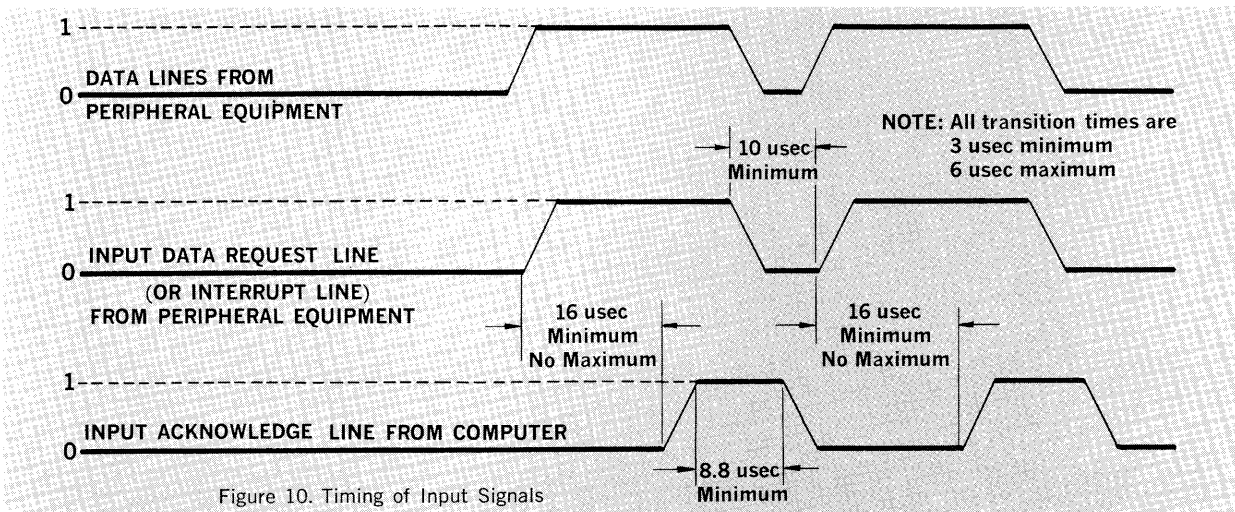


Figure 10. Timing of Input Signals

The Input Acknowledge indicates to peripheral equipment that its 30 data lines have been sampled. The Input Acknowledge signal is set for a fixed time interval. The Peripheral Equipment must be capable of detecting as an Input Acknowledge, a signal which may exist in the stable *one* state for as little as 8.8 microseconds. The Input Data Request (or Interrupt) line and data lines may be dropped to the *zero* state anytime after detecting the Input Acknowledge. The Input Data Request cannot be reset immediately to indicate readiness of a second data word because the computer will not recognize the second Input Data Request unless a minimum time delay of 10 microseconds is allowed between the start of the dropping of the first Input Data Request and the start of the setting of the second Input Data Request. Figure 10 shows that the timing would allow peripheral equipment wishing to transmit data to the computer at a maximum rate to legitimately set the Input Data Request to the *one* state for the second time before the first Input Acknowledge had dropped to the *zero* state. However, this will not affect operation of the cycle since a 16-microsecond minimum delay will again be required between the setting of the second Input Data Request and the setting of the second Input Acknowledge.

### Output Timing Considerations (Buffer Output)

Peripheral equipment must first set the Output Data Request line, indicating that it is in a condition to accept a word of data from the computer. This is necessary because data will be available to the peripheral equipment in a stable state for an interval which may be as short as 17.4 microseconds, if the computer is performing output operations at a maximum rate. Data lines will not necessarily be cleared to the *zero* state before being reset to the *one* state. The minimum time which may elapse between the request and the placement of answering data on the lines is 16 microseconds. The maximum time depends upon the computer program, the priority of the particular channel, and the data rates of the other peripheral equipment.

The Output Acknowledge signal indicates to the peripheral equipment that its requested data are now present on the data lines, which should now be sampled. As shown in Figure 11, the Output Acknowledge will be sent a minimum of 4.0 microseconds after the data has been placed on the lines. The peripheral equipment must be capable of recognizing, as an Output Acknowledge, a signal which may exist in the stable *one* state for as short a time as 8.8 microseconds. The computer will maintain stable data on the lines for a minimum of 4 microseconds after it starts to drop the Output Acknowledge. The Output Data Request may be dropped to the *zero* state anytime after detecting the Output Acknowledge. The Output Data Request cannot be reset immediately to indicate readiness of the

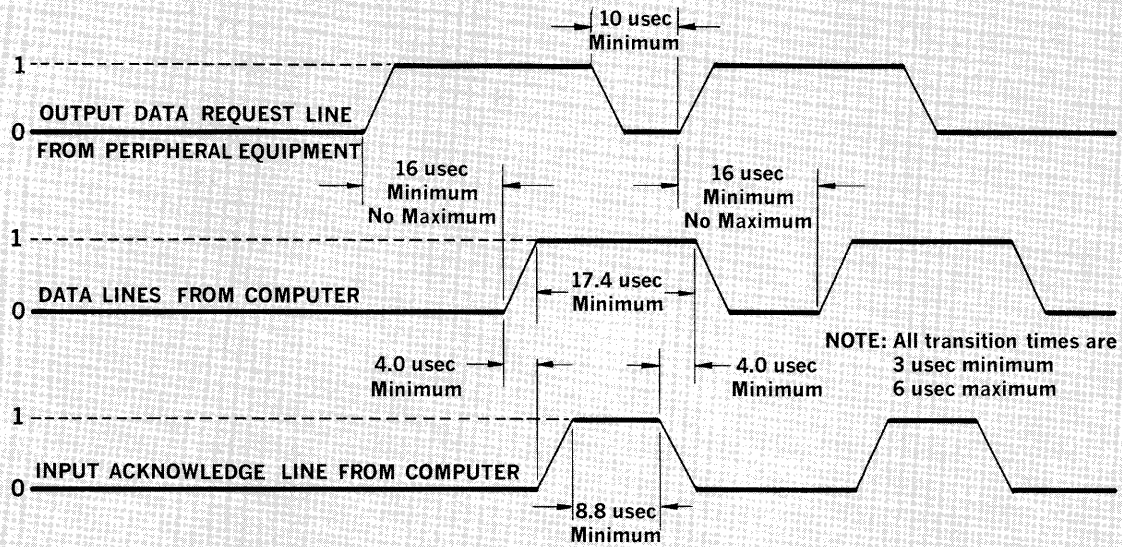


Figure 11. Timing for Normal Output Signals

peripheral equipment to accept a second data word because the computer will not recognize the second Output Data Request unless a minimum time delay of 10 microseconds is allowed between the start of the dropping of the first Output Data Request and the start of the setting of the second Output Data Request. Figure 11 shows that the timing would allow peripheral equipment wishing to receive data from the computer at a maximum rate to legitimately set the Output Data Request to the *one* state for the second time before the first Output Acknowledge had dropped to the *zero* state. However, this will not affect operation of the cycle since a 16-microsecond minimum delay will again exist between the setting of the second Output Data Request and the availability of the second word on the data lines.

### Output Timing Considerations (External Function Output)

External Function output timing is unique in that no response is sent by the peripheral equipment. The computer places the External Function code on the 30 output data lines, and a minimum of 4.0 microseconds later energizes the External Function line. The External Function line indicates to the peripheral equipment that an External Function code is present on the data lines, which should now be sampled. The External Function line will remain in the stable one state for an interval which may be as short as 8.8 microseconds. The External Function code will remain on the data lines for a minimum of 4-microseconds after the External Function signal begins to drop.

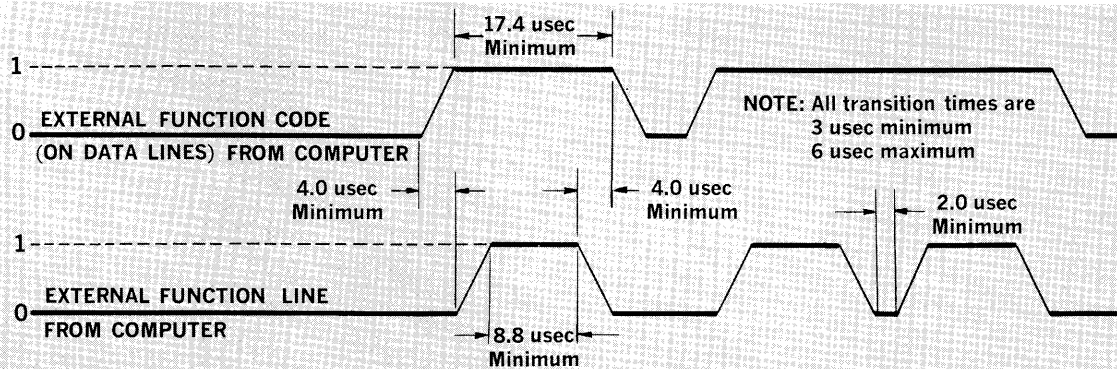


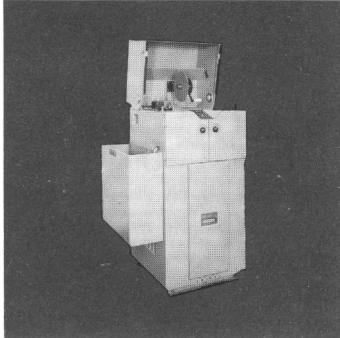
Figure 12. Timing for External Function Output

The peripheral equipment has no control over the rate at which External Function codes are sent. If two External Function instructions to the same peripheral equipment are executed consecutively, the result will appear as in cycles 2 and 3 of Figure 12. The External Function line will drop to the zero state for an interval which may be as short as 2.0 microseconds. If the peripheral equipment cannot accept External Functions at this rate, restrictions must be made in the programming of External Function instructions to the equipment.



# PERIPHERAL EQUIPMENT

The following paragraphs contain brief functional descriptions of the various items of peripheral equipment available for use with the UNIVAC 1206 Military Computer. Table 1 summarizes the physical characteristics of the computer as well as the peripheral equipment.

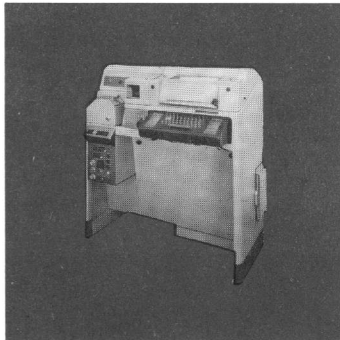


## *PAPER TAPE UNIT*

*Inches:* Height 54.7, Width 22.0, Depth 22.5

*Centimeters:* Height 141.5, Width 55.8, Depth 57.2

The Paper Tape Unit contains within a single enclosure a photoelectric reader, a high-speed punch and the necessary buffering and control circuits to match the interface of the UNIVAC 1206 Computer. The reader operates at 200 frames per second, the punch at 60 frames per second. The unit provides the capability of retrieving anything stored in the computer memory in a form that facilitates manual checking, modification, and subsequent re-entry into the system.



## *TELETYPE\* MODEL ASR-28 AND ADAPTER*

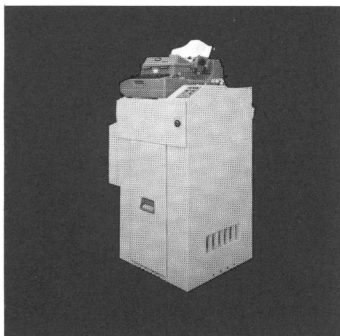
*Inches:* Height 40.5, Width 46.0, Depth 26.5

*Centimeters:* Height 103.0, Width 116.5, Depth 67.3

Teletype\* Model ASR-28, Automatic Send-Receive set, is a code-actuated page printer and 5-level paper tape punch and reader with keyboard input. It is capable of operating at rates of 60, 75, and 100 words (5 characters plus a space) per minute. It can prepare printed copy from keyboard input, remotely generated signals, or 5-level paper tape. It can produce 5-level punched paper tape from keyboard input, remotely generated signals, or from another 5-level tape. It can produce output signals for transmission either from keyboard input or 5-level tape. It can produce printed copy, punched paper tape, and remote output signals concurrently.

Via the Teletype Adapter, the Teletype provides one medium for data entry into, and retrieval from the UNIVAC 1206 computer. The Teletype Adapter contains the logic necessary to match the interfaces of the two pieces of equipment; Teletype signals are serial (one character at a time), whereas the computer transfers data in the parallel mode.

\*Teletype is a trade mark of the Teletype Corporation.

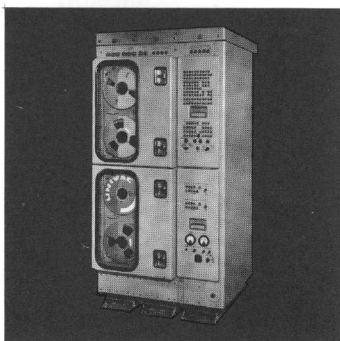


### *MONITORING TYPEWRITER*

*Inches:* Height 48.0, Width 27.3, Depth 25.0

*Centimeters:* Height 122.0, Width 69.2, Depth 63.5

The Monitoring typewriter for use with the UNIVAC 1206 Military Computer is the Flexowriter\*, model FL. Upon appropriate commands from the computer, it can prepare printed copy of any information stored in the computer. Operating independently of the computer, it can prepare simultaneously (if desired) printed copy and seven-level punched paper tape either from keyboard entry or from a compatible paper tape. The seven-level paper tape produced is compatible with the tape used by the 1206 punched paper tape unit. Thus, the Monitoring typewriter, in addition to its monitoring function, is a very useful tool for the preparation and editing of paper tapes.



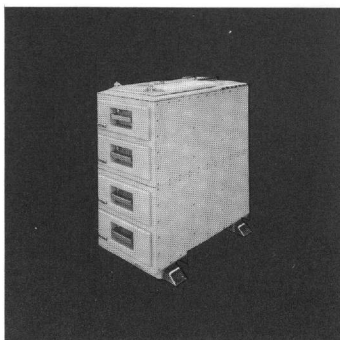
### *MAGNETIC TAPE UNIT*

*Inches:* Height 72.0, Width 35.4, Depth 31.4

*Centimeters:* Height 183.0, Width 90.0, Depth 79.6

The Magnetic Tape Unit consists of two magnetic tape transports and associated control unit in a single cabinet. The control unit matches the interface of the UNIVAC 1206 computer. Data recording density is 200 seven-channel frames per inch on standard Mylar\*\* tape, 2400 feet long and one-half inch wide. Tape speed during read/write operation is 112.5 inches per second which yields a character transfer rate of 22.5 kc. The tape transports provide optical loadpoint and end-of-tape sensing.

The Magnetic Tape Unit provides a convenient means of transferring large amounts of stored data to and from the computer memory. The capacity of the unit is such that this can include program and parameter back-up. Data transfer to and from magnetic tape are asynchronous with respect to normal computer operation and is accomplished in the request/acknowledge mode.



### *MODULE TESTER*

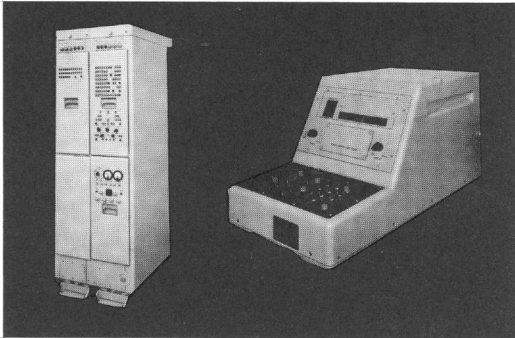
*Inches:* Height 37.3, Width 18.8, Depth 41.0

*Centimeters:* Height 93.3, Width 47.7, Depth 104.1

\*Flexowriter is a trade mark of Friden, Inc.

\*\*Mylar is a trade mark of the E. I. DuPont Nemours and Company, Inc.

The printed circuit module tester consists of a cabinet housing control panel, test circuits, power supplies, storage space for adapter plugs, and operation manuals. The module tester performs dynamic operational tests under simulated operating conditions. Approximately 140 adapter plugs, used to test the 1206 printed circuit modules, are furnished in addition to 14 blank adapter plugs. Necessary tools are also included for wiring the blank adapter plugs.



### *KEYSET SYSTEM*

*Inches:* Height 72.0, Width 24.2, Depth 31.4

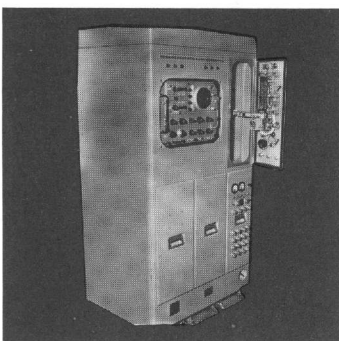
*Centimeters:* Height 183.0, Width 61.4, Depth 79.6

The Keyset System employs two types of equipment—Keysets and a Keyset Central. The Keyset is a general purpose manual digital data entry device. It includes a number of keys for the assembly of data entries, a readout device to monitor data entries during assembly, an Entry key to signal the presence of data to Keyset Central, a fault indicator, and a Clear key. It is readily adaptable to a variety of input functions by the use of appropriate keyboard overlays. Keyset Central is a scanning/switching device that can accept inputs from up to 48 Keysets. It matches the UNIVAC 1206 computer interface and provides error-detection capability in the form of automatic parity checking. Under computer control it sequentially scans all Keysets for the presence of data, transmits any such data to the computer, and clears the Keyset when the computer accepts the data.

Keyset Central can transfer data to the computer from digital sources other than Keysets. It can also transfer data from analog devices via 14 special analog-to-digital conversion channels. Any data entry requires preplanning so that the computer program will recognize and accept the entry as a valid input.

Unlike many forms of data entry, the Keyset System permits data transfer to the UNIVAC 1206 computer during an operational program without stopping the program. Typical functions that exploit this feature include:

- Enter and update data.
- Control the operation of communications links.
- Direct the computer to shift operational modes.
- Manually override normal computer decisions under abnormal conditions.



### *VIDEO PROCESSOR*

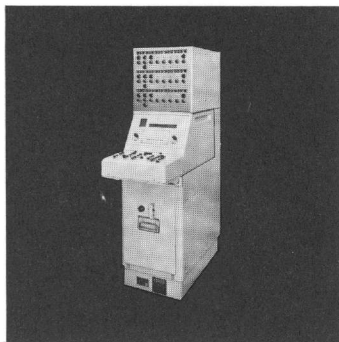
*Inches:* Height 72.0, Width 35.3, Depth 31.8

*Centimeters:* Height 183.0, Width 89.5, Depth 80.6

The Video Processor accepts raw radar video and converts it to a form suitable for further processing by the UNIVAC 1206 computer. Under control of the system program it can discriminate against noise,

clutter, and other unwanted signals.

The Video Processors now in production at UNIVAC operate only with specific radar sets. Operation with other radars does not require extensive modification of present equipment.

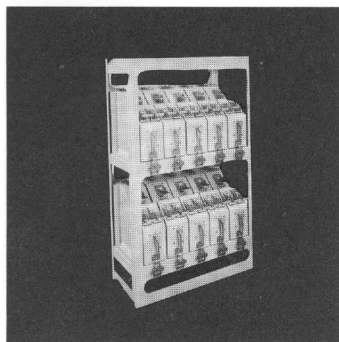


### *SYSTEM MONITORING PANEL*

*Inches:* Height 49.1, Width 13.0, Depth 38.8

*Centimeters:* Height 124.7, Width 33.0, Depth 98.3

The System Monitoring Panel provides manual computer control in a multi-computer complex. It enables an operator to control the entry of various selected programs and monitor intercomputer data transfer. It permits some degree of program error analysis by providing displays of certain pertinent information.

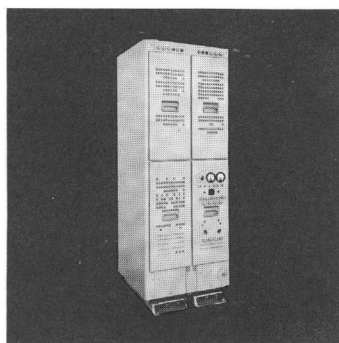


### *INTERCONNECTION PANEL*

*Inches:* Height 52.0, Width 31.0, Depth 15.0

*Centimeters:* Height 132.1, Width 78.7, Depth 38.1

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

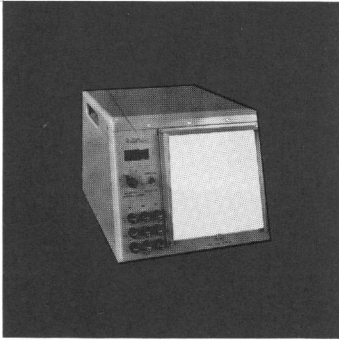


### *TERMINAL EQUIPMENT LOGIC*

*Inches:* Height 72.0, Width 24.2, Depth 31.4

*Centimeters:* Height 183.0, Width 61.4, Depth 79.6

The interface between the 1206 computer and the communications terminal equipment is a unit called TEL (Terminal Equipment Logic). It can operate with either of two data links but not both simultaneously. Under computer control, it establishes and maintains net control and synchronization and accomplishes the automatic transmission of all data between sites. TEL also includes error detection and control-word recognition features.



### *NET PROGRAM DISPLAY*

*Inches:* Height 10.8, Width 13.0, Depth 14.4

*Centimeters:* Height 27.4, Width 33.0, Depth 36.5

The Net Program Display is a small alphanumeric display device that operates in conjunction with the Terminal Equipment Logic. Normally it displays the address of the station last called; if the called station misses its call, it displays the words MISSED CALL.

The UNIVAC 1206 can also use the following UNIVAC 490 /1107 standard peripheral equipment:

Model 46 High-Speed Printer (600 lines per minute)

Model 45 Card Reader (600 cards per minute)

Model 67 Card Punch (150 cards per minute)

FH-880 Flying Head Magnetic Drum (approximately 4½ million characters)

Uniservo IIA Magnetic Tape Subsystem

Table 1. Summary of Equipment Physical Characteristics

EQUIPMENT	DIMENSIONS						UNIT WEIGHT		UNIT POWER REQUIREMENTS (Watts)		COOLING REQUIREMENTS
	Inches			Centimeters			POUNDS	KILOGRAMS	115 volt 60 cycle single phase	115 volt 400 cycle 3-phase	
	HEIGHT	WIDTH	DEPTH	HEIGHT	WIDTH	DEPTH					
UNIVAC 1206 Military Computer Motor-Generator Set*	72.0	38.1	36.9	182.8	96.8	93.7	2320	1052		2000*	Water** at 65F ±5F, (Gallons per min.)
MG Set Controller	25.3	31.4	29.4	64.2	79.6	74.5	1000	454			
Maintenance Console	58.5	19.5	16.8	148.5	49.5	42.5	316	144			
	48.0	51.0	31.5	121.9	129.5	80.0	359	163		100	
Paper Tape Unit	54.7	22.0	22.5	141.5	55.8	57.2	260	118	500	200	
Teletype & Adapter	40.5	46.0	26.5	103.0	116.5	67.3	305	139	200	300	
Monitoring Type- writer (Alternate)	48.0	27.3	25.0	122.0	69.2	63.5	300	137	100	150	
Magnetic Tape Unit	72.0	35.4	31.4	183.0	90.0	79.6	1400	636	2000	1400	3.8
Module Tester	37.3	18.8	41.0	93.3	47.7	104.1	400	181		120	
Keypad Central	72.0	24.2	31.4	183.0	61.4	79.6	960	436		1400	1.7
Keypad Universal	28.0	12.7	13.0	71.1	32.2	33.0	95	43		200	
Video Processor	72.0	35.3	31.8	183.0	89.5	80.6	1500	681	500	1600	3.7
System Monitoring Panel	49.1	13.0	38.8	124.7	33.0	98.3	400	182		240	
Interconnection Panel	52.0	31.0	15.0	132.1	78.7	38.1	250	114			
Terminal Equipment Logic	72.0	24.2	31.4	183.0	61.4	79.6	995	452		1400	1.7
Net Program Display	10.8	13.0	14.4	27.4	33.0	36.5	50	23		28	

\*To isolate the computer from line transients, the MG set supplies 2500 watts of 115 volts, 400 cycle, 3-phase power to operate the computer—this does NOT include the 2000 watts required to operate the blowers. The MG set, as normally supplied, operates from 440 volts, 60 cycle, 3-phase—it is available, however, to operate from any standard voltage.

\*\*Only for the items indicated; remainder of the equipment shown is cooled by Forced Air at 70°±5F. Optionally the items indicated as being water-cooled can also be supplied equipped for Forced Air cooling.

# INSTALLATION REQUIREMENTS

A motor alternator rated at 8.5 kva furnishes regulated, transient-free 115v 400cycle 3-phase power to the computer. Units are available to operate from the following voltages:

440v 60cycle 3-phase (3-wire)  
208v 60cycle 3-phase (4-wire)  
208v 400cycle 3-phase (4-wire)

An additional two kva of 115v line-to-line 400cycle 3-phase unregulated power is required to operate the cooling fan. This power may be supplied by the computer generator in installations where the total equipment load does not exceed the rated capacity of the generator. Although the peripheral equipment operates from the same voltage as the computer it should not be connected to the same generator as the computer. This precaution is necessary to prevent transients, resulting from on-off operation of peripheral equipment, reaching the computer memory.

The computer has been designed to be cooled by either water or air. The standard configuration is a water-cooled plenum using approximately seven gallons per minute of  $70^{\circ} \text{ F} \pm 5^{\circ}$  water. An optional air-cooled plenum is available for use in shelters, such as trailers, helihuts, and fixed land-based shelters. Such shelters must be air conditioned ( $80^{\circ} \text{ F}$  maximum).

Signal interconnection of the units comprising a 1206 system is accomplished through interconnecting cables terminated in connectors which are plugged into mating receptacles at each unit. Power enters the computer through a standard AN connector.

Signal cable of three general types is available to meet various environmental requirements:

Special purpose 40-pair armored cable in accordance with UNIVAC specification DS5177, intended for shipboard service. It conforms as far as possible to Specification MIL-C-915, and has been approved by BuShips for use in the Naval Tactical Data System.

Heavy-duty neoprene jacketed signal cable in accordance with UNIVAC specification DS5215, designed for interconnection of land-based modular shelters such as helihuts or trailers. It is designed to withstand severe flexing and impact loads such as the passage of heavy vehicles over it, and conforms to Specification MIL-C-13777.

Polyvinylchloride-jacketed 40-pair shielded cable in accordance with UNIVAC specification DS5192, intended for installations where severe environmental conditions will not be experienced. It is smaller in diameter than the types listed above, is more flexible, and is considered adequate for land-based installations in the interior of shelters where the ability to withstand severe mechanical abuse is not required.

Special power cable is not required. A standard cable conforming to procuring activity requirements can be selected.

The 1206 dimensions are governed by general shipboard requirements of minimum floor area and maximum height. Dimensions are within the 72-inch maximum permitted by Specification MIL-E-16400. Minimum overhead clearance is 84 inches to permit access to the interconnecting cable receptacles located on top of each unit. The computer design provides front access for normal maintenance; but requires clearance on each side equal to the door width to enable the doors to be swung open for chassis removal.

# APPENDIX A

## REPERTOIRE OF INSTRUCTIONS AND PROGRAM TIMING

### INTRODUCTION

This appendix presents the instruction repertoire for the UNIVAC 1206 Military Computer; details presented apply primarily to computer programming. The UNIVAC 1206 is a self-modifying, stored program, one-address computer. Although this means that one reference or address is provided for the execution of an instruction, this reference can be modified automatically during a programmed sequence. Instructions to the UNIVAC 1206 Computer are in the form of 30-bit instruction words stored sequentially in the magnetic core memory; the computer executes the instructions sequentially unless an instruction alters the sequence.

By means of five designators, a computer instruction word completely defines a computer operation. The highest-order six bits of an instruction word constitute the function code designator, **f**. There are sixty-two function codes each of which corresponds to a machine instruction. The balance of the upper half of the instruction word contains, in the order indicated, the designators **j**, **k**, and **b**. The lower half of the word contains the operand designator, **y**.

The expression  $Y = y + (B^b)$  relates the **b**- and **y**-designators. The three bits of **b** specify (the address of) the B-register whose *contents* increment **y**; for **b** = 0,  $Y = y$ . The precise meaning of **Y**, for a given instruction word, depends on the function code designator and the value of the operand interpretation designator, **k**.

Normally, the **j**- and **k**-designators are three bits each, however, an input-output instruction requires *four* bits to specify one of the fourteen input channels. This is the four-bit  $\hat{j}$  designator; it is used exclusively with input-output instructions. The resulting operand interpretation designator,  $\hat{k}$ , contains only *two* bits.

Table A-1 briefly describes the operation that corresponds to each of the function codes and indicates the execution times that apply. Table A-2 summarizes the interpretation of **j**, **k**, and  $\hat{k}$  with respect to the various function codes. In a particular application of the **j**-designator, not indicated in Table A-2, **j** specifies (the address of) a B-register in each of the four B-box instructions, 12, 16, 71, and 72.

### FUNCTION CODE DESIGNATOR—**f**

The **f** designator (6 bits) appears in bit-positions 29 through 24 of the U-register, or an instruction, designating the function to be performed by that instruction. All values of **f** other than 00 and 77 are defined in the instruction list. The two codes 00 and 77 are *fault conditions* which, if executed, will cause a *fault interrupt*. This results in an interrupt jump to core address 00000, the Fault Entrance Register, or address 00 of *wired* memory, depending on the Automatic Recovery Switch setting (see page A-5).

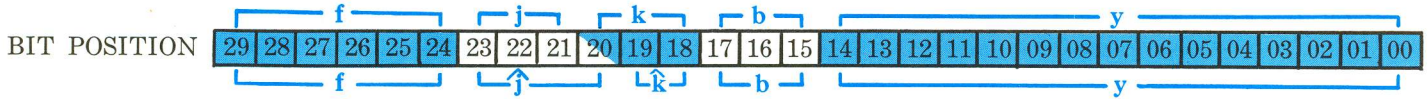
### BRANCH CONDITION DESIGNATOR—**j**

The **j** designator (3 bits) appears in bit-positions 23, 22, and 21 of the U-register, or an instruction; it is used in a majority of the instructions. There are three primary categories of use: 1) for Jump, Skip, and Stop determination, 2) for B-register specification, and 3) for repeat status interpretation. Appropriate interpretations of the **j** designator are listed either below or under the descriptions of the individual instructions.



Table A-1. UNIVAC 1206 Repertoire of Instructions

BIT ALLOCATION FOR NORMAL INSTRUCTION WORD



BIT ALLOCATION FOR INPUT-OUTPUT INSTRUCTION WORD

FUNCTION CODE, f	MACHINE INSTRUCTION  (Boldface type indicates appropriate CS-1 mono-code)	BRIEF DESCRIPTION OF OPERATION PERFORMED BY THE INSTRUCTION	INSTRUCTION EXECUTION TIME,** Microseconds					
			Single Instruction (or first execution repeat mode)			Repeat Mode		
			k-designator			k-designator		
			0, 4	7	Other	0, 4	7	Other
00	Illegal Code	Fault Interrupt						
01	Right SHift ● Q	Shift (Q) right by Y	9.6/	11.2/	16.0	9.6	8.0	11.2
02	Right SHift ● A	Shift (A) right by Y	12.8	14.4	↓	↓	↓	↓
03	Right SHift ● AQ	Shift (AQ) right by Y	11.2/16.0 16.0/20.8			↓	↓	↓
04	COMpare ● A, ● Q, ● AQ	*Sense j; (A) <sub>i</sub> = (A) <sub>f</sub>	12.8	11.2	16.0	↓	↓	↓
05	Left SHift ● Q	Shift (Q) left by Y	9.6/12.8			↓	↓	↓
06	Left SHift ● A	Shift (A) left by Y	↓			↓	↓	↓
07	Left SHift ● AQ	Shift (AQ) left by Y	11.2/16.0 16.0/20.8			↓	↓	↓
10	ENTer ● Q	Y → Q	11.2	9.6	16.0	8.0	6.4	9.6
11	ENTer ● A	Y → A	↓	↓	↓	↓	↓	↓
12	ENTer ● B <sup>j</sup>	Y → B <sup>j</sup>	8.0	↓	↓	4.8	↓	↓
13	EXternal FunCTion on ● C <sup>j</sup>	Enable channel j	12.8	↓	24.0	↓	↓	↓
14	SToRe ● Q	(Q) → Y	↓	↓	16.0	6.4	↓	9.6
15	SToRe ● A	(A) → Y	↓	↓	↓	↓	↓	↓
16	SToRe ● B <sup>j</sup>	(B <sup>j</sup> ) → Y	↓	↓	↓	↓	↓	↓
17	SToRe ● C <sup>j</sup>	(C <sup>j</sup> ) → Y	↓	↓	↓	↓	↓	↓
20	ADD ● A	(A) + Y → A	11.2	9.6	↓	8.0	6.4	9.6
21	SUBtract ● A	(A) - Y → A	↓	↓	↓	↓	↓	↓
22	MULTiply	(Q) Y → AQ	35.2 to 112			31.2 to 108		
23	DIVide	*(AQ)/Y → Q; Remainder → A	112	↓	112	108	↓	108
24	RePLace ● A + Y	(A) + (Y) → Y & A	↓	↓	24.0	↓	↓	16.0
25	RePLace ● A - Y	(A) - (Y) → Y & A	↓	↓	↓	↓	↓	↓
26	ADD ● Q	*(Q) + Y → Q; (A) <sub>i</sub> = (A) <sub>f</sub>	12.8	11.2	16.0	9.6	8.0	11.2
27	SUBtract ● Q	*(Q) - Y → Q; (A) <sub>i</sub> = (A) <sub>f</sub>	↓	↓	↓	↓	↓	↓
30	ENTer ● Y + Q	Y + (Q) → A	11.2	9.6	↓	8.0	6.4	9.6
31	ENTer ● Y - Q	Y - (Q) → A	↓	↓	↓	↓	↓	↓
32	SToRe ● A + Q	(A) + (Q) → Y & A	12.8	↓	↓	6.4	↓	↓
33	SToRe ● A - Q	(A) - (Q) → Y & A	↓	↓	↓	↓	↓	↓
34	RePLace ● Y + Q	(Y) + (Q) → Y & A	↓	↓	24.0	↓	↓	16.0
35	RePLace ● Y - Q	(Y) - (Q) → Y & A	↓	↓	↓	↓	↓	↓
36	RePLace ● Y + 1	(Y) + 1 → Y & A	↓	↓	↓	↓	↓	↓
37	RePLace ● Y - 1	(Y) - 1 → Y & A	↓	↓	↓	↓	↓	↓

\*For special interpretation of j-designator, see table A-2.

\*\*The times given apply to the no skip (or no jump) or unconditional skip (or jump) options of the various instructions. For any of the conditional options, add 8.0 microseconds.

## LEGEND

**R** a register, A, Q, or B<sub>i</sub>  
**Y** a storage address Y, y+(B<sup>b</sup>)  
**B<sup>n</sup>** B-register n, n = b or j  
**C<sub>j</sub>** input (or output) channel j  
**(R)** contents of register R  
**(R)'** complement of the contents of R  
**(Y)** contents of address Y  
**Y** the operand, Y or (Y)

**(R)<sub>i</sub>** initial contents of R  
**(R)<sub>f</sub>** final contents of R  
**Y<sub>U</sub>** highest order 15 bits of Y  
**(R)<sub>L</sub>** lowest order 15 bits of the contents of R  
**Y<sub>n</sub>** nth bit of Y  
**(R)<sub>n</sub>** nth bit of the contents of R  
**Y<sub>m</sub>** value of Y for mth iteration of a repeated instruction  
**L[Y(R)]** logical (bit-by-bit) product of Y and the contents of R

Table A-1. (Continued)

FUNCTION CODE, f	MACHINE INSTRUCTION  (Boldface type indicates appropriate CS-1 mono-code)	BRIEF DESCRIPTION OF OPERATION PERFORMED BY THE INSTRUCTION	INSTRUCTION EXECUTION TIME,** Microseconds					
			Single Instruction (or first execution repeat mode)			Repeat Mode		
			k-designator			k-designator		
			0, 4	7	Other	0, 4	7	Other
40	<b>ENTer</b> • Logical Product	*L[Y(Q)] → A	11.2	9.6	16.0	8.0	6.4	9.6
41	<b>ADD</b> • Logical Product	L[Y(Q)] + (A) → A	↓	↓	↓	↓	↓	↓
42	<b>SUBtract</b> • Logical Product	(A) - L[Y(Q)] → A	↓	↓	↓	↓	↓	↓
43	<b>COMParE</b> • <b>MASKed</b>	(A) - L[Y(Q)]; sense j; (A) + L[Y(Q)]; (A) <sub>i</sub> = (A) <sub>f</sub>	↓	↓	↓	↓	↓	↓
44	<b>RePLace</b> • Logical Product	*L(Y)(Q) → Y & A	↓	↓	24.0	↓	↓	16.0
45	<b>RePLace</b> • A + LP	L(Y)(Q) + (A) → Y & A	↓	↓	↓	↓	↓	↓
46	<b>RePLace</b> • A - LP	(A) - L(Y)(Q) → Y & A	↓	↓	↓	↓	↓	↓
47	<b>SToRe</b> • Logical Product	L(A)(Q) → Y; (A) <sub>i</sub> = (A) <sub>f</sub>	12.8	↓	16.0	6.4	↓	9.6
50	<b>SElective</b> • <b>SET</b>	Set (A) <sub>n</sub> for Y <sub>n</sub> = 1	12.8	11.2	16.0	9.6	8.0	11.2
51	<b>SElective</b> • <b>ComPlement</b>	Complement (A) <sub>n</sub> for Y <sub>n</sub> = 1	↓	↓	↓	↓	↓	↓
52	<b>SElective</b> • <b>CLear</b>	Clear (A) <sub>n</sub> for Y <sub>n</sub> = 1	↓	↓	↓	↓	↓	↓
53	<b>SElective</b> • <b>SUBstitute</b>	Y <sub>n</sub> → (A) <sub>n</sub> for (Q) <sub>n</sub> = 1	↓	↓	↓	↓	↓	↓
54	<b>RePLace SElective</b> • <b>SET</b>	Set (A) <sub>n</sub> for (Y) <sub>n</sub> = 1; → Y & A	↓	↓	24.0	↓	↓	16.0
55	<b>RePLace SElective</b> • <b>ComPlement</b>	Complement (A) <sub>n</sub> for (Y) <sub>n</sub> = 1; → Y & A	↓	↓	↓	↓	↓	↓
56	<b>RePLace SElective</b> • <b>CLear</b>	Clear (A) <sub>n</sub> for (Y) <sub>n</sub> = 1; → Y & A	↓	↓	↓	↓	↓	↓
57	<b>RePLace SElective</b> • <b>SUBstitute</b>	(Y) <sub>n</sub> → (A) <sub>n</sub> for (Q) <sub>n</sub> = 1; → Y & A	↓	↓	↓	↓	↓	↓
60	<b>JumP</b> (arithmetic)	*) Jump to Y per special interpretation of j	8.0	9.6	16.0	Repeat Mode (does not apply for 60 ≤ f ≤ 76)		
61	<b>JumP</b> (manual)		↓	↓	↓			
62	<b>JumP</b> on • C <sup>j</sup> <b>ACTIVE INPut</b> Buffer		} Jump to Y if channel j has active input/output buffer	↓	↓			
63	<b>JumP</b> on • C <sup>j</sup> <b>ACTIVE OUTPut</b> Buffer	↓		↓	↓			
64	<b>Return JumP</b> (arithmetic)	*) Store address of next (or current) instruction in Y <sub>L</sub> and jump to Y + 1 per special interpretation of j	12.8/	11.2/	16.0/			
65	<b>Return JumP</b> (manual)		19.2	17.6	24.0			
66	<b>TERMinate</b> • C <sup>j</sup> • <b>INPut</b> Buffer	} Terminate input/output buffer on channel j	8.0	↓	16.0			
67	<b>TERMinate</b> • C <sup>j</sup> • <b>OUTPut</b> Buffer		↓	↓	↓			
70	<b>RePeaT</b>	Execute next instruction Y times	↓	9.6	↓			
71	<b>B SKip</b> on • B <sup>j</sup>	(B <sup>j</sup> ) = Y, skip next instruction and clear B <sup>j</sup> (B <sup>j</sup> ) ≠ Y, advance B <sup>j</sup> and execute next instruction	9.6	11.2	↓			
72	<b>B JumP</b> on • B <sup>j</sup>	(B <sup>j</sup> ) = 0, execute next instruction (B <sup>j</sup> ) ≠ 0, decrease B <sup>j</sup> and jump to Y	8.0	9.6	↓			
73	<b>INPut</b> Buffer on • C <sup>j</sup>	} Initiate input (or output) buffer with (or without) monitor on channel j	↓	↓	↓			
74	<b>OUTPut</b> Buffer on • C <sup>j</sup>		↓	↓	↓			
75	<b>INPut</b> Buffer on • C <sup>j</sup> with • <b>MONITOR</b>		↓	↓	↓			
76	<b>OUTPut</b> Buffer on • C <sup>j</sup> with • <b>MONITOR</b>		16.0	↓	24.0			
77	<b>Illegal Code</b>	Fault Interrupt	↓	↓	↓			

### ADDITIONAL CS-1 MONO-CODES

**CLear** • A, • Q, • B<sup>j</sup>, or • Y  
**TEST** • C<sup>0</sup> or C<sup>1</sup>

**NO** **OPERation**  
**ComPlement** • A or • Q

**Remove Interrupt Lockout**  
**Remove Interrupt Lockout and JumP** • Y

Table A-2. Interpretation of UNIVAC 1206 Instruction-Word Designators

( **Bold Face Type** indicates the appropriate CS-1 code)

VALUE OF DESIGNATOR		0	1	2	3	4	5	6	7			
NORMAL INTERPRETATION OF j-DESIGNATOR		Skip the next instruction if the condition specified by j is met										
SPECIAL INTERPRETATION OF j DESIGNATOR FOR FUNCTION CODE(S) See Table A-1	40 44	Execute the next instruction  (fj = 600,601 also clears interrupt and bootstrap modes)	SKIP the next instruction unconditionally	Q is POSITIVE	Q is NEGATIVE	A is ZERO	A is NOT zero	A is POSITIVE	A is NEGATIVE			
	23			There is an EVEN no. of ones in A	There is an ODD no. of ones in A							
	04			There was NO OverFlow	There was OverFlow	Y is WITHIN the Range (Q) > (A)	Y is OUTside the Range (Q) > (A)	Y is LESS than or equal to (A)	Y is MORE than (A)			
	26 27			A is POSITIVE	A is NEGATIVE	Q is ZERO	Q is NOT zero	Q is POSITIVE	Q is NEGATIVE			
	60 64			Jump to Y unconditionally	Jump to the address in Y if the condition specified by j is met							
	61 65			Jump to the address in Y			Jump to the address in Y and					
				unconditionally	if KEY 1 is set	if KEY 2 is set	if KEY 3 is set	STOP	stop if STOP 5 is set	stop if STOP 6 is set	stop if STOP 7 is set	
	INTERPRETATION OF r-DESIGNATOR FOR FUNCTION CODE 70			The operand address, $Y_m+1$ , for the next iteration of the repeated instruction is								
		$Y_m$ no code	$Y_m+1$ ADV	$Y_m-1$ BACK	$Y_m+B^b$ ADD B	$Y_m [+B^e]$ R	$Y_m+1[+B^e]$ ADV R	$Y_m-1 [+B^e]$ BACK R	$Y_m+B^b[+B^e]$ ADD B R			
		For a repeated instruction of the <i>replace</i> type, $B^e$ increments the Y-address for the <i>store</i> portion as indicated.										
INTERPRETATION OF k- (or k-) DESIGNATOR	Read Type	$Y_U =$ all zeros			Y = Y	all bits of $Y_U$ same as $Y_{14}$		all bits of $Y_U$ same as $Y_{29}$	Y = (A)			
		$Y_L = Y$ no code	$Y_L = (Y)_L$ L	$Y_L = (Y)_U$ U	W	$Y_L = Y$ X	$Y_L = (Y)_L$ LX	$Y_L = (Y)_U$ UX	A			
	Store Type <sup>Ⓜ</sup>	Store (R) in Q Q	Store (R) <sub>L</sub> in $Y_L$ L	Store (R) <sub>L</sub> in $Y_U$ U	Store (R) in Y W	Store (R) in A A	Store (R) <sub>L</sub> in $Y_L$ CPL	Store (R) <sub>L</sub> in $Y_U$ CPU	Store (R) <sub>L</sub> in Y CPW			
Replace Type		$Y_U =$ all zeros			Y = Y	all bits of $Y_U$ same as						
		$Y_L = (Y)_L$ Store (R) <sub>L</sub> in $Y_L$ L	$Y_L = (Y)_U$ Store (R) <sub>L</sub> in $Y_U$ U	Store (R) in Y W		$Y_{14}$ $Y_L = (Y)_L$ Store (R) <sub>L</sub> in $Y_L$ LX	$Y_{29}$ $Y_L = (Y)_U$ Store (R) <sub>L</sub> in $Y_U$ UX					

<sup>Ⓜ</sup> Storing a 15-bit quantity in the upper (or lower) half of Y does not disturb the other half. Storing the contents of a register in the register itself, complements the register.

For those instructions in which the *j* designator has no special interpretation, it specifies the condition under which the next sequential instruction in the program will be skipped. This provides for branching from a sequence without executing a Jump instruction, as would normally occur if a Skip condition were not satisfied.

Skip of the next sequential instruction is determined by the following rules in all instructions except 04, 12, 13, 16, 17, 26, 27, 60 through 67, and 70 through 76.

- j* = 0: Execute the next instruction.
- j* = 1: Skip the next instruction.
- j* = 2: Skip the next instruction if (Q) is positive.
- j* = 3: Skip the next instruction if (Q) is negative.
- j* = 4: Skip the next instruction if (A) is zero, (i.e., *positive zero*)
- j* = 5: Skip the next instruction if (A) is non-zero.
- j* = 6: Skip the next instruction if (A) is positive.
- j* = 7: Skip the next instruction if (A) is negative.

When the branch (Skip or Jump) condition involves the sign of the quantity in A or Q, the evaluation examines the sign bit of these quantities; hence, a positive zero (all *zeros*) is considered a positive quantity, and a negative zero (all *ones*) is considered a negative quantity.

### INPUT-OUTPUT CHANNEL DESIGNATOR— $\hat{j}$

The  $\hat{j}$  designator (4 bits) appears in bit-positions 23, 22, 21, and 20 of the U-register, or an input/output instruction, specifying the C-channel for the instruction. Bit 23 assumes a value of eight, bit 22 a value of four, bit 21 a value of two, and bit 20 a value of one; thus the  $\hat{j}$  designator provides accessibility to the 14 input-output channels number 0-15<sub>8</sub>.

Instructions 13, 17, 62, 63, 66, 67, 73, 74, 75, and 76 use the  $\hat{j}$  designator configuration.

### OPERAND INTERPRETATION DESIGNATOR—*k* or $\hat{k}$

The *k* designator appears in bit-positions 20, 19, and 18 of the U-register, or an instruction; a  $\hat{k}$  designator appears only in bit-positions 19 and 18, since bit 20 is a portion of the  $\hat{j}$  designator. Instructions 13, 17, 62, 63, and 73 through 76 use the  $\hat{k}$  designator configuration since they perform input-output activities and require a  $\hat{j}$  designator for channel specification.

The *k* and  $\hat{k}$  designators control operand interpretation. Those instructions which *read* an operand but do not replace it after the arithmetic is performed are designated *Read* instructions. Those instructions which do not *read* an operand but *store* one are designated *Store* instructions. Instructions which both *read* and *store* operands are classified as *Replace* instructions.

The various values of *k* or  $\hat{k}$  affect the operand in the following list except where otherwise noted under individual instruction descriptions.

- 1) *Read* instructions (01 through 13, 20 through 23, 26, 27, 30, 31, 40 through 43, 50 through 53, and 60 through 76):

- k* or  $\hat{k}$  = 0:  $Y_U$  = all zeros;  $Y_L$  = Y.
- k* or  $\hat{k}$  = 1:  $Y_U$  = all zeros;  $Y_L$  = (Y)<sub>L</sub>.
- k* or  $\hat{k}$  = 2:  $Y_U$  = all zeros;  $Y_L$  = (Y)<sub>U</sub>.
- k* or  $\hat{k}$  = 3: Y = Y.
- k* = 4:  $Y_U$  = same bits as  $Y_{14}$ ;  $Y_L$  = Y.
- k* = 5:  $Y_U$  = same bits as  $Y_{14}$ ;  $Y_L$  = (Y)<sub>L</sub>.
- k* = 6:  $Y_U$  = same bits as  $Y_{29}$ ;  $Y_L$  = (Y)<sub>U</sub>.
- k* = 7: Y = (A).

For instructions 23, 52, and 53, *k* = 7 is not used.

For instruction 13, only  $\hat{k}$  = 3 is permitted.

For instructions 73 through 76,  $\hat{k}$  = 2 is not used.

- 2) *Store* instructions (14 through 16, 17, 32, 33, and 47):
- k** = 0: Store (A or B<sup>j</sup>) in Q\*.
  - k** = 1: Store (A<sub>L</sub>, Q<sub>L</sub>, or B<sup>j</sup>) in Y<sub>L</sub>, leaving (Y)<sub>U</sub> undisturbed.
  - k** = 2: Store (A<sub>L</sub>, Q<sub>L</sub>, or B<sup>j</sup>) in Y<sub>U</sub>, leaving (Y)<sub>L</sub> undisturbed.
  - k** or **Ĵ** = 3: Store (A, Q, C<sup>j</sup>, or B<sup>j</sup>) in Y.
  - k** = 4: Store (Q or B<sup>j</sup>) in A\*\*.
  - k** = 5: Store complement of (A<sub>L</sub>, Q<sub>L</sub>, or B<sup>j</sup>) in Y<sub>L</sub>, leaving (Y)<sub>U</sub> undisturbed.
  - k** = 6: Store complement of (A<sub>L</sub>, Q<sub>L</sub>, or B<sup>j</sup>) in Y<sub>U</sub>, leaving (Y)<sub>L</sub> undisturbed.
  - k** = 7: Store complement of (A, Q, or B<sup>j</sup>) in Y. (Storing the complement of B<sup>j</sup> is the same complement as for a 30-bit register.)
- For instruction 17, only **Ĵ** = 3 is permitted.

- 3) *Replace* instructions (24, 25, 34 through 37, 44 through 46, and 54 through 57):
- k** = 0: Not used.
  - k** = 1: *Read* portion—Y<sub>U</sub> = all zeros; Y<sub>L</sub> = (Y)<sub>L</sub>.  
*Store* portion—stores (A<sub>L</sub>, Q<sub>L</sub>, or B<sup>j</sup>) in Y<sub>L</sub>, leaving (Y)<sub>U</sub> undisturbed.
  - k** = 2: *Read* portion—Y<sub>U</sub> = all zeros; Y<sub>L</sub> = (Y)<sub>U</sub>.  
*Store* portion—stores (A<sub>L</sub>, Q<sub>L</sub>, or B<sup>j</sup>) in Y<sub>U</sub>, leaving (Y)<sub>U</sub> undisturbed.
  - k** = 3: *Read* portion—Y = Y.  
*Store* portion—stores (A, Q, or B<sup>j</sup>) in Y.
  - k** = 4: Not used.
  - k** = 5: *Read* portion—Y<sub>U</sub> = same bits as Y<sub>14</sub>; Y<sub>L</sub> = (Y)<sub>L</sub>.  
*Store* portion—stores (A<sub>L</sub>, Q<sub>L</sub>, or B<sup>j</sup>) in Y<sub>L</sub>, leaving (Y)<sub>U</sub> undisturbed.
  - k** = 6: *Read* portion—Y<sub>U</sub> = same bits as Y<sub>29</sub>; Y<sub>L</sub> = Y<sub>U</sub>.  
*Store* portion—stores (A<sub>L</sub>, Q<sub>L</sub>, or B<sup>j</sup>) in Y<sub>U</sub>, leaving (Y)<sub>L</sub> undisturbed.
  - k** = 7: Not used.

The *Repeat* instruction requires special interpretation when followed by a *Replace* instruction. See details on page A-22, Instruction No. 70, *REPEAT*.

### INDEX DESIGNATOR—**b**

The **b** designator (3 bits) appears in bit-positions 17, 16, and 15 of the U-register, or an instruction specifying which of the B-registers, if any, will be used to modify the Operand Designator, **y**, to form  $Y = y + (B^b)$ . This operation employs an additive accumulator; hence, a quantity consisting of all *zero* cannot result unless the bits of both the Operand Designator, **y**, and (B<sup>b</sup>) are all *zeros*.

Effect of the various values of **b**, the Index Designator, is summarized:

- b** = 0: Do not modify **y**.
- b** = 1: Add (B<sup>1</sup>) to **y** (modulo 2<sup>15</sup>−1).
- b** = 2: Add (B<sup>2</sup>) to **y** (modulo 2<sup>15</sup>−1).
- b** = 3: Add (B<sup>3</sup>) to **y** (modulo 2<sup>15</sup>−1).
- b** = 4: Add (B<sup>4</sup>) to **y** (modulo 2<sup>15</sup>−1).
- b** = 5: Add (B<sup>5</sup>) to **y** (modulo 2<sup>15</sup>−1).
- b** = 6: Add (B<sup>6</sup>) to **y** (modulo 2<sup>15</sup>−1).
- b** = 7: Add (B<sup>7</sup>) to **y** (modulo 2<sup>15</sup>−1).

### OPERAND DESIGNATOR—**y**

The **y** designator (15 bits) appears in bit-positions 14 through 0 of an instruction. The operand or address of the operand, **Y**, is relative to **y** since  $Y = y + (B^b)$ .

\*A 14000 00000 instruction complements (Q).

\*\*A 15040 00000 instruction complements (A).

## MAGNETIC CORE MEMORY ASSIGNMENT

The main Magnetic Core memory consists of 32,768 addressable storage locations. Seventy-two of these locations are special-purpose and provide seven distinct functions:

- 1) The Fault Entrance Register
- 2) The Real-Time Clock Register
- 3) External Interrupt Entrance Register for each channel
- 4) Internal Interrupt Entrance Register for each *input* channel
- 5) Internal Interrupt Entrance Register for each *output* channel
- 6) Input Buffer Control Register for each *input* channel
- 7) Output Buffer Control Register for each *output* channel.

Each of the other memory locations are used for:

- 1) Instructions
- 2) Data storage

Table A-3 is a list of core memory address assignments.

Table A-3. UNIVAC 1206 – Core Memory Address Assignments

ADDRESS (octal)	STORAGE FUNCTION
0 0 0 0 0	Fault Entrance Register.
0 0 0 0 1–1 7	No Fixed Assignment – Holds contents of wired memory during program recovery.
0 0 0 2 0–3 5	External Interrupt Entrance Registers for Channels 0 thru 15 <sub>s</sub> .
0 0 0 3 6	Real-time Clock Register.
0 0 0 3 7	No Fixed Assignment.
0 0 0 4 0–5 5	Internal Interrupt Entrance Registers for Input Channels 0 thru 15 <sub>s</sub> .
0 0 0 5 6–5 7	No Fixed Assignment.
0 0 0 6 0–7 5	Internal Interrupt Entrance Registers for Output Channels 0 thru 15 <sub>s</sub> .
0 0 0 7 6–7	No Fixed Assignment.
0 0 1 0 0–1 5	Input Buffer Control Registers for Input Channels 0 thru 15 <sub>s</sub> .
0 0 1 1 6–1	No Fixed Assignment.
0 0 1 2 0–3 5	Output Buffer Control Registers for Output Channels 0 thru 15 <sub>s</sub> .
0 0 1 3 6 up	32,673 Word Locations—No Fixed Assignment.

## WIRED MEMORY

The UNIVAC 1206 Military Computer contains 16 words of wired storage. Programming this memory area requires a process of wiring-in the desired instructions. The nature of these storage locations prevents accidental destruction of program instructions contained therein since entries cannot be made via main memory.

An Input Bootstrap routine occupies this memory, and its execution is controlled by the Automatic Recovery Switch.

## AUTOMATIC RECOVERY

In the event of a *fault* condition (encountering either a 00 or 77 function code), the Automatic Recovery Switch directs computer activity. This switch has three positions: *Up*, *Center*, and *Momentary Down*.

- In the *Up* position a fault causes an Interrupt to address 00 of wired memory which results in automatic execution of the wired Bootstrap routine.

- In the *Center* position a fault causes an Interrupt to address 00000 of main memory. Action continues as programmed at address 00000.
- The *Momentary Down* position is for manual initiation of the wired Bootstrap routine after a MASTER CLEAR.

## BUFFER MODES

The UNIVAC 1206 Military Computer provides two buffer modes: 1) with monitor and 2) without monitor.

Buffer with monitor transfers words sequentially, starting at the given initial address through the given terminal address, on the specified input or output channel. The computer continues execution of program instructions concurrent with the buffer process. Completion of the buffer process causes an Internal Interrupt to the Internal Interrupt Entrance Register assigned to the input or output channel. This register usually contains a RETURN-JUMP instruction\*. (See Instructions 75 and 76.)

Buffer without the monitor is identical to buffer with monitor except that the interrupt occurs upon completion. (See Instructions 73 and 74.)

## LIST OF INSTRUCTIONS

This section lists the repertoire of instructions used with the UNIVAC 1206 Military Real Time Computer. Common usage of these instructions is also included; no attempt is made to indicate more sophisticated use.

### 01 RIGHT SHIFT Q

This instruction shifts (Q) to the right Y bit positions. The higher-order bits are replaced with the original sign bit as the word is shifted. Only the lower-order six bits of Y are recognized for this instruction. The higher-order 24 bits are ignored.

Example of right shift in Q: Y = 2

Content of Q	Content of Q
(Q) <sub>i</sub> (positive) = 0 1 0 1	(Q) <sub>i</sub> (negative) = 1 0 1 0
First shift      0 0 1 0	First shift      1 1 0 1
Second shift     0 0 0 1	Second shift     1 1 1 0

### 02 RIGHT SHIFT A

This instruction shifts (A) to the right Y bit positions. The higher-order bits are replaced with the original sign bit as the word is shifted. Only the lower-order six bits of Y are recognized for this instruction. The higher-order 24 bits are ignored. The overall operation is analogous to the example given in the foregoing instruction.

### 03 RIGHT SHIFT AQ

This instruction shifts (A) and (Q) as one 60-bit register. The shift is to the right Y bit positions with the lower-order bits of A shifting into the higher-order bit positions of Q. The higher-order bits of A are replaced with the original sign bit as the word is shifted. Only the lower-order six bits of Y are recognized for this instruction. The higher-order 24 bits are ignored.

\*Suggested instruction for the Internal Interrupt Register is:

650nn nnnnn—Exit to an Interrupt subroutine for remedial action. This subroutine ends with a 601nn instruction which clears the Interrupt mode, then returns control to the main routine.

Example of right shift in AQ:  $Y = 2$

Content of AQ		Content of AQ	
$(AQ)_i$ (positive) =	0 1 0 1 0 0 1 1	$(AQ)_i$ (negative) =	1 0 0 0 1 0 1 0
First shift	0 0 1 0 1 0 0 1	First shift	1 1 0 0 0 1 0 1
Second shift	0 0 0 1 0 1 0 0	Second shift	1 1 1 0 0 0 1 0

#### 04 COMPARE

This instruction compares the signed value of  $Y$  with the signed value of  $(A)$  and/or  $(Q)$ . It does not alter either  $(A)$  or  $(Q)$ . The Branch Condition Designator,  $j$ , is interpreted in a special way for this instruction as listed below:

- $j = 0$ : Execute the next instruction.
- $j = 1$ : Skip the next instruction.
- $j = 2$ : Skip the next instruction if  $Y$  is less than, or equal to,  $(Q)$ .
- $j = 3$ : Skip the next instruction if  $Y$  is greater than  $(Q)$ .
- $j = 4$ : Skip the next instruction if  $(Q)$  is greater than, or equal to  $Y$ , and  $Y$  is greater than  $(A)$ .
- $j = 5$ : Skip the next instruction if  $Y$  is greater than  $(Q)$  or if  $Y$  is less than, or equal to,  $(A)$ .
- $j = 6$ : Skip the next instruction if  $Y$  is less than, or equal to,  $(A)$ .
- $j = 7$ : Skip the next instruction if  $Y$  is greater than  $(A)$ .

#### 05 LEFT SHIFT Q

This instruction shifts  $(Q)$  circularly to the left  $Y$  bit positions\*. The lower-order bits are replaced with the higher-order bits as the word is shifted. Only the lower-order six bits of  $Y$  are recognized for this instruction. The higher-order 24 bits are ignored.

Example of left circular shift in  $Q$ :  $(Y) = 2$

Content of Q		Content of Q	
$(Q)_i$ (positive) =	0 0 1 1	$(Q)_i$ (negative) =	1 1 0 0
First shift	0 1 1 0	First shift	1 0 0 1
Second shift	1 1 0 0	Second shift	0 0 1 1

#### 06 LEFT SHIFT A

This instruction shifts  $(A)$  circularly to the left  $Y$  bit positions.\* The lower-order bits are replaced with the higher-order bits as the word is shifted. Only the lower-order six bits of  $Y$  are recognized for this instruction. The higher-order 24 bits are ignored. The over-all operation is analogous to the example given in the foregoing instruction.

#### 07 LEFT SHIFT AQ

This instruction shifts  $(A)$  and  $(Q)$  as one 60-bit register. The shift is circular to the left  $Y$  bit positions.\* The lower-order bits of  $A$  are replaced with the higher-order bits of  $Q$  and the lower-order bits of  $Q$  are replaced with the higher-order bits of  $A$ . Only the lower-order six bits of  $Y$  are recognized by this instruction. The higher-order 24 bits are ignored.

\*Maximum shift count permitted is 59 places.



Example of left circular shift in AQ:  $Y = 2$

Content of AQ	Content of AQ
(AQ) <sub>i</sub> (positive) = 0 1 0 1 0 0 1 1	(AQ) <sub>i</sub> (negative) = 1 0 0 0 1 0 1 1
First shift            1 0 1 0 0 1 1 0	First shift            0 0 0 1 0 1 1 1
Second shift          0 1 0 0 1 1 0 1	Second shift          0 0 1 0 1 1 1 0

10 *ENTER Q*

Clear the Q-register. Then transmit  $Y$  to Q.

11 *ENTER A*

Clear A. Then transmit  $Y$  to A.

12 *ENTER B<sup>j</sup>*

Clear B-register  $j$ . Then transmit the lower-order 15 bits of  $Y$  to B-register  $j$ . The higher-order 15 bits of  $Y$  are ignored in this instruction. The Branch Condition Designator,  $j$ , is used to specify the selected B-register for this instruction and is not available for its normal function.

13 *EXTERNAL FUNCTION ON C<sup>j</sup>*

$\hat{j} = 0$  or  $1$ . Interrogate the two bits connected to the input-active designator (flip-flops) on an interconnected computer. If the interconnected computer's input buffer is active, skip the next instruction. If the interconnected computer's input buffer is not active, execute the next instruction. There are no External Function lines on  $C^0$  or  $C^1$ .  $\hat{k} = 3$  is required for timing. When  $\hat{j} \neq 0$  or  $1$ , transmit  $Y$ , the External Function, over the channel specified by  $\hat{j}$ . Only  $\hat{k} = 3$  is permitted.

14 *STORE Q*

Store (Q) at storage address  $Y$  as directed by the Operand Interpretation Designator,  $k$ . If  $k = 0$  complement (Q). If  $k = 4$ , store in A.

15 *STORE A*

Store (A) at storage address  $Y$  as directed by the Operand Interpretation Designator,  $k$ . If  $k = 4$ , complement (A). If  $k = 0$ , store in Q.

16 *STORE B<sup>j</sup>*

Store a 30-bit quantity whose lower-order 15 bits correspond to the content of B-register  $j$  and whose higher-order 15 bits are *zero* at storage address  $Y$  as directed by the Operand Interpretation Designator,  $k$ . The Branch Condition Designator,  $j$ , is used to specify the selected B-register for this instruction and is not available for its normal function.

17 *STORE C<sup>j</sup>*

Store the content of the C-channel specified by  $\hat{j}$  at storage address  $Y$ . An Input Acknowledge signal is then sent on the C-channel. Only  $\hat{k} = 3$  is permitted.

20 *ADD A*

Add  $Y$  to the previous content of the Accumulator.

\*Instruction 17 is intended for use in the computer's reply to an interrupt. It is not synchronized with the input buffering process and cannot be used in the repeat mode. Successive iterations of instruction 17 must be programmed with a suitable time delay between iterations, e.g., a 12000 00000 instruction.

## 21 *SUBTRACT A*

Subtract **Y** from the previous content of the Accumulator.

## 22 *MULTIPLY*

Multiply (**Q**) times **Y** leaving the double-length product in **AQ**. If the factors are considered as integers, the product is an integer in **AQ**.

The Branch Condition Designator, **j**, is interpreted prior to sign correction\* permitting sensing of a product with  $(A)_t = 0$ . If **j** equal 4, a skip of the next instruction is made when  $(A)_t = 0$ . When  $(A)_t \neq +0$ , a double-length product has been formed with significant bit(s) in the Accumulator; however, if a Skip does occur for **j** = 4, the Multiply instruction can be re-executed with the same operand and with **j** = 2 or 3 to determine if  $Q_{29}$  contains a significant bit (a *one*) of the product.

In this instruction, **k** = 7 should not be used.

## 23 *DIVIDE*

Divide (**AQ**) by **Y** leaving the quotient in the Q-register and the remainder in the A register. The remainder bears the same sign as the quotient. In this instruction, **k** = 7 should not be used.

NOTE:

*An overflow indicates that the answer is not correct. Overflow occurs upon division by positive or negative zero or when the quotient exceeds the Q-register (29 bits plus sign).*

*In instruction 23 the Branch Condition Designator, **j**, has the following meaning:*

- j** = 0: Execute the next instruction.
- j** = 1: Skip the next instruction unconditionally.
- j** = 2: Skip the next instruction if overflow has not occurred.
- j** = 3: Skip the next instruction if overflow has occurred.
- j** = 4: Skip the next instruction if (A) is zero (no remainder).
- j** = 5: Skip the next instruction if (A) is non-zero (remainder exists).
- j** = 6 or 7 should not be used.

## 24 *REPLACE A + Y*

Add (A) to the previous content of A. Store (A) at storage address Y.

## 25 *REPLACE A - Y*

Subtract (Y) from the previous content of A. Then store (A) at storage address Y.

## 26 *ADD Q*

Interchange (A) and (Q). Then add **Y** to (A). Interchange (A) and (Q). The content of A is undisturbed by this instruction. The Branch Condition Designator, **j**, has special meaning in this instruction as in instruction 27.

## 27 *SUBTRACT Q*

Interchange (A) and (Q). Then subtract **Y** from (A). Interchange (A) and (Q). The content of A is undisturbed by this instruction. The Branch Condition Designator, **j**, has special meaning in this instruction as listed below.

\*The multiplication operation itself is always carried out with *positive* numbers; negative factors are automatically complemented and appropriate corrections are applied to the product.

*In instructions 26 and 27 the Branch Condition Designator, j, has the following meaning:*

- j* = 0: Execute the next instruction.
- j* = 1: Skip the next instruction unconditionally.
- j* = 2: Skip the next instruction if (A) is positive.
- j* = 3: Skip the next instruction if (A) is negative.
- j* = 4: Skip the next instruction if (Q) is zero.
- j* = 5: Skip the next instruction if (Q) is non-zero.
- j* = 6: Skip the next instruction if (Q) is positive.
- j* = 7: Skip the next instruction if (Q) is negative.

30 *ENTER Y+Q*

Clear A. Then transmit (Q) to A. Then add Y to (A).

31 *ENTER Y-Q*

Clear A. Then transmit (Q) to A. Then subtract Y from (A). Finally, complement (A).

32 *STORE A+Q*

Add (Q) to the previous content of A. Then store (A) at storage address Y as directed by the Operand Interpretation Designator, k.

33 *STORE A-Q*

Subtract (Q) from the previous content of A. Then store (A) at storage address Y as directed by the Operand Interpretation Designator, k.

34 *REPLACE Y+Q*

Clear A. Then transmit (Q) to A. Then add (Y) to (A). Then store (A) at storage address Y.

35 *REPLACE Y-Q*

Clear A. Then transmit (Q) to A. Then subtract (Y) from (A). Then complement (A) and store at storage address Y.

36 *REPLACE Y+1*

Clear A. Then set (A) = 1. Then add (Y) to (A). Then store (A) at storage address Y.

37 *REPLACE Y-1*

Clear A. Then set (A) = 1. Then subtract (Y) from (A). Then complement (A) and store at storage address Y.

40 *ENTER LOGICAL PRODUCT*

Enter in A the bit-by-bit product of Y and (Q).

The *j* designator is interpreted in a special way for this instruction for the value *j* = 2 or 3. If *j* = 2, Skip if the parity of (A)<sub>t</sub> is even. If *j* = 3, Skip if the parity of (A)<sub>t</sub> is odd.

NOTE:

*Even parity means an even number of ONES in the A-register.*

*Odd parity means an odd number of ONES in the A-register.*

41 *ADD LOGICAL PRODUCT*

Add to (A) the bit-by-bit product of Y and (Q).

42 *SUBTRACT LOGICAL PRODUCT*

Subtract from (A) the bit-by-bit product of Y and (Q).

43 *COMPARE MASKED*

Subtract from (A) the bit-by-bit product of Y and (Q), and perform the branch point evaluation for Skip of next sequential instruction as directed by the Branch Condition Designator, j.

This instruction results in no net change in the content of any operational register. It provides, through the Branch Condition Designator, j, a comparison of a portion of Y with (A).

44 *REPLACE LOGICAL PRODUCT*

Enter in A the bit-by-bit product of (Y) and (Q). Then store (A) at storage address Y.

The j designator is interpreted in a special way for this instruction for the values j = 2 or 3. If j = 2, Skip if the parity of (A)<sub>t</sub> is even. If j = 3, Skip if the parity of (A)<sub>t</sub> is odd.

NOTE:

*Even parity means an even number of ONES in the A-register.*

*Odd parity means an odd number of ONES in the A-register.*

45 *REPLACE A+LOGICAL PRODUCT*

Add to (A) the bit-by-bit product of (Y) and (Q). Then store (A) at storage address Y.

46 *REPLACE A-LOGICAL PRODUCT*

Subtract from (A) the bit-by-bit product of (Y) and (Q). Then store (A) at storage address Y.

47 *STORE LOGICAL PRODUCT*

Store in address Y the bit-by-bit product of (A) and (Q) as directed by the Operand Interpretation Designator, k.

50 *SELECTIVE SET*

Set the individual bits of A to *one* corresponding to *ones* in Y leaving the remaining bits of A unaltered.

51 *SELECTIVE COMPLEMENT*

Complement the individual bits of A corresponding to *ones* in Y leaving the remaining bits of A unaltered.

52 *SELECTIVE CLEAR*

Clear the individual bits of A corresponding to *ones* in Y leaving the remaining bits of A unaltered. In this instruction, k = 7 should not be used.

53 *SELECTIVE SUBSTITUTE*

Set the individual bits of A with bits of Y corresponding to *ones* in Q leaving the remaining bits of A unaltered.

In this instruction, k = 7 should not be used. If this instruction is to be repeated, k = 0 or k = 4 should not be used.

#### 54 *REPLACE SELECTIVE SET*

Set the individual bits of A to *one* corresponding to *ones* in (Y) leaving the remaining bits of A unaltered. Then store (A) at storage address Y.

#### 55 *REPLACE SELECTIVE COMPLEMENT*

Complement the individual bits of A corresponding to *ones* in (Y) leaving the remaining bits of A unaltered. Then store (A) at storage address Y.

#### 56 *REPLACE SELECTIVE CLEAR*

Clear individual bits of A corresponding to *ones* in (Y) leaving the remaining bits of A unaltered. Then store (A) at storage address Y.

#### 57 *REPLACE SELECTIVE SUBSTITUTE*

Clear individual bits of A corresponding to *ones* in Q leaving the remaining bits of A unaltered. Then form the bit-by-bit product of (Y) and (Q), and set *ones* of this product in corresponding bits of A leaving the remaining bits of A unaltered. Then store (A) at storage address Y.

#### 60 *JUMP (Arithmetic)*

This instruction clears the Program Address Register, P, and enters a new program address in P for certain conditions of either the A- or Q-register content. The Branch Condition Designator, *j*, is interpreted in a special way for this instruction and thus determines the conditions under which a Jump in program address occurs. If the Jump condition is not satisfied, the next sequential instruction in the current sequence is executed in a normal manner. If the Jump condition is satisfied, as listed below, then Y becomes the address of the next instruction and the beginning of a new program sequence.

- j* = 0: No jump. Set Interrupt Enable to remove interrupt lockout, thus clearing Bootstrap and Interrupt modes. Continue with current program sequence.
- j* = 1: Execute jump. Set Interrupt Enable to remove interrupt lockout, thus clearing Bootstrap and Interrupt modes.
- j* = 2: Execute jump if (Q) is positive.
- j* = 3: Execute jump if (Q) is negative.
- j* = 4: Execute jump if (A) is zero.
- j* = 5: Execute jump if (A) is non-zero.
- j* = 6: Execute jump if (A) is positive.
- j* = 7: Execute jump if (A) is negative.

#### 61 *JUMP (Manual)*

The instruction clears the Program Address Register, P, and enters a new program address in P for certain conditions of manual JUMP key selections. The Branch Condition Designator, *j*, is interpreted in a special way for this instruction and thus determines the conditions under which a jump in program address occurs. If the Jump condition is not satisfied, the next sequential instruction in the current sequence is executed in a normal manner. If the Jump condition is satisfied, as listed below, then Y becomes the address of the next instruction and the beginning of a new program sequence.

Program execution may be stopped by certain STOP selections on execution of this instruction. The Branch Condition Designator, *j*, specifies which key selections are effective.

- $j = 0$ : Execute jump regardless of key selections.
- $j = 1$ : Execute jump if JUMP 1 is selected.
- $j = 2$ : Execute jump if JUMP 2 is selected.
- $j = 3$ : Execute jump if JUMP 3 is selected.
- $j = 4$ : Execute jump. Stop computation.
- $j = 5$ : Execute jump. Stop computation if STOP 5 is selected.
- $j = 6$ : Execute jump. Stop computation if STOP 6 is selected.
- $j = 7$ : Execute jump. Stop computation if STOP 7 is selected.

## 62 *JUMP ON $C^j$ ACTIVE INPUT BUFFER*

This instruction clears the Program Address Register, P, and enters a new program address in P for certain input buffer conditions on the channel designated by  $\hat{j}$ . If the buffer is active, the Jump condition is satisfied; then Y becomes the address of the next instruction. If the buffer is inactive, the Jump condition is not satisfied. The next sequential instruction in the current sequence is executed in the normal manner.  $\hat{k} = 0, 1, 2, \text{ or } 3$  is permitted.

## 63 *JUMP ON $C^j$ ACTIVE OUTPUT BUFFER*

This instruction clears the Program Address Register, P, and enters a new address in P for certain output buffer conditions on the channel designated by  $\hat{j}$ . If the buffer is active, the Jump condition is satisfied; then Y becomes the address of the next instruction. If the buffer is inactive, the Jump condition is not satisfied. The next sequential instruction in the current sequence is executed in the normal manner.  $\hat{k} = 0, 1, 2, \text{ or } 3$  is permitted.

## 64 *RETURN JUMP (Arithmetic)*

This instruction executes a Return-Jump sequence for certain conditions of either the A- or Q-register content. The Branch Condition Designator, j, is interpreted in a special way for this instruction and determines the conditions under which the Return-Jump sequence is executed. If the Return-Jump condition is not satisfied, the next sequential instruction in the current sequence is executed in a normal manner. If the Return-Jump condition is satisfied, as listed below, the following sequence is performed.

Store  $(P) + p^*$  in the lower half of memory address Y. Then jump to  $Y + 1$ .

- $j = 0$ : No action; continue with the current program sequence.
- $j = 1$ : Execute return jump.
- $j = 2$ : Execute return jump if (Q) is positive.
- $j = 3$ : Execute return jump if (Q) is negative.
- $j = 4$ : Execute return jump if (A) is zero.
- $j = 5$ : Execute return jump if (A) is non-zero.
- $j = 6$ : Execute return jump if (A) is positive.
- $j = 7$ : Execute return jump if (A) is negative.

## 65 *RETURN JUMP (Manual)*

This instruction executes a Return Jump sequence for certain conditions of manual key selections. The Branch Condition Designator, j, is interpreted in a special way for this instruction and determines the conditions under which the Return Jump sequence is executed. If the Return Jump condition is not satisfied, the next sequential instruction in the current sequence is executed in a normal manner. If the Return Jump condition is satisfied, as listed below, the following sequence is performed.

Store  $(P) + p^*$  in the lower half of memory address  $Y$ . Then jump to  $Y + 1$ .

- $j = 0$ : Execute return jump regardless of key selections.
- $j = 1$ : Execute return jump if JUMP 1 is selected.
- $j = 2$ : Execute return jump if JUMP 2 is selected.
- $j = 3$ : Execute return jump if JUMP 3 is selected.
- $j = 4$ : Execute return jump. Then stop computation.
- $j = 5$ : Execute return jump. Stop computation if STOP 5 is selected.
- $j = 6$ : Execute return jump. Stop computation if STOP 6 is selected.
- $j = 7$ : Execute return jump. Stop computation if STOP 7 is selected.

## 66 *TERMINATE C<sup>j</sup> INPUT BUFFER*

This instruction terminates the input buffer on channel  $\hat{j}$ . No Input Buffer Monitor Interrupt will occur.

The Operand Interpretation Designator,  $\hat{k}$ , the Index Designator,  $b$ , and the Operand Designator,  $y$ , bits are not translated for this instruction.

## 67 *TERMINATE C<sup>j</sup> OUTPUT BUFFER*

This instruction terminates the output buffer on channel  $\hat{j}$ . No Output Buffer Monitor Interrupt will occur.

The Operand Interpretation Designator,  $\hat{k}$ , the Index Designator,  $b$ , and the Operand Designator,  $y$ , bits are not translated for this instruction.

## 70 *REPEAT*

Clear  $B^7$  and transmit the lower 15 bits of  $Y$  to  $B^7$ . If  $Y$  is non-zero, transmit  $(j)$  to  $r$  (designator register), thereby initiating the *repeat mode*. If  $Y$  is zero, skip the next instruction.

*REPEAT MODE*—The *repeat mode* executes the instruction immediately following the Repeat instruction  $Y$  times;  $B^7$  contains the number of executions remaining throughout the repeat mode.

If no Skip condition is met for the repeated instruction, the *repeat mode* terminates. The instruction following the repeated instruction is then executed. If the Skip condition for the repeated instruction is met, the *repeat mode* terminates, and the instruction following the repeated instruction is skipped.

Following the *repeat mode* termination, the count remains in  $B^7$ . In no way does the *repeat mode* alter the repeated instruction in core memory.

The three lower-order bits of the  $r$  designator are set from  $j$  of instruction 70. It affects operand indexing of the repeated instruction as follows:

- $r = 0$ : Do not modify the operand address of the repeated instruction after each individual execution.
- $r = 1$ : Increase the operand address of the repeated instruction by one after each execution of the repeated instruction.
- $r = 2$ : Decrease the operand address of the repeated instruction by one after each execution of the repeated instruction.
- $r = 3$ : Repeat the initial B-register modification of the repeated instruction before each execution.
- $r = 4$ : Do not modify the operand address of the repeated instruction after each individual execution. If the repeated instruction is a Replace instruction, the operand address is incremented by  $(B^6)$  for the store portion of the Replace Instruction.

\*The  $p$ -designator. Normally set to +1, it is cleared during operation in the *interrupt mode*.

- $r = 5$ : Increase the operand address of the repeated instruction by one after each execution of the repeated instruction. If the repeated instruction is a Replace instruction, the operand address is incremented by  $(B^6)$  for the store portion of the Replace instruction.
- $r = 6$ : Decrease the operand address of the repeated instruction by one after each execution of the repeated instruction. If the repeated instruction is a Replace instruction, the operand address is incremented by  $(B^6)$  for the store portion of the Replace instruction.
- $r = 7$ : Repeat the initial B-register modification of the repeated instruction before each execution. If the repeated instruction is a Replace instruction, the operand address is incremented by  $(B^6)$  for the store portion of the Replace instruction.

NOTE:

*Instruction 70 j designator establishes the repeat mode r designator, since j is transmitted to r.*

### 71 B SKIP ON $B^j$

If the content of B-register  $j$  is equal to  $Y$ , skip the next instruction in the current sequence and proceed to the instruction following. Clear B-register  $j$ .

If the content of B-register  $j$  is not equal to  $Y$ , proceed to the next instruction in the sequence in a normal manner. Increase the content of B-register  $j$  by one.

The Branch Condition Designator,  $j$ , is used to designate the selected B-register in this instruction and is not available for its normal function. Only the lower-order 15 bits of  $Y$  are used in the comparison described in the preceding paragraph.

### 72 B JUMP ON $B^j$

If the content of B-register  $j$  is *non-zero* execute a jump in program address to address  $Y$ . Reduce the content of B-register  $j$  by one.

If the content of B-register  $j$  is *zero*, proceed to the next instruction in a normal manner. Do not alter the content of B-register  $j$ .

The Branch Condition Designator,  $j$ , is used to designate the selected B-register in this instruction and is not available for its normal function. If the Jump condition is satisfied, then the lower-order 15 bits of  $Y$  become the address of the next instruction and the beginning of the new program sequence. The higher-order 15 bits of  $(Y)$  cannot be used in this instruction.

### 73 INPUT BUFFER ON $C^j$ (without MONITOR Mode)

This instruction establishes an input buffer via input buffer channel  $\hat{j}$  to Magnetic Core Storage with an initial storage address  $Y$ . Subsequent to this instruction, individual transfers will be executed at a rate determined by an external device. The storage address initially established by this instruction will be advanced by one preceding each individual transfer. The next current address will be maintained throughout the buffer process in the lower-order 15 bits of Magnetic Core Storage address  $00100$  plus  $\hat{j}$ . This mode will continue until it is superseded by a subsequent initiation or termination of an input buffer via the same input channel or until the higher-order half and the lower-order half of storage address  $00100$  plus  $\hat{j}$  contain equal quantities, whichever occurs first.

This instruction is implemented as follows: If  $\hat{k} = 3$ , store  $(Y)$  in storage location  $00100$  plus  $\hat{j}$ . If  $\hat{k} = 1$ , store the lower-order 15 bits of  $(Y)$  in the lower-order half of storage location  $00100$  plus  $\hat{j}$  leaving the higher-order half undisturbed. If  $\hat{k} = 0$ , store  $Y$  in the lower-order half of storage location  $00100$  plus  $\hat{j}$  leaving the higher-order half undisturbed. Proceed to the next instruction.  $\hat{k} = 2$  is not permitted.



#### 74 OUTPUT BUFFER ON $C^j$ (without MONITOR Mode)

This instruction establishes an output buffer via output buffer channel  $\hat{j}$  from initial storage address Y in Magnetic Core Storage. Subsequent to this instruction, the individual transfers will be executed at a rate determined by an external device. The storage address initially established by this instruction will be advanced by one preceding each individual transfer. The next current address will be maintained throughout the buffer process in the lower-order 15 bits of Magnetic Core Storage address 00120 plus  $\hat{j}$ . This mode will continue until it is superseded by a subsequent initiation or termination of an output buffer via the same output channel or until the higher-order half and the lower-order half of storage address 00120 plus  $\hat{j}$  contain equal quantities, whichever occurs first.

This instruction is implemented as follows: If  $\hat{k} = 3$ , store (Y) in storage location 00120 plus  $\hat{j}$ . If  $\hat{k} = 1$ , store the lower-order 15 bits of (Y) in the lower-order half of storage location 00120 plus  $\hat{j}$  leaving the higher-order half undisturbed. If  $\hat{k} = 0$ , store Y in the lower-order half of storage location 00120 plus  $\hat{j}$  leaving the higher-order half undisturbed. Proceed to the next instruction.  $\hat{k} = 2$  is not permitted.

#### 75 INPUT BUFFER ON $C^j$ (with MONITOR Mode)

This instruction establishes an input buffer via input buffer channel  $\hat{j}$  to Magnetic Core Storage with an initial storage address Y. Subsequent to this instruction, the individual transfers will be executed at a rate determined by an external device. The storage address initially established by this instruction will be advanced by one preceding each individual transfer. The next current address will be maintained throughout the buffer process in the lower-order 15 bits of Magnetic Core Storage address 00100 plus  $\hat{j}$ . This mode will continue until it is superseded by a subsequent initiation or termination of an input buffer via the same input channel or until the higher-order half and the lower-order half of storage address 00100 plus  $\hat{j}$  contain equal quantities, whichever occurs first. Initiation of this input buffer selects the input channel  $\hat{j}$  and establishes a buffer monitor on input channel  $\hat{j}$ . A Monitor Interrupt follows completion of the buffer operation:  $(00100 + \hat{j})_U = (00100 + \hat{j})_L$ .

This instruction is implemented as follows: If  $\hat{k} = 3$ , store (Y) in storage location 00100 plus  $\hat{j}$ . If  $\hat{k} = 1$ , store the lower-order 15 bits of (Y) in the lower-order half of storage location 00100 plus  $\hat{j}$  leaving the higher-order half undisturbed. If  $\hat{k} = 0$ , store Y in the lower-order half of storage location 00100 plus  $\hat{j}$ . Proceed to the next instruction.  $\hat{k} = 2$  is not permitted.

#### 76 OUTPUT BUFFER ON $C^j$ (with MONITOR Mode)

This instruction establishes an output buffer via output buffer channel  $\hat{j}$  from initial storage address Y in Magnetic Core Storage. Subsequent to this instruction, the individual transfers will be executed at a rate determined by an external device. The storage initially established by this instruction will be advanced by one preceding each individual transfer. The next current address will be maintained throughout the buffer process in the lower-order 15 bits of Magnetic Core Storage address 00120 plus  $\hat{j}$ . This mode will continue until it is superseded by a subsequent initiation or termination of an output buffer via the same output channel or until the higher-order half and the lower-order half of storage address 00120 plus  $\hat{j}$  contain equal quantities, whichever occurs first. Initiation of this output buffer selects the output channel  $\hat{j}$  and establishes a buffer monitor on output channel  $\hat{j}$ . A Monitor Interrupt follows the completion of the buffer operation:  $(00120 + \hat{j})_U = (00120 + \hat{j})_L$ .

This instruction is implemented as follows: If  $\hat{k} = 3$ , store (Y) in storage location 00120 plus  $\hat{j}$ . If  $\hat{k} = 1$ , store the lower-order 15 bits of (Y) in the lower-order half of storage location 00120 plus  $\hat{j}$  leaving the higher-order half undisturbed. If  $\hat{k} = 0$ , store Y in the lower-order half of storage location 00120 plus  $\hat{j}$  leaving the higher-order half undisturbed. Proceed to the next instruction.  $\hat{k} = 2$  is not permitted.

Table A-4. CS-1 Compiler – Phase Three

PROBLEM-ORIENTED PROGRAMMING OPERATIONS		
LABEL	OPERATOR	OPERANDS
[label]	➔ SYSTEM	• [programmer's name] • [date]
	➔ SYS-INDEX	• [B-register] • [data name] <sup>(8)</sup>
	➔ SEL-DD	• [label of SYS-DD]
	➔ SEL-PROC	• [label of SYS-PROC, key] • [label, key of each non-unique label]
	➔ SEL-SYS	• [key] • [label, key of each non-unique label]
[label]	➔ SYS-DD	• [programmer's name] • [date]
	➔ END-SYS-DD	
	➔ LINK	
	➔ END-LINK	
[label]	➔ SYS-PROC	• [programmer's name] • [date]
	➔ LOC-DD	
	➔ TABLE	• [name] <sup>(2)</sup> • H (or V) • [words/item] • [max. items] • [name maj. index] <sup>(1)</sup>
	➔ SUB-TABLE	• [name] <sup>(2)</sup> • [initial item no.] • [max. items] • [name maj. index] <sup>(1)</sup>
	➔ FIELD	• [name] • FXPOS (or FXWS, MW) • [word loc.] • [no. words or bit pos.] • [binary point] <sup>(1)</sup>
	➔ ITEM-AREA	• [name] • . . . .
	➔ END-TABLE	• [name] <sup>(2)</sup>
	➔ VRBL	• [name] • FXW (or FXH, FXHPOS) • [binary point] <sup>(1)</sup>
	➔ VRBL	• [name] • FXL (or FXU, FXLPOS, FXUPOS) • [name of FXW variable] • [binary point] <sup>(1)</sup>
	➔ SWITCH	• [name] • [statement label (s)] • . . . .
	➔ SWITCH	• [name] <sup>(4)</sup>
	➔ S	• [statement label] <sup>(4)</sup>
	➔ END-SWITCH <sup>(4)</sup>	• [name]
	➔ END-LOC-DD	
[label]	➔ DATA	• [constant], [binary point] • . .
	➔ PROCEDURE	• [name] • INPUT • [formal name(s)] • OUTPUT • [formal name(s)] • EXIT • [formal name(s)]
	➔ LOC-INDEX	• [name] • . . . .
	➔ SET	• [data name] • TO (or EQ) • [data name, const., alg. exp.] • . . . SAVING • REMAINDER <sup>(1)</sup> • . . . DIVFLT • [label] <sup>(1)</sup> • . . . .
	➔ GOTO	• [statement label]
	➔ GOTO	• [switch name] • [switch setting]
	➔ IF	• [data name] • [decider] <sup>(5)</sup> • [data name, const., alg. exp.] • AND <sup>(1)</sup> • . . . . OR <sup>(1)</sup> • . . . . THEN • . . . .
	➔ IF	• DATA • VALID (or INVALID) • THEN • . . . .
	➔ IF	• [data name] • ODDP (or EVENP) • THEN <sup>(1)</sup> • . . . .
[label]	➔ VARY	• [data name] • [prepositional operand] <sup>(6)</sup>
	➔ RESUME	• [FIND or VARY label]
	➔ END	• [VARY label]
[label] <sup>(7)</sup>	➔ FIND	• [data name] • [decider] <sup>(5)</sup> • [data name, const., alg. exp.] • VARYING <sup>(1)</sup> • * • [prepositional operand] <sup>(1)</sup> • *, <sup>(6)</sup>
	➔ IF	• DATA • FOUND (or NOT FOUND) • THEN • . . . .
	➔ RETURN	• [formal statement label] <sup>(1)</sup> • STOP (or STOP 5, 6, 7) <sup>(1)</sup>
	➔ RETURN	• RIL <sup>(3)</sup>
	➔ [procedure name]	• INPUT <sup>(1)</sup> • [data name, const., alg. exp.] • OUTPUT <sup>(1)</sup> • [data name(s)] • EXIT <sup>(1)</sup> • [statement label(s)]
	➔ P-SWITCH	• [switch name] • INPUT • [formal name(s)] • OUTPUT • [formal name(s)]
	➔ P	• [procedure name]
	➔ END-P-SW	• [switch name]
	➔ [p-switch name]	• USING • [switch index] • INPUT • [data name, const., alg. exp.] • OUTPUT • [data name] • THEN • . . . .
	➔ TYPE	• [data name] • [data name] • . . . . THEN <sup>(1)</sup> • . . . .
	➔ TYPE-TEXT	• [text and flex commands]
	➔ PUNCH	• [data name] • [data name] • . . . . THEN <sup>(1)</sup> • . . . .
	➔ PUNCH-TEXT	• [text and flex commands]
	➔ FORM	• [buffer name] • [initial char. position] • [data name] • THEN <sup>(1)</sup> • . . . .
	➔ FORM-TEXT	• [buffer name] • [initial char. position] • [text]
	➔ PRINT-BUF	• [base addr.] • [jump cond.]
	➔ PRINT-TBL	• [data name] • [data name] • THEN <sup>(1)</sup> • . . . .
	➔ PUT-ADR	• [data name] • IN • [data or reg. name] • THEN <sup>(1)</sup> • . . . .
	➔ COMMENT	• [message]
	➔ STOP	
	➔ END-PROC	• [name]

LIST OF DECIDERS		LIST OF PREPOSITIONAL OPERANDS		
CODE	MEANING	CODE	OBJECT FORMS PERMITTED	OBJECT PRESCRIBES
• EQ •	= Equal	• FROM	• Data Name, Alg. Exp., Constant, Index	Starting Point
• NOT •	≠ Not Equal	• THRU	• Data Name, Alg. Exp., Constant, Index	Terminal Point
• LTEQ •	≤ Less Than or Equal To	• BY	• Data Name, Alg. Exp., Constant, Index	Index Increment
• LT •	< Less Than	• WITHIN	• Table or Sub-Table Name	Table or Sub-Table Parameters
• GT •	> Greater Than			
• GTEQ •	≥ Greater Than or Equal To			

(1) Use is optional (3) Only 1 Return permitted in each interrupt procedure (6) See list of prepositional operands  
 (1)\* Optional if varied throughout table (4) Used in switch table design (7) Required if return made by RESUME or GOTO  
 (2) Limited to 5 alphanumeric characters (5) See list of deciders (8) Reserves B-register for data unit throughout program

**SUPPORT OPERATIONS**

COMPILER-CONTROL OPERATIONS			
LABEL	OPERATOR	OPERANDS	COMMENTS
[Label]	➔ C-CONTROL	• [programmer's name] • [date]	General Header for Compiler Control Operators
	➔ P-IGNORE	• [procedure name] • [procedure name] • •	Specifies procedures (of a procedure chain) not to be compiled
	➔ DEBUG-AIDS	• [output no.] • [output no.] • • • •	Informs compiler that Debugging Aids are desired
	➔ OUTPUTS	• [output no.] • [output no.] • • • •	Informs compiler which outputs are desired in object program
	➔ CHAN-SET		Operator heading communications channel assignments
[label]	➔ [input/output assignment]		Specifies desired communication channel assignments
	➔ EXCHANGE		Operator heading element exchanges
[old designation]	➔ [new designation]		Specifies new element designation for the old designation
	➔ ALLOCATION		Operator heading normal allocation instructions
BASE	➔ [absolute address]		Specifies Initial Address for compiler allocation of L <sub>4</sub> object program
ENTRANCE	➔ [primary procedure name or S/R label]		Generates manual entrance at object program's base address
S ENTRANCE	➔ [primary procedure or S/R label]		Generates manual entrance at object program's base + 1 or RJP entrance at its base
[label]	➔ [numeric allocation value, label, tag]		Direct allocation instruction format
DEBUG	➔ [absolute address]		Specifies initial address for Debugging Package when not placed at 76000
TABLE POOL	➔ [absolute address]		Specifies initial POOL address for table allocation
[label]	➔ DELETE		Deletes indicated label from compilers allocation tables
	➔ INDR-ALLOC		Operator heading indirect allocation instructions
[S/R label]	➔ [6 digit number, k and y]		Specifies memory cell (k-designator and address) containing initial address of subroutine
	➔ REL-ALLOC		Operator heading relative allocation instructions
[label]	➔ [increment alloc. value]		Specifies increment to a given base address (place new base in B <sup>7</sup> while loading)

**DEBUGGING OPERATIONS**

➔ DEF-AREA	• [area name] • [initial area tag, label, address] • [no. of words]	
➔ CORE-IMAGE	• [area name] • [initial image tag, label, address] • KEY 1 (or 2, 3) <sup>(1)</sup>	
➔ TEST-IMAGE	• [area name] • [area name] • • • • • KEY 1 (or 2, 3) <sup>(1)</sup>	
➔ DUMP-REG	• KEY 1 (or 2, 3) <sup>(1)</sup>	
➔ DUMP-AREA	• [area name] • [area name] • • • • • KEY 1 (or 2, 3) <sup>(1)</sup>	(1) Use is optional

**PROGRAM CORRECTION OPERATIONS**

[label]	➔ CORRECT - L <sub>1</sub> • [programmer's name] • [date]	Operator heading list of program corrections
[L <sub>1</sub> ID • Ins No]	➔ [insert operation]	Specifies L <sub>1</sub> -ID with insert number for insert position
	➔ [insert operation]	Indicates insert to be made
[L <sub>1</sub> ID]	➔ DELETE (or Replacement Operation)	Specifies L <sub>1</sub> -ID for correction position
		Indicates correction to be made

**LIBRARY UPDATING OPERATIONS**

[label]	➔ LIBRARY
	➔ INS-DD • [label] • [programmer's name] • [date]
	➔ RPL-DD • [label] • [programmer's name] • [date]
	➔ DEL-DD • [label] • [programmer's name] • [date]
	➔ INS-PROC • [label], [key] • [programmer's name] • [date]
	➔ RPL-PROC • [label], [key] • [programmer's name] • [date]
	➔ DEL-PROC • [label], [key] • [programmer's name] • [date]

**LIBRARY LISTING OPERATIONS**

[label]	➔ LIBRARY
	➔ LIST-PROC • [label], [key]
	➔ LIST-DIR
	➔ LIST-DIR • [directory number] • [directory number] • •
	➔ LIST-DD • LIBRARY
	➔ LIST-PROC • LIBRARY
	➔ LIST-DD • [label]
	➔ LIST-DD • HISTORY

## TYPICAL INSTRUCTION WORDS

The following example of machine coding is intended to illustrate the use of UNIVAC 1206 Instruction Repertoire. It represents a portion of a computer program; in octal notation, it specifies the contents of several successive core memory storage locations.

STORAGE ADDRESS	f	j	k	b	y		
0 2 0 0 0	1	0	2	1	3	3 0 0 0 0	}
0 2 0 0 1	6	1	0	0	0	2 0 2 0	
0 2 0 0 2	1	1	0	0	0	0 0 0 0	
0 2 0 0 3	2	3	2	2	0	3 0 0 1 0	}
0 2 0 0 4	6	1	0	0	0	2 0 2 0	
0 2 0 0 5	1	4	0	3	0	3 0 0 0 7	

Execution proceeds as follows:

- a) Enter the Q-register with the lower 15 bits of the word stored at memory address: 3000<sub>8</sub> (plus the contents of B-register 3). If the operation makes the contents of the Q-register negative, jump to memory address 0202<sub>8</sub>, if not, clear the A-register.
- b) Divide the number now stored in the Q-register by the entire 30-bits of the number stored at memory address 3001<sub>8</sub>. If the operation results in an overflow, jump to memory address 0202<sub>8</sub>, if not, store the entire 30-bits of the Quotient at memory address 3007<sub>8</sub>.

# UNIVAC®

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

## REGIONAL OFFICES

WASHINGTON 7, D.C.  
2121 WISCONSIN AVE. N.W.  
P.O. BOX 3657  
338-8510

DAYTON, OHIO  
20 NORTH JEFFERSON  
224-0755

COCOA BEACH, FLORIDA  
CANAVERAL COMMERCIAL CENTER  
1 WEST UNIVAC BLVD.  
783-8461

LEXINGTON 73, MASS.  
1776 MASSACHUSETTS AVE.  
862-2650

GREAT NECK, LONG ISLAND, N.Y.  
SPERRY GYROSCOPE BLDG.  
574-3480

LOS ANGELES 45, CALIF.  
SUITE 220  
5316 W. IMPERIAL HIWAY  
678-2531