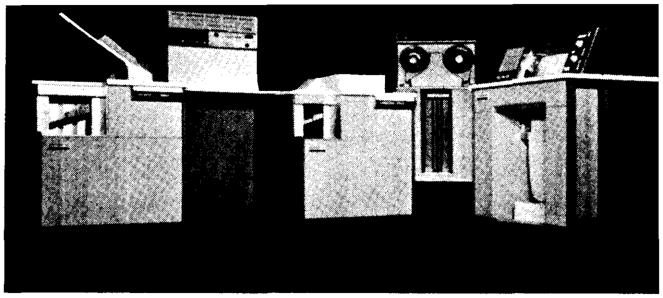


GENERAL REFERENCE MANUAL

UNIVAC® 1050 SYSTEM

PRINTED IN U.S.A.

CONTENTS

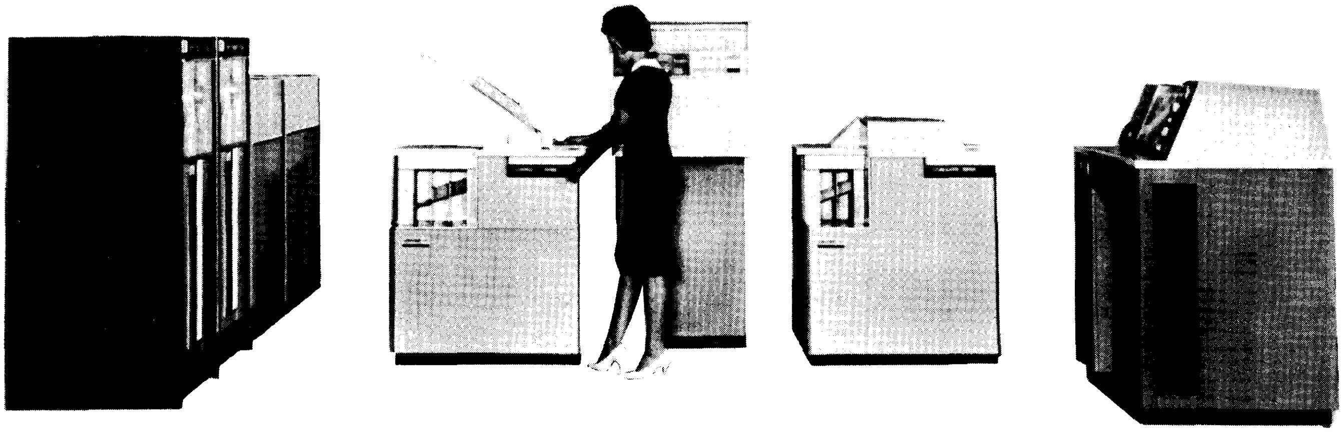
ILLUSTRATIONS

TABLES

This manual is published by the UNIVAC Division in loose leaf format as a rapid and complete means of keeping recipients apprised of UNIVAC Systems developments. The UNIVAC Division will issue updating packages, utilizing primarily a page for page or unit replacement technique. Such issuance will provide notification of hardware and/or software changes and refinements. The UNIVAC Division reserves the right to make such additions, corrections, and/or deletions as, in the judgment of the UNIVAC Division, its respective Systems development's may require.

1. FOREWORD

1. FOREWORD



This manual introduces the UNIVAC® 1050 System and describes the system's hardware and software aspects. Information concerning the system's uses, programming, operation and installation is also included. Moreover, the manual relates the approach to the design of the system and the provisions Sperry Rand Univac has made for its efficient installation in office, plant or laboratory. It also specifies what Univac will do to help the user put the UNIVAC 1050 System into productive use.

Because the manual will reach a varied audience, an attempt has been made to provide information of interest to users of business and scientific computing systems; operators and supervisors of punched card equipment; programmers of small to large scale equipment; and managers of research, engineering, financial, production and marketing departments.

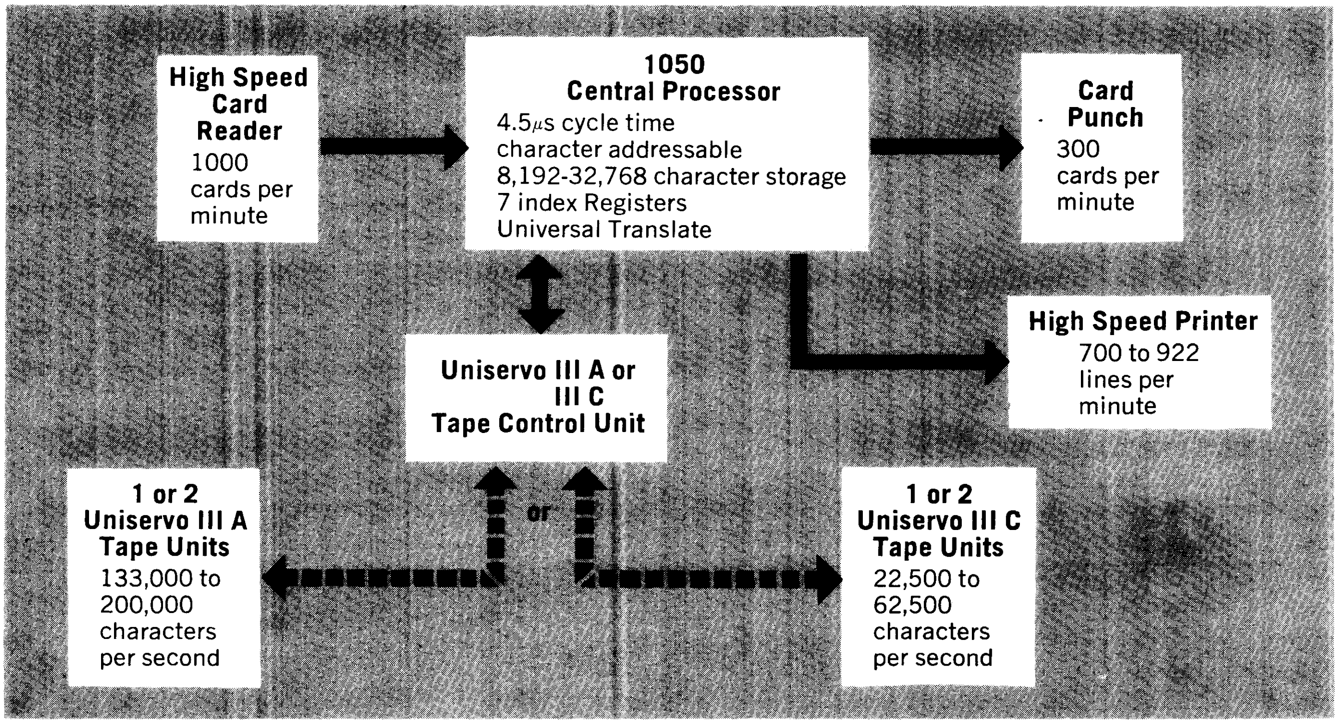
Naturally, not all the information about all aspects of the 1050 can be contained in this single manual. Subsequent manuals will present complete details about subjects introduced here. However, after reading this manual, the reader should be able to visualize the UNIVAC 1050 System at work in his environment, and will then have some specific questions which the Univac representative will answer for him.

2. GENERAL DESCRIPTION

2. GENERAL DESCRIPTION

The UNIVAC 1050 System may be used as a subsystem or auxiliary device to large scale equipment such as the UNIVAC III, 490, or 1107 Systems. In addition, the UNIVAC 1050 System can be effectively utilized in a remote plant or branch office location as an integral part of a centralized electronic data processing system.

The UNIVAC 1050 System consists of a Central Processor and a UNISERVO tape handling unit and any of the following: a High Speed Printer, a High Speed Card Reader, a Card Punch Unit, and a second UNISERVO tape handling unit. The UNISERVO tape handling units may be either UNISERVO III A units for the UNIVAC compatible satellite system or UNISERVO III C units for the IBM compatible satellite system, but not one of each. Standard configurations and software available for each are shown in section 5 (Figure 5-1).



High Speed Card Reader
1000 cards per minute
1050 Central Processor
4.5μs cycle time
character addressable
8,192-32,768 character storage
7 index Registers
Universal Translate
Card Punch
300 cards per minute
High Speed Printer
700 to 922 lines per minute
Uniservo III A or III C Tape Control Unit
or
1 or 2 Uniservo III A Tape Units
133,000 to 200,000 characters per second
1 or 2 Uniservo III C Tape Units
22,500 to 62,500 characters per second

Figure 2-1. Block Diagram of a Typical System.

CONCEPT OF DESIGN

The design objective of the UNIVAC 1050 System was to provide a system to relieve large scale systems of the burden of housekeeping functions detracting from the critically important tasks of computation, analysis, and the preparation of management reports. Included among these housekeeping or auxiliary functions are the necessary operations of converting information from punched cards to magnetic tape, magnetic tape to punched cards, or magnetic tape to printed hard copies, and the maintenance of master record tapes.

The UNIVAC 1050 System can accept many different codes and convert them efficiently. Moreover, the satellite system can be called upon to operate at speeds commensurate with the powerful large scale system's demands and at the same time, perform highly complex editing operations. Furthermore, the nature of the attendant functions peculiar to each installation demand a custom configuration of the satellite system and thus modularity is a key factor in dictating the design of the UNIVAC 1050 System.

With the experience and ingenuity that made UNIVAC the first name in automatic data-processing systems, the UNIVAC 1050 System was designed with features that surpassed these stringent requirements making it the foremost satellite computing system available.

FEATURES

All of the features listed below are standard for the UNIVAC 1050 System except for the expandability of memory. This allows for a high degree of cross-system compatibility.

A HIGH DEGREE OF COMPATIBILITY WITH LARGE SCALE SYSTEMS

The UNIVAC 1050 System can accept tapes from or prepare tape for: the UNIVAC III, 490, and 1107 Systems, or the IBM 1410, 705, 7070, 7080, and 7090 System. The UNIVAC 1050 System can read or punch any card code including straight binary code. The system can automatically translate from standard 80-column Hollerith code to the internal UNIVAC 1050 code, and vice versa. The system has an instruction that automatically translates a block of data from any one 6-bit code to any other 6-bit code.

HIGH RATES OF SPEED ON ALL PERIPHERAL UNITS

High-Speed Reader —	1,000 cards per minute
Card Punch Unit —	300 cards per minute
High-Speed Printer —	700-922 lines per minute
IIIA Tape Transfer — Rate (UNIVAC Mode)	100,000-133,000 characters per second 150,000-200,000 digits per second
III C Tape Transfer — Rate (IBM Mode)	22,500 or 62,500 characters per second

FAST INTERNAL STORAGE SPEED

The core storage of the UNIVAC 1050 System has a cycle time of 4.5 microseconds.

PROGRAMMING SIMPLICITY

A joint software and hardware development has resulted in an efficient programming system which takes full advantage of the powerful instruction repertoire. The software package contains:

PAL Assembler

An easy to use, easy to learn assembly system which includes input-output macro-instructions, diagnostic macro-instructions, as well as specializable precoded input-output routines.

Operator

An integrated Co-ordination Routine and Relocatable Relative Loader which allows the operator to communicate with the system and to load and start programs that will operate independently or concurrently. The Co-ordination Routine assures effective utilization of the various input-output channels.

Source Code Librarian

A routine which facilitates the maintenance of a tape file of source programs.

Patch Assembler

A routine which allows the user to make source code changes, insertions and deletions to the object code.

Input-Output Library

A complete set of input-output routines which can be specialized and incorporated with the users program at assembly time.

Regent

A report program generator which automatically translates report requirements into machine language programs.

Utility Routines

A standard set of Utility Routines will be provided as part of the UNIVAC 1050 System software package. These routines will allow the user to perform simple card-to-tape, tape-to-printer and tape-to-card operations. Standard data tape conventions will be used.

All of the above software aids reduce over-all program preparation time.

MODULAR DESIGN

The magnetic core storage is available in modules of 4096 six-bit alphanumeric characters and may be expanded from a minimum of two (8192 characters) to a maximum of eight modules with a total capacity of 32,768 alphanumeric characters.

INDEX REGISTERS

Seven built-in index registers are included for programming ease and processing efficiency.

VARIABLE FIELD LENGTH

Data fields are variable in length to enable flexible and efficient use of the storage and the data-processing capacity of the system.

FLEXIBLE USE OF STORAGE

Any area of main storage can be used for input-output buffer storage. Also, blocks of input or output data can be speedily transferred from one section of storage to another.

EDITING EFFICIENCY

Editing instructions are provided for automatically and efficiently inserting punctuation marks, symbols, and characters and suppressing non-significant commas and zeros in an output field. An editing pattern may be established for use on successive output fields.

COMPREHENSIVE INSTRUCTION REPERTOIRE

The system contains a complete and comprehensive instruction repertoire with commands for performing both binary and decimal arithmetic operations, including decimal multiplication and division.

AUTOMATIC PROGRAM INTERRUPT

The system contains an interrupt network which facilitates maximum utilization of the UNIVAC 1050 peripheral devices. Interrupts also indicate to the user when decimal overflow and improper division is attempted. Central Processor error faults and emergency conditions also cause automatic interrupt so that a possible computer malfunction will be detected.

OPERATING SIMPLICITY

The operator's tasks can be performed quickly and easily. The system is designed to operate automatically with a minimum of monitoring.

UNIVAC SERVICES

In support of its design, production, and distribution of data-processing systems, Sperry Rand Univac supplies a wide range of services to ensure the effective and profitable use of each system. To begin with, an information service is maintained to afford accurate and complete reference material for all users. This manual represents the initial phase of the service which will be extended through forthcoming manuals. To reinforce these efforts, Univac representatives will obtain answers to questions not anticipated in the reference manuals.

Univac provides consultant services for systems analysis and programming. The scope of these services begins with feasibility studies, and related systems and cost analyses. It continues with system and subsystem design, including card and tape file layout, printed report formatting, and over-all system block diagramming. Also included are: consultation on personnel planning and peripheral equipment and supplies; selection of appropriate software to use in program preparation; provision for necessary training and programming aids; and technical guidance in the writing and checkout of initial programs. In addition, the services provide recommendations for the establishment of good practices for programming and operating procedures.

The UNIVAC 1050 System is designed with reliability as a primary goal. Highly reliable solid-state circuit components, such as transistors and solar-cell detectors, are used throughout the system. Both the circuits and mechanical assemblies are conservatively designed with built-in safety margins that enable them to operate correctly under adverse conditions. Extensive operation and testing have proven the reliability inherent in the design of each input-output device. The basic printer design, for example, has been proven by years of mass-production printing on the UNIVAC I, UNIVAC II, UNIVAC Solid-State and UNIVAC LARC Systems. The printer of the 1050 System is a high-performance device with a reliability matching that of the electronic circuits in the system.

To ensure that reliability is kept up to its initial high level, a competent staff of Univac service engineers is available to service the 1050 System. Univac service engineers follow a strict preventive-maintenance program designed to remove the cause of a potential failure in the system before it develops. Periodically the system is thoroughly tested and serviced. Marginally operating components are replaced before they can cause a failure. Additionally, between periodic systems tests, the user himself may test the performance of the system. A complete set of programs is furnished for this purpose.

3. COMPONENT DESCRIPTION

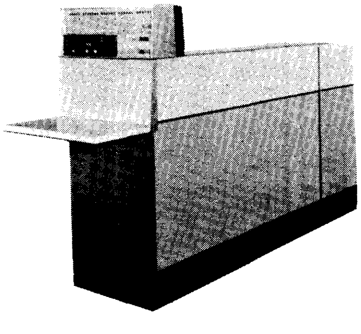
3. COMPONENT DESCRIPTION

The system consists of a Central Processor with magnetic core storage, and may include a High-Speed Reader, a Card Punch Unit, a High-Speed Printer, and one or two tape units (either UNISERVO III A or UNISERVO III C).

The basic UNIVAC 1050 System has been designed for future expansion to meet the constantly growing data-processing requirements of modern business. The memory capacity of the System can be expanded from two to eight memory modules.

The expansion inherent in the UNIVAC 1050 System is achieved without in any way compromising the basic design. All systems are capable of on-site expansion to any larger configuration (up to the maximum) with a minimum of interruption and elapsed time. Expanding the storage capacity, for example, involves little more than plugging in additional printed-circuit cards and core-storage units.

CENTRAL PROCESSOR



The Central Processor houses the control, arithmetic, storage, indexing, and other processing components which, operating at program direction, perform the input-output, logical and arithmetic functions of the system.

Storage Capacity

The UNIVAC 1050 System employs an expandable magnetic core storage capable of receiving or dispensing data at a rate of one alphanumeric character every 4.5 microseconds. Core storage is divided into modules, each having a capacity of 4096 six-bit alphanumeric characters, each of which is addressable.

Core storage is character addressable (positions 0-32,767). For ease in presentations and illustration, this storage can be considered as rows of information with 64 positions (columns) in each row. The first row (row 0) in memory contains positions 0-63; the second row 64-127, etc.

Tetrads

The first 256 characters (4 rows) of storage are grouped into 64 fields of 4 characters each. These fields, called *Tetrads* can be addressed as four character groups. They are used as Arithmetic Registers, Index Registers, and fields specifying origins and destinations of data for input-output operations. Individual characters of a Tetrad may also be addressed. See Figure 3-1 for special assignments.

Arithmetic Registers

Two 16-character Arithmetic Registers are provided to reduce the necessity of storing intermediate results. The Arithmetic Registers can be addressed as AR1 or AR2; 4-character groups within an AR can be addressed as Tetrads (0-7) or a single character within an AR can be addressed as a storage position (00-31).

Index Registers

There are seven Index Registers. The low order three characters of Tetrads 9-15 are used to store the Index Registers. Index Register 1 is contained in Tetrad 9, Index Register 2 in Tetrad 10, . . . Index Register 7 in Tetrad 15.

Input-Output Channels

The core storage serves as the main communication link between the Central Processor circuitry and the input-output control units. It is a key part of a balanced system of internal communication and control that enables input and output operations to proceed continuously, at full speed, simultaneously with the processing of data. There are five assigned input-output channels which are used to connect the input-output control units to the core storage and the input-output controls in the Central Processor. A list of the assigned channels and their allocations are:

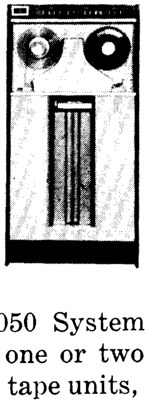
CHANNEL	INPUT-OUTPUT
0	Printer
1	Card Reader
2	Card Punch
4	Magnetic Tape Read
5	Magnetic Tape Write

Associated with each of the Input-Output channels is a fixed storage area each consisting of a group of four consecutive Tetrads (see storage assignment, Figure 3-1). These areas contain information for the control of the peripheral devices such as the number of lines of paper advance before the next line is printed, the location where information from the card reader is to be sent, and so on.

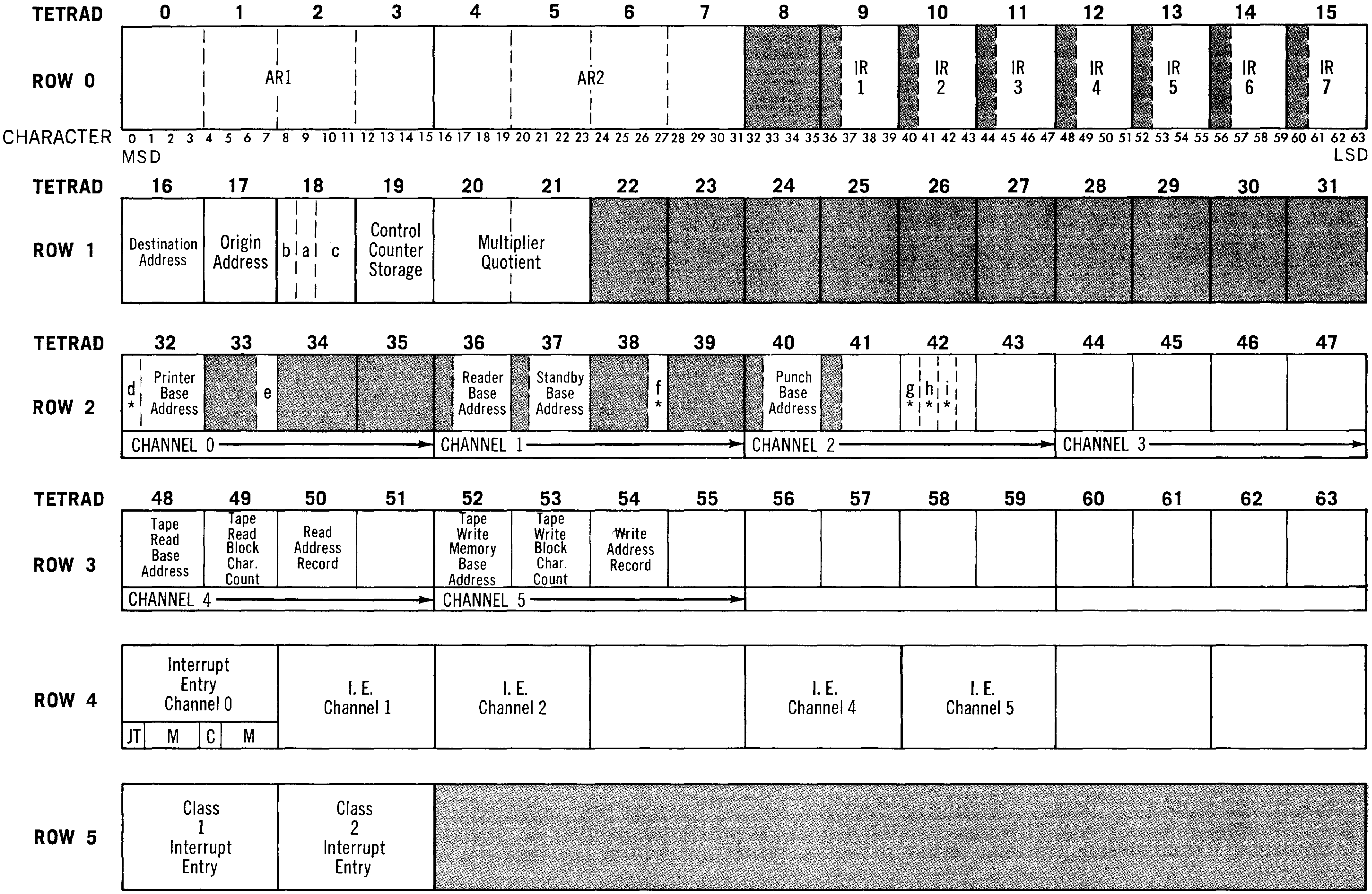
There are no fixed storage areas reserved for input-output data. Data read from a card or data to be punched on a card or printed on a line can be stored in any area of the storage that is not being used for other purposes. The only restriction is that the beginning of each card input or output area must coincide with the beginning of a 64-character row of the storage.

Also associated with each channel is an interrupt entry. The information in this area is necessary for the optimum operation of the various peripheral devices and will be explained in a later section of the manual.

UNISERVO TAPE UNITS



In the UNIVAC 1050 System the user has the option of including one or two UNISERVO III A or UNISERVO III C tape units, but not one of each. The UNISERVO III A tape units are magnetic



TETRAD	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ROW 0	AR1				AR2				▨	▨ IR 1	▨ IR 2	▨ IR 3	▨ IR 4	▨ IR 5	▨ IR 6	▨ IR 7
CHARACTER	0 1 2 3 MSD	4 5 6 7	8 9 10 11	12 13 14 15	16 17 18 19	20 21 22 23	24 25 26 27	28 29 30 31	32 33 34 35	36 37 38 39	40 41 42 43	44 45 46 47	48 49 50 51	52 53 54 55	56 57 58 59	60 61 62 63 LSD

TETRAD	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ROW 1	Destination Address	Origin Address	b a c	Control Counter Storage	Multiplier Quotient		▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

TETRAD	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
ROW 2	d* Printer Base Address	▨ e	▨	▨	▨ Reader Base Address	▨ Standby Base Address	▨ f*	▨	▨ Punch Base Address	▨	g* h* i*					
	CHANNEL 0 →				CHANNEL 1 →				CHANNEL 2 →				CHANNEL 3 →			

TETRAD	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
ROW 3	Tape Read Base Address	Tape Read Block Char. Count	Read Address Record		Tape Write Memory Base Address	Tape Write Block Char. Count	Write Address Record									
	CHANNEL 4 →				CHANNEL 5 →											

ROW 4	Interrupt Entry Channel 0 JT M C M	I. E. Channel 1	I. E. Channel 2		I. E. Channel 4	I. E. Channel 5		
ROW 5	Class 1 Interrupt Entry	Class 2 Interrupt Entry	▨	▨	▨	▨	▨	▨

a = zero count
b = translation table address
c = block transfer count
*d = character count
e = line advance count

*f = row count
*g = hole count (post-punch read and punch)
*h = hole count (wait and pre-punch)
*i = row count (punch)

*used **only** by control units

Figure 3-1. Storage Assignment Chart.

tape units compatible with large scale UNIVAC Systems, while the UNISERVO III C units are compatible with IBM Systems.

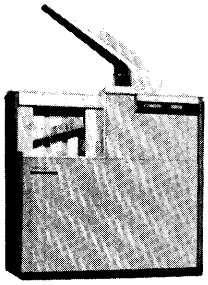
There is a Control Unit for either type of tape unit which accepts, interprets and acts upon control signals, instructions, and data from the Central Processor. The Control Unit converts the data to appropriate tape formats for recording.

The Control Unit indicates its status and the status of the available tape units to the computer through testable indicators. These indicators are explained in a later section.

The following table presents the characteristics of the two different types of tape units:

CHARACTERISTICS	UNISERVO III A TAPE UNIT	UNISERVO III C TAPE UNIT
TAPE SPEED	100 inches per second	112.5 inches per second
PULSE DENSITY	1000 pulses per inch	200 and 556 pulses per inch
SPACE BETWEEN BLOCKS	0.75 inch	0.75 inch
DATA TRANSMISSION RATE	133 KC Max (Alphanumeric) 200 KC Max (Numeric) 100 KC (III, 490, 1107 compatible)	22.5 KC and 62.5 KC
REWIND TIME	125 Seconds For Tape 3500 Feet In Length	87 Seconds For Tape 2400 Feet In Length
REVERSAL TIME	600 Milliseconds	
STOP/START TIME (DIFFERENT SERVOS)	7.8 Milliseconds	10.2 Milliseconds
STOP/START TIME (SAME SERVO)	13.2 Milliseconds	14.2 Milliseconds
READ/WRITE OPERATION	Reading in forward and backward directions; writing in the forward direction only	Reading and writing operations proceed in the forward direction only
TAPE WIDTH	½ inch	
TAPE BASE THICKNESS	0.001 inch	0.0015 inch ± 7.5%
TAPE LENGTH	600 feet 1800 feet 3500 feet	2400 feet
REEL CHANGE TIME	30 Seconds	
CHANNELS ON TAPE	9 Channels – 2 Parity bits for each 3 frames 7 channels – 6 data, 1 parity	7 channels – 6 data, 1 parity
FILE PROTECTION	When the Write Enable ring is inserted, a write operation can be effected	
WRITE CHECKING	This ensures that no errors or bad spots pass undetected	

HIGH-SPEED READER



CHARACTERISTICS	
CARDS PER MINUTE	1000
CAPACITY OF INPUT HOPPER	3000 STANDARD-THICKNESS CARDS
NUMBER OF OUTPUT STACKERS	3
CAPACITY OF EACH OUTPUT STACKER	1000 STANDARD-THICKNESS CARDS

The High-Speed Reader senses 80- or 90-column cards at a maximum rate of 1000 cards per minute. Sensing is accomplished by highly reliable solar-cells. Before each read operation, all of these cells are automatically checked. This feature insures accuracy in the information transmitted to the Central Processor.

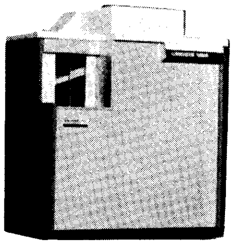
Ninety column cards are placed in the input hopper face up with the row 9 edge leading; 80-column cards are placed face down with the row 9 edge leading.

The maximum reading rate is easily achieved with the aid of the interrupt network. A program interrupt occurs every card cycle (60 milliseconds), unless inhibited. Card feed orders are issued at this point, if possible, maintaining the maximum rate. Storage access times required are less than 5 milliseconds for the 80-column system and less than 3.8 milliseconds for the 90-column system; leaving most of card cycle time available for processing.

The input card hopper has a capacity of 3000 standard-thickness cards. Small or large quantities of cards can be easily loaded while the reader is operating. There are three output stackers each with a capacity of 1000 standard-thickness cards. When the first stacker is full, card output is automatically switched to the second stacker. When the operator has removed the cards from the first stacker and depressed the stacker reset button, card output is switched back to the first stacker. Normally, the third stacker serves as a reject stacker and receives cards following the detection of an error.

As a customer option, the card reader can be equipped to read stub cards (51 column size on 80 column reader or 29/58 column size on 90 column reader) at 1000 cards per minute, as well as full-size cards. The stub cards are commonly used in billing, inventory control and many other applications. The card reader can also be equipped, as a customer option, to read cards of post-card thickness, as well as standard-thickness cards. Cards may be perforated to allow for subsequent separation or scored to allow for subsequent folding.

CARD PUNCH UNIT



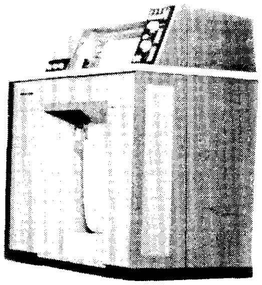
CHARACTERISTICS	
CARDS PER MINUTE	300
CAPACITY OF INPUT HOPPER	1000 STANDARD-THICKNESS CARDS
NUMBER OF OUTPUT STACKERS	2
CAPACITY OF EACH OUTPUT STACKER	850 STANDARD-THICKNESS CARDS

The Card Punch Unit can punch 80-column or 90-column cards at a maximum rate of 300 cards per minute. Under control of the processor programs, cards may be selectively punched or selected cards may be advanced without punching. There are four stations in the card transport system; two wait stations, a punch station and a post-punch check-read station. The post-punch check-read station enables a positive hole-count check to be made of data that was previously punched.

Maximum rates can easily be attained as the punch causes a program interrupt at the completion of its cycle, whereupon another punch order can be issued.

The card input hopper has a capacity of 1000 standard-thickness cards, which are placed in the hopper in the same manner as in the card reader. Small or large quantities of cards can be easily loaded while the Punch Unit is operating. There are two output stackers each with a capacity of 850 standard-thickness cards. Normally cards are directed to only one of the stackers. The second stacker functions as a reject stacker that receives cards following the detection of an error in the Card Punch Unit.

HIGH-SPEED PRINTER



CHARACTERISTICS	
LINES PER MINUTE (SINGLE-SPACED, ALPHANUMERIC)	700 TO 922
CHARACTERS PER LINE	128
LINES PER INCH (VERTICAL)	6 OR 8 AT OPTION OF OPERATOR
CHARACTERS PER INCH (HORIZONTAL)	10
NUMBER OF PRINTABLE CHARACTERS	63
CONTINUOUS PAPER-FEED RATE	20 INCHES PER SECOND
PAPER STOCK	ANY SPROCKET-FED PAPER, 4 TO 22 INCHES WIDE, UP TO AND INCLUDING CARD STOCK-THICKNESS, EITHER BLANK OR PREPRINTED FORMS
COPIES	AN ORIGINAL AND AT LEAST FIVE CARBON COPIES

A buffered unit, the High-Speed Printer operates under control of the processor program producing high-quality multiple-copy records in a completely edited format. At the option of the program, either 128-character full lines or 64-character half lines may be printed. The paper can be fed in steps for single-line spacing or fast fed for multiple-line spacing under control of the program. With single-line spacing, data may be printed at a rate ranging from 700 to 922 lines per minute depending upon the range of characters employed by the processor program. Maximum rates can easily be attained as the printer causes a program interrupt at the completion of its cycle, whereupon another print order can be issued.

A standard print drum with the symbols shown in table 5-1 is provided. At an additional cost, print drums may be ordered having any combination of 63 characters including special symbols and non-English alphabetic characters.

A standard-size shipping container for continuous paper forms can be accommodated within the base of the printer. The paper will feed directly from the container. Continuous paper forms are automatically stacked during printing on an adjustable shelf on the rear of the printer. To ensure proper stacking of the paper, the printer is designed to prevent the build-up of static electricity.

Paper forms with stock ranging 4 to 22 inches in width can be easily and accurately positioned on the printer. Numbered calibrations on the printer enable the operator to record the position of a form and set the same type of form to the recorded position at a later date. Fine adjustments are provided that enable the operator to shift the paper horizontally or vertically the space of one character or line or less in either direction. This adjustment can be performed either while the printer is operating or while it is in a standby condition.

AUTOMATIC PROGRAM INTERRUPT

Automatic program interrupt is a technique provided to permit efficient utilization of all the UNIVAC 1050 peripheral devices and to monitor the performance of the UNIVAC 1050 Central Processor. The technique operates as follows: when a condition which requires immediate attention occurs, the program which is running is temporarily interrupted to service the condition. After the condition has been serviced, control is returned to the interrupted program at the point of interruption. The type of servicing required depends upon the class of interrupt. Three classes of interrupt are provided:

- **CLASS I** – This class of interrupt occurs when an error or fault condition arises in the Central Processor. The servicing here brings the processor to a halt.
- **CLASS II** – This class of interrupt occurs when improper division is attempted, decimal overflow occurs or when the operator interrupt switch is depressed. The servicing here is variable and depends upon the cause of the interrupt.
- **CLASS III** – This class of interrupt occurs when a peripheral unit has completed an operation or when the peripheral unit cannot perform an operation (that is, no cards in punch unit.) The servicing also varies; if the unit cannot operate, the system is brought to a halt so that the operator can remedy the cause of trouble. If the unit is operating another order will be issued, if possible, to keep the UNIVAC 1050 peripheral units operating at maximum speeds.

A priority scheme is also associated with the classes of interrupt. Class I has the highest priority; Class III the lowest. That is, a Class III interrupt will not be allowed while a Class II interrupt is being serviced. However, the Class III interrupt signal is stored for servicing after the Class II interrupt has been serviced. A more detailed description appears in Section 7 of this reference manual.

4. APPLICATIONS FOR THE UNIVAC 1050 SYSTEM

4. APPLICATIONS FOR THE UNIVAC 1050 SYSTEM

Designed to operate as a satellite computer, the UNIVAC 1050 System is particularly well-suited to the tasks of auxiliary functions such as conversion of information from one medium to another—from punched cards to magnetic tape, from magnetic tape to punched cards, and from magnetic tape to printed hard copies. The system performs equally well, however, in other areas, as evidenced by the use of REGENT, a report program generator, to perform the translation of organizational reporting requirements into detailed machine-language programs.

Because of its modularity and compatibility, the UNIVAC 1050 System can vary in configuration, to meet the demands of the specific large scale system it is satellite to. This variation allows a presentation of only a few typical applications in this chapter.

CARD TO TAPE

The majority of electronic data processing systems today maintains master file records on magnetic tape and utilize punched cards for various input-output transactions. The large scale system's fastest input-output devices are magnetic tape units. If there is a card reader on a large scale system, it is feasible to use this unit to read in data transactions. However, when the volume of such transactions is high, the processing is generally input bound. In these instances it may be advantageous to eliminate the necessity for reading cards into the large scale computer. Instead card-to-tape operations would be relegated to "off-line" devices, and input to the large scale system accomplished by use of magnetic tapes.

The UNIVAC 1050 System is exceptionally well equipped to perform card to tape operations because of certain features. For example, the High-Speed Reader is available for either 80- or 90-column cards. Stub cards can also be read, 51 column size on 80-column cards and 29 or 58 columns on 90-column cards. The universal translate instruction code permits wide flexibility in assigning any special character sets that are required, such as the Army/Navy sorting sequence. If necessary, data can be edited into a different format. The combination of high tape density and variable tape record length (from 1 character to 4095 characters) provides for large scale tape record length requirements. It also aids in maximizing the number of cards that can be recorded on a single reel of tape.

There are both direct and indirect benefits that can be achieved by assigning card to tape operations to the UNIVAC 1050 System. The major advantages are:

- *Accomplished at a low cost*
- *Allows more efficient utilization of the large scale system (that is, tape-to-tape processing)*
- *Provides flexibility in equipment scheduling*
- *Makes systems design a simpler task*

Various factors enter into the selection of the tape record length for punched card data. Large scale systems normally are most efficient in processing longer tape records. In addition, the more punched card data there is per tape record, the fewer tape reels required. This reduces the number of manual operations in changing tape reels. The following diagram illustrates the punched card data capacity for the various tape reel sizes and densities.

UNISERVO III A Unit

Assumes a 1440-character tape record length, composed of either sixteen 90- or eighteen 80-column cards, at a density of 1000 ppi.

Reel Size	Total # of 90 Col.	Estim. CTT Time	Total # of 80 Col.	Estim. CTT Time
600 ft.	62,884	67 min.	70,792	75 min.
1200 ft.	125,888	134 min.	141,624	150 min.
3500 ft.	367,200	390 min.	413,100	436 min.

UNISERVO III C Unit

Assumes a 1440-character tape record length, composed of eighteen 80-column cards.

	Low Density 200 ppi		High Density 556 ppi	
Reel Size	Total Cards	Estim. CTT Time	Total Cards	Estim. CTT Time
2400 ft.	65,196	73 min.	154,800	172 min.

Figure 4-1 illustrates the application of the UNIVAC 1050 System for card-to-tape function only. The concurrent application of this function will be discussed in a subsequent section.

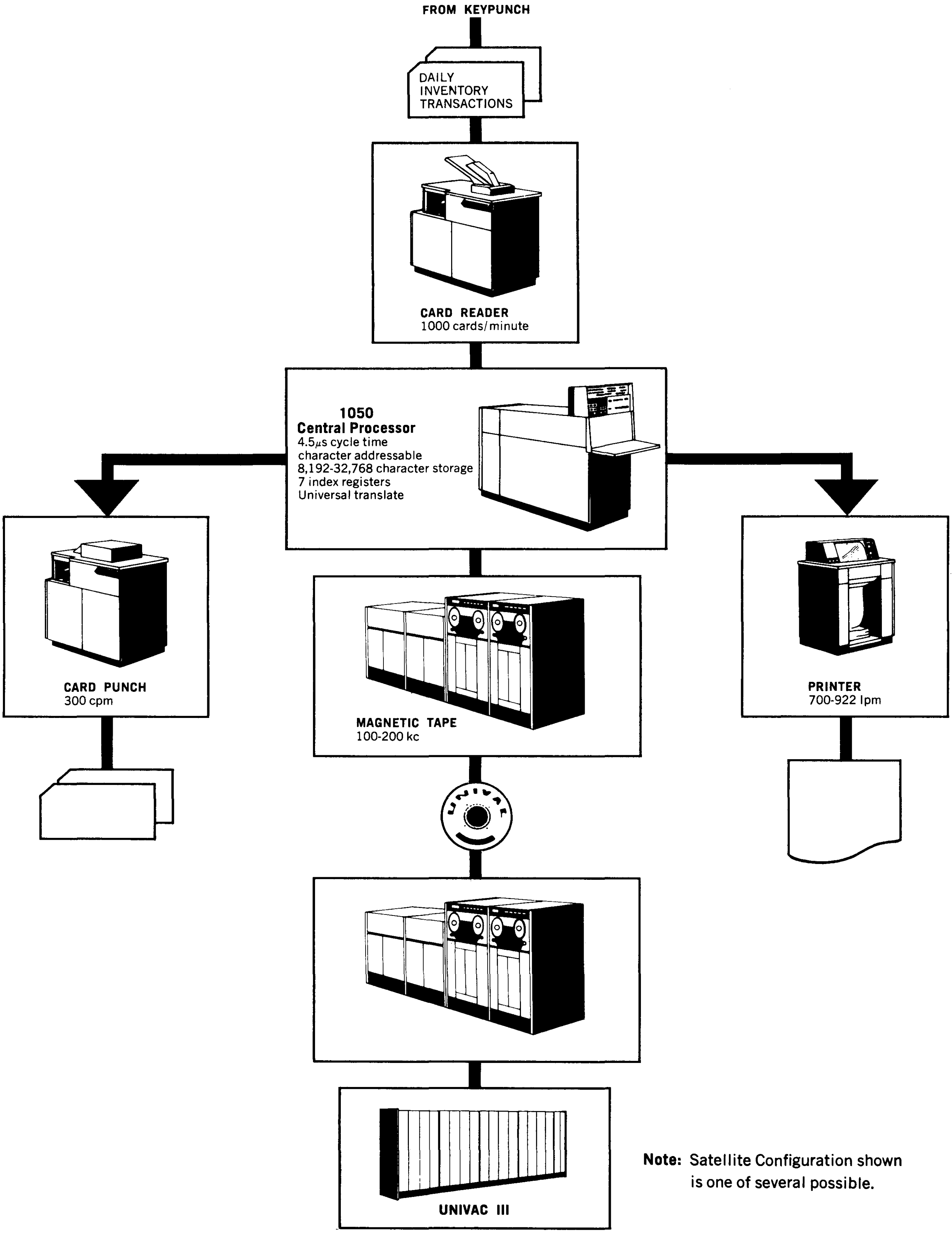
In the example shown, it is possible to perform a programmed input audit on the punched card data for sequence checking, zone punching, numeric field validation, and so on. The punch or printer can be used to signal invalid items that require off-line corrections. Similarly, control totals or listings can be produced. The output tape data can be edited as required.

TAPE TO PRINTER

A large scale system with high internal processing speeds can be limited on some applications by the output speed of its printer. Therefore, while data has been processed, computations and editing completed, the entire large scale system is marking time waiting for the completion of the previous printer operation. In such cases, it is recommended that the output be recorded on magnetic tape. By doing so the large scale system utilizes its fastest output device, which most closely approximates its internal processing speed. The advantages to be derived will be both a decreased large scale systems cost and an increased amount of large scale systems time available for other applications.

In terms of output speeds, the printer can produce printed reports at a rate of 922 lines per minute for alphanumeric data. The printer provides a 128-character print line. Where shorter reports are required, a 64-character print line can be utilized.

The forms design and format considerations are simplified since paper from 4″ to 22″ in width, containing up to 5 carbons can be accommodated.



Note: Satellite Configuration shown is one of several possible.

Figure 4-1. Card to Tape Conversions.

The print spacing is 10 characters per inch horizontally and either 6 or 8 lines per inch vertically.

Paper advance is program controlled allowing a wide degree of choice. The instruction repertoire of the UNIVAC 1050 System provides several instructions that simplify the programming of the appropriate controls. In addition there is a set of manual adjustments on the printer control panel to aid in forms alignment both initially and while the printer is in operation.

Magnetic tapes from large scale systems such as the UNIVAC III, UNIVAC 490, UNIVAC 1107 Systems can be accommodated through the UNISERVO III A Control Unit. Tapes compatible with the IBM 1410, 705, 7070, 7080, 7090 Systems can be accommodated through the III C Control Unit.

The III C Tape Control Unit provides automatic translation of tapes written in BCD mode. Tape length is 2400 feet. The previous approach of using either 132- or 792-character record length is followed. The tape reel capacity is shown in the chart at the bottom of the page.

Figure 4-2 illustrates the tape to printer operation of the UNIVAC 1050 System. Concurrent applications will be discussed later.

Tape speeds are more than sufficient to provide a steady flow of input data to match output requirements. The high density of magnetic tape plus variable record length format allows a high degree of flexibility in the preparation of data for the High-Speed Printer. The data may be edited either on the large scale system or the UNIVAC 1050 System.

The III A tape reels available are 600, 1800 and 3500 feet in length providing for light, medium or heavy volumes of data as required.

To illustrate the data capacity of magnetic tape reels, two approaches have been chosen. In the first a tape record is composed of either a single 128-character, or two 64-character print lines, plus controls. The total tape record length is 132 characters. In the second, the record is composed of six 128-character, or twelve 64-character print lines, plus controls. The total tape record length is 792 characters. The first is the simplest method of assigning one print line to one tape record. The second is a more sophisticated approach of assigning a number of print lines to a tape record. The second method increases tape reel capacity and reduces the amount of manual operator intervention for the replacement of tape reels.

UNISERVO IIIA TAPE CAPACITY

132ch. Record Length

Tape Length	No. of 128ch. Lines	Approx. Print Time*	No. of 64ch. Lines	Approx. Print Time
600 ft.	8480	10 min.	16,960	19 min.
1800 ft.	25,441	28 min.	50,882	56 min.
3500 ft.	50,883	56 min.	101,766	111 min.

792ch. Record Length

Tape Length	No. of 128ch. Lines	Approx. Print Time*	No. of 64ch. Lines	Approx. Print Time*
600 ft.	32,142	35 min.	64,284	70 min.
1800 ft.	96,426	105 min.	192,852	210 min.
3500 ft.	192,852	210 min.	385,704	419 min.

*Timing is based on continuous printing, single line spacing (922 lpm), rounded to whole minutes.

UNISERVO IIIC TAPE CAPACITY

132ch. Record Length

	No. of 128ch. Lines	Approx.* Print Time	No. of 64ch. Lines	Approx.* Print Time
Low Density	20,422	23 min.	40,844	45 min.
High Density	29,149	32 min.	58,298	64 min.

792ch. Record Length

	No. of 128ch. Lines	Approx.* Print Time	No. of 64ch. Lines	Approx.* Print Time
Low Density	36,684	40 min.	73,368	80 min.
High Density	79,410	87 min.	158,820	173 min.

*Timing is based on continuous printing, single line spacing, rounded to whole minutes.

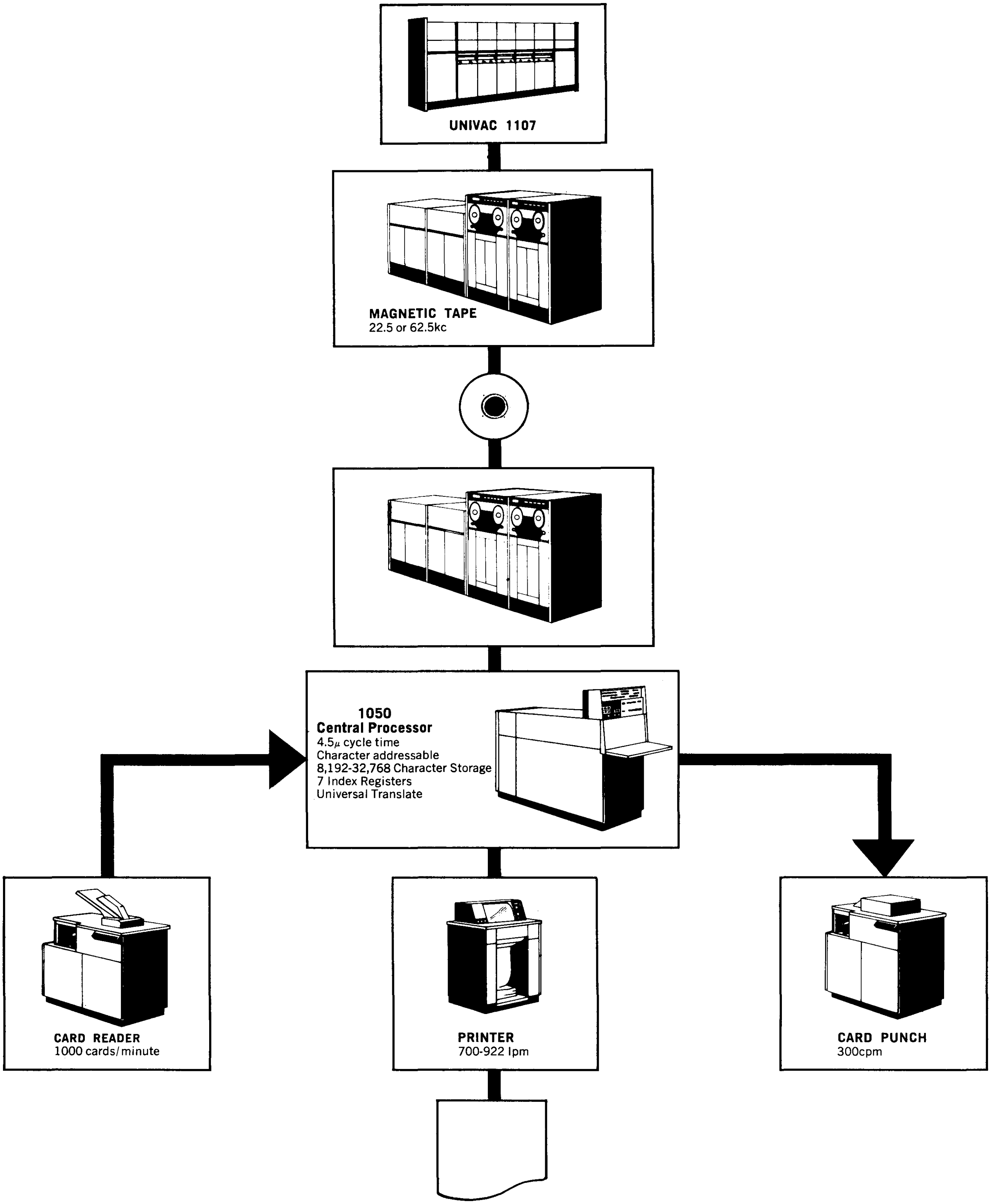


Figure 4-2. Tape to Printer Operation.

TAPE TO CARD

The maintenance of punched card files and the creation of punched card "turn around" documents or summary cards are necessary functions that can be performed as a satellite application. By writing this data on magnetic tape, the large scale system can perform this task in the shortest possible time. The tapes can then be processed by the UNIVAC 1050 System to produce the punched card output required. Because the burden of production is shifted to the satellite computer which is especially suited for such application, the large scale system is free for other activities.

The equipment characteristics of both the large scale and the UNIVAC 1050 Systems must be considered in determining the tape record length to be used. Large scale systems are most efficiently utilized when producing long tape records. However, the amount of storage available for tape input and the size of the program on the satellite tend to limit the size of a tape record.

Based on the above considerations, the following illustrates the effect of tape record length in relation to tape reel capacity and processing time per reel.

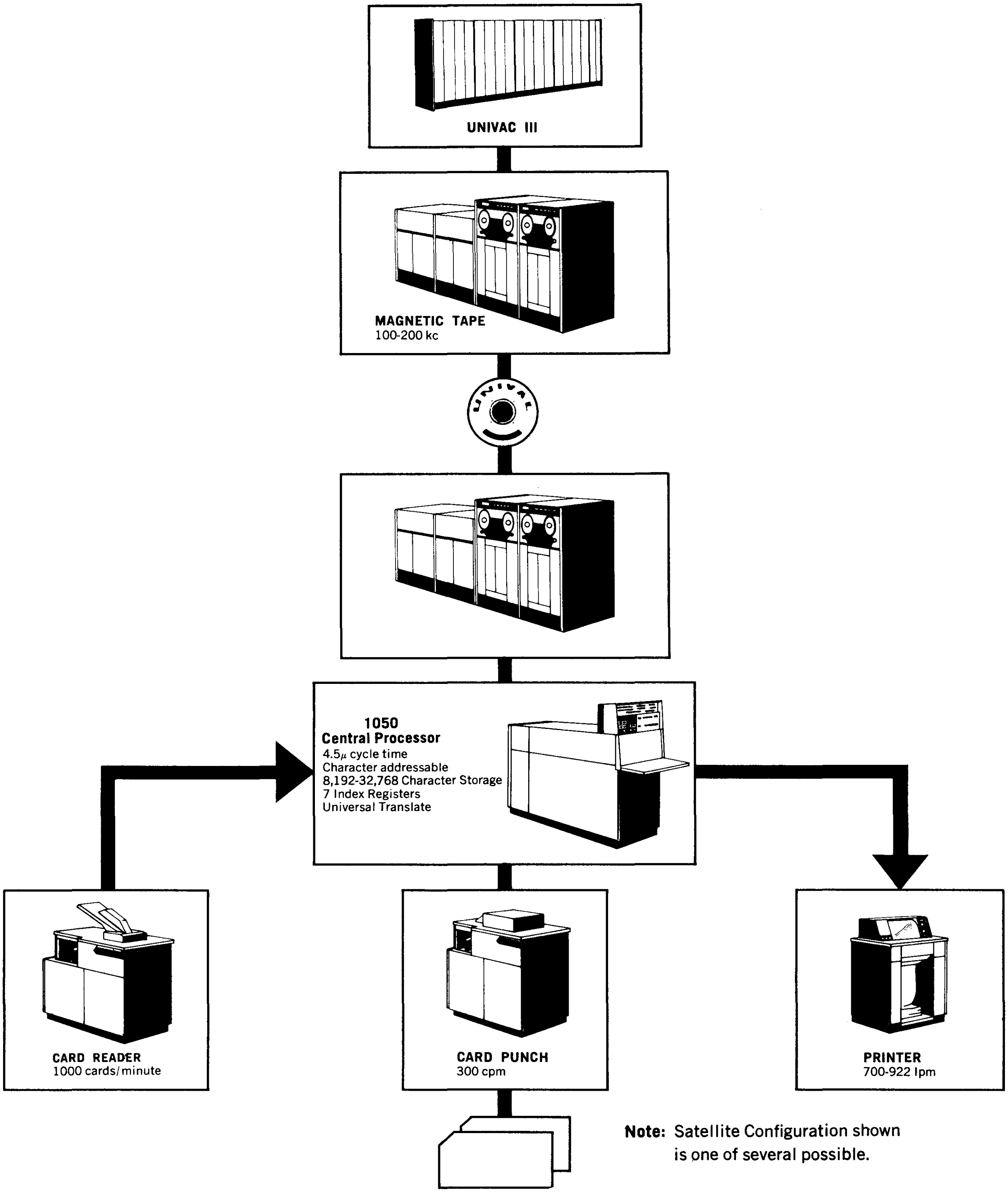
Figure 4-3 illustrates the tape to card application of the UNIVAC 1050 System. Concurrent applications will be discussed later.

UNISERVO III A TAPE UNIT (600 FT. REEL)

BLOCK SIZE, NO. OF CHARACTERS	320	360	720	1440
Number of 80-col. cards	29,088		50,229	70,792
Approx. Time, minutes	102		172	239
Number of 90-col. cards		28,232	44,648	62,882
Approx. Time, minutes		99	153	212

UNISERVO III C TAPE UNIT (2400 FT. REEL)

	320ch. No. Cards	Tape Record Approx. Time	720ch. No. Cards	Tape Record Approx. Time	1,440ch. No. Cards	Tape Record Approx. Time
Low Density (200 ppi)	49,020	172 min.	59,589	204 min.	65,196	220 min.
High Density (556 ppi)	86,876	304 min.	126,684	433 min.	154,800	522 min.



Note: Satellite Configuration shown is one of several possible.

Figure 4-3. Tape to Card Conversion.

CONCURRENT PROGRAM OPERATION

The outstanding features of the UNIVAC 1050 System are especially apparent during the running of concurrent programs. Two programs will not only share storage but will also overlap input-output operations. This effective utilization of storage is accomplished through the advanced design of the program interrupt concept. Each peripheral unit has a unique area in storage to which a program will proceed as soon as the input-output function has been completed. Therefore, as soon as an input-output unit accomplishes its function, the present program is interrupted temporarily and control is given to the specific unit's unique area. In this manner, input-output operations are overlapped so that optimum speeds may be achieved.

Tape reading or tape writing can proceed concurrently with printing since the printer buffer reduces the storage demands. Of course, the storage requirements of the High-Speed Reader and Card Punch Unit must be considered in weighing such factors as block size, storage available for input-output areas, and complexity of the program. In general, there is a direct relationship between the block size and the attainment of optimum card reader or punch speeds. This is illustrated in the following comparison which shows the effect of block size on reader speeds and the estimated time to perform card-to-tape and tape-to-printer operations, both individually and concurrently.

CARD-TO-TAPE			TAPE-TO-PRINTER			TOTAL SINGLE RUNS TIME*	CONCURRENT TIME*
NO. OF CARDS	CARDS PER BLOCK	TIME*	NO. OF LINES	LINES PER BLOCK	TIME*		
25,000	5	30	25,000	5	28	58	35
25,000	11	28	25,000	11	28	56	31
25,000	17	27	25,000	17	28	55	29

*NOTE: All times are in minutes.

5. PROGRAMMING THE UNIVAC 1050 SYSTEM

5. PROGRAMMING THE UNIVAC 1050 SYSTEM

Most programming for the UNIVAC 1050 System will be written in the language of the PAL Assembly System — an easy to learn, easy to use source language. The PAL assembler, with its related Input-Output Library and diagnostic services, is designed for convenience of writing and understanding programs for the UNIVAC 1050 System. Numerous checking facilities and diagnostic services are furnished to enable the user to work entirely in the language of PAL and its associated services. Any action based on attempts to employ forms of the instructions not described in this reference manual deviates from Univac recommendations and must be the user's responsibility.

A summary of the total software package, along with the system configurations required to utilize the software is shown in Figure 5-1. A detailed description of the software routines appears later in this section.

PAL ASSEMBLY SYSTEM

A source language statement may generate a single machine language instruction; generate several machine language instructions; or direct the assembler. Statements which generate a single machine instruction are called symbolic machine instructions. All others are called symbolic non-machine instructions. PAL incorporates a minimum set of assembler directives and rules and allows the programmer to refer to fields in storage with a single expression. In addition to the assembly system described in the following sections, there is a card version of PAL. This does not provide I/O specialization as part of the assembly — a special pass is required. The card assembler also does not provide input/output macro instructions, or the assembly 'directives PROC, NAME, or DO.

Operator System

These routines allow the operator to load and start programs which are to operate independently or concurrently. These routines assure effective utilization of the various input-output channels for two programs running concurrently. A detailed description of the Operator System will be published separately.

Input-Output Library

The library includes a complete set of input-output routines which allows the user to call on (incorporate at assembly time) the particular specialized routines which are required by the program.

Configuration No.		Tapes	Reader	Punch	Printer	Card Assembler	Tape Assembler	Operator	Input-Output Library	Source Code Librarian	Patch Assembler	Regent
0	8 K to 32 K in steps of 4 K	1	0	0	1			a				
1		1	0	1	1			a				
2		1	1	0	1			■			c	
3		1	1	1	0	NO LISTING		■	■	NO LISTING	NO LISTING	
4		1	1	1	1	■		■	■	■	■	■
5		2	0	0	1			a				
6		2	0	1	1			a				
7		2	1	0	1	b		■	b	■	c	b
8		2	1	1	0	NO LISTING		■		NO LISTING	NO LISTING	d
9		2	1	1	1	■	■	■	■	■	■	■

a – Operator system will operate assuming it is loaded from tape. Load routine will be changed to operate from tape.

b – Output can be placed on tape.

c – Patch assembler not prepared to update a tape.

d – Regent can operate – no listing of program or of program errors. Regent is primarily for card-or-tape to printer operations.

■ – Available for this configuration.

Figure 5-1. Standard 1050 Configurations.

Patch Assembler

This routine allows the user to make modifications to an object program without reassembly. The Patch Assembler accepts corrections in PAL source language and produces an output which supplements and amends the original object code produced by the PAL Assembly System. These same corrections can be used later to update the original source code for a full reassembly.

Report Program Generator (REGENT)

This routine accepts input in the form of specification cards which describe the input format and the desired report output format, and produces an object program ready to run.

Source Code Librarian

This routine facilitates the maintenance of a file of source code programs. The Source Code Librarian allows for changes, insertions and deletions.

A detailed description of the Source Code Librarian will be published separately.

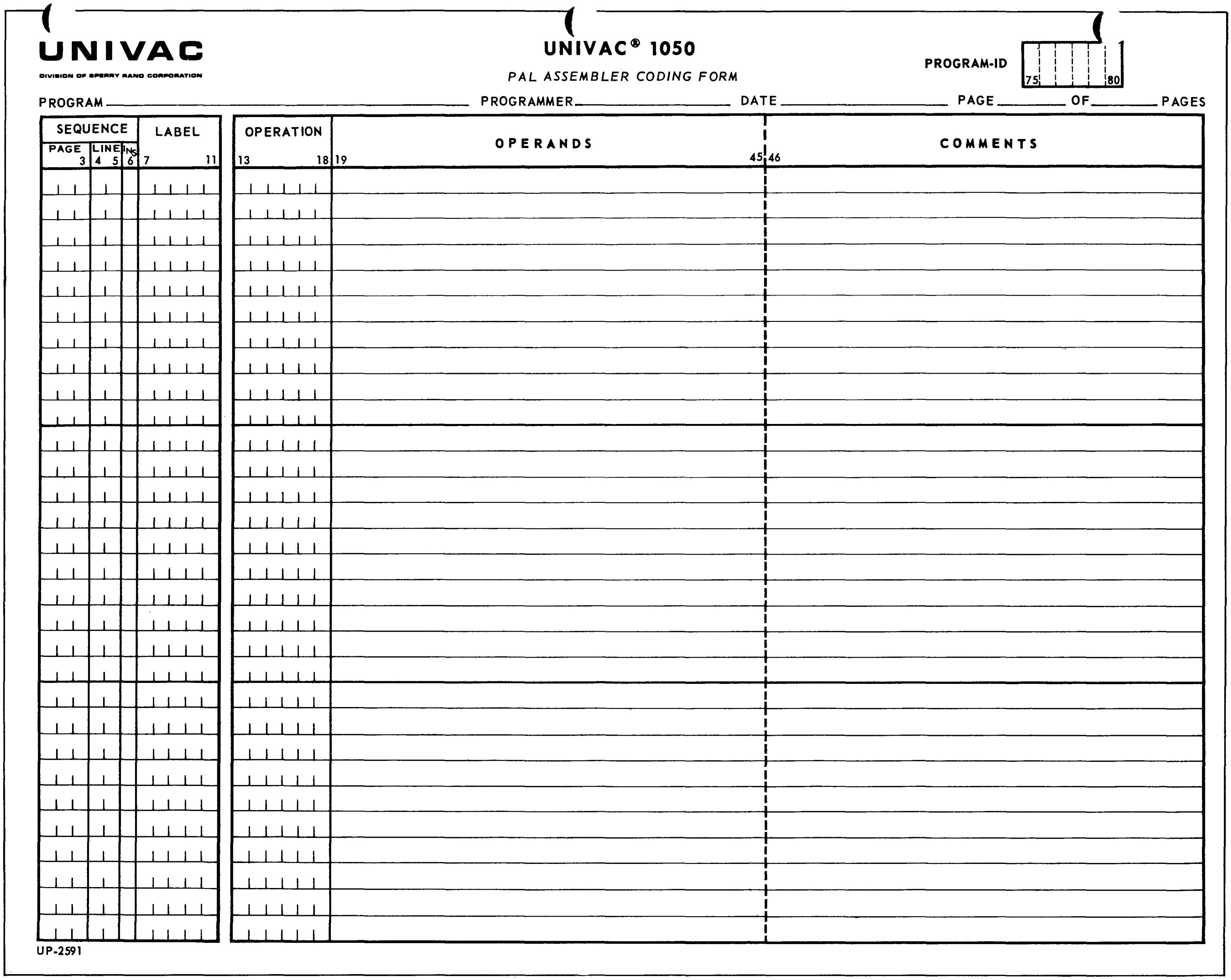
SYMBOLIC CODING FORMAT

Programs are written on the standard coding form shown in Figure 5-2. In the description of this form, which follows, certain terms are used with specific definitions:

- *Alphabetic character means a character of the English alphabetic set A through Z.*
- *Numeric character means a character of the Arabic numeral set 0 through 9.*
- *Alphanumeric character means a character of either of the above sets (A through Z, 0 through 9).*

The symbolic coding format is composed of fixed format fields for program identification, page, line, insert, label, operation, and variable format fields for operands and comments.

It will be noted that numbers are associated with each subdivision of the coding form. These indicate the card columns into which the characters written by the programmer are to be punched. These column numbers hold true for both 80- and 90-column cards. The 80-column source card is shown in Figure 5-3; the 90-column source card, in Figure 5-4.



UNIVAC

DIVISION OF SPERRY RAND CORPORATION

UNIVAC® 1050

PAL ASSEMBLER CODING FORM

PROGRAM-ID 75 80

PROGRAM ________ PROGRAMMER ________ DATE ________ PAGE ____ OF ____ PAGES

SEQUENCE			LABEL	OPERATION	OPERANDS	COMMENTS
PAGE	LINE	INS				
3	4 5	6	7 11	13 18	19 45	46

UP-2591

Figure 5-2. PAL Coding Form.

Program-Id Field

N	E	T	P	A	Y
75					80

The program identification, in effect the name of the program, is composed of one to six alphanumeric characters. The first character is left justified; that is, character position 75 is always used.

Sequence

PAGE, LINE, AND INSERT FIELDS

The page field entry is three numeric characters and is regarded as part of a five character field consisting of the page and line fields.

The line field entry consists of two numeric characters. Each line entry must be sequentially higher than any preceding line with the same page entry. The insert field entry consists of one numeric character. This field is used when a line of coding is to be inserted on a particular page following a particular line. To insert a line of coding between lines 23 and 24 of page 10, the coding used could be:

SEQUENCE		
PAGE 3	LINE 4 5	INS 6
0 1 0	2 3	7

The only restriction on the character used for INS is that, if more than one instruction is to follow a particular page line, each insertion line must have a sequentially higher INS number than any preceding it.

Note: The card punched for an INS line must be physically inserted in its proper place in the program deck punched from the rest of the PAL coding. A card produced from the previous example would have to be inserted between the cards for lines 23 and 24 of page 10.

Label Field

If a line is to have a label, it is written in the LABEL field. A label is one kind of symbol. A symbol is composed of one to five alphanumeric characters, the first of which is an alphabetic character. The first character of a LABEL field entry must be left justified. Except in the case of area statements, the LABEL is assigned an address equal to the address of the least significant character of the line of information with which it is associated.

INS 6	LABEL 7 11
	S T A R T
	P T G A

Operation Field

The OPERATION field is a six character field and may contain an assembler directive, a mnemonic machine instruction code, or a data generating code. An entry in this field must be left justified.

OPERATION 13 18
B D 2
P R O C

Operands Field

The OPERANDS field is composed of a string of expressions separated by commas and is terminated by the first blank following a nonblank other than comma. Column 72 of the line also terminates the OPERANDS field. Any expression is terminated by a blank or comma; however, it can have any number of preceding blanks. If an expression is terminated by a comma, this indicates that another expression follows. The maximum number of expressions on a line and the interpretation of each expression is determined by the contents of the OPERATION field; however, any line may have less than the maximum number of expressions. A line can be continued on a second line by writing a minus (–) in the most significant character position of the OPERATION field of that line. Only one continuation line is permitted.

The form of the OPERANDS field for each symbolic machine instruction is shown in **Figure 5-5.**

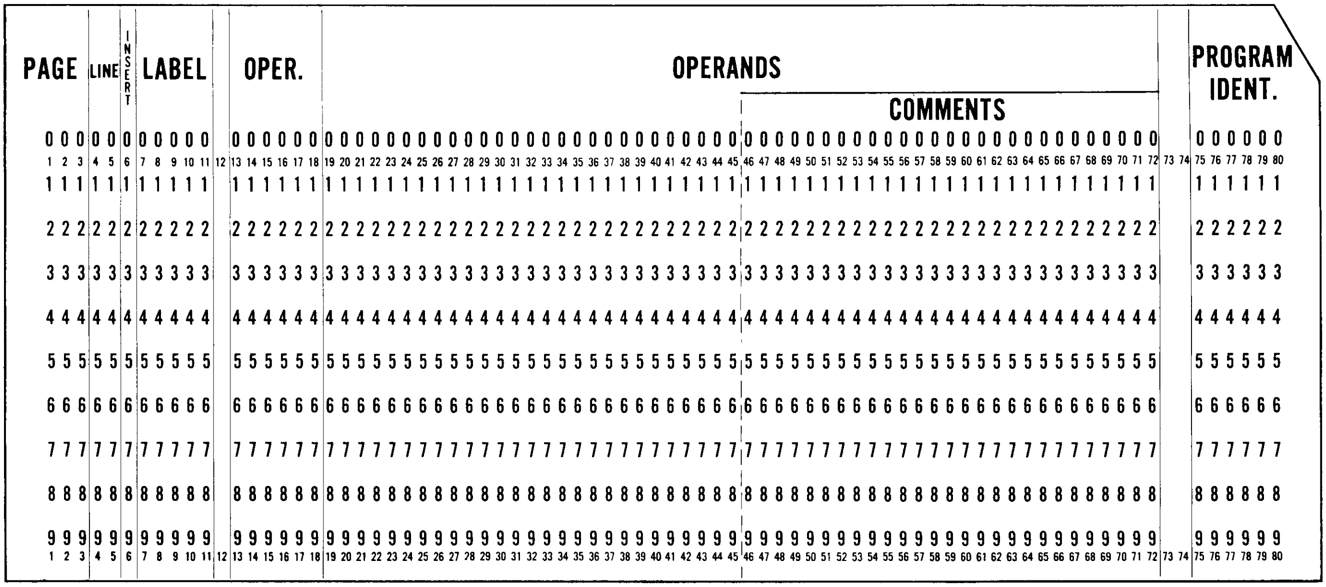


Figure 5-3. PAL 80-Column Source Card.

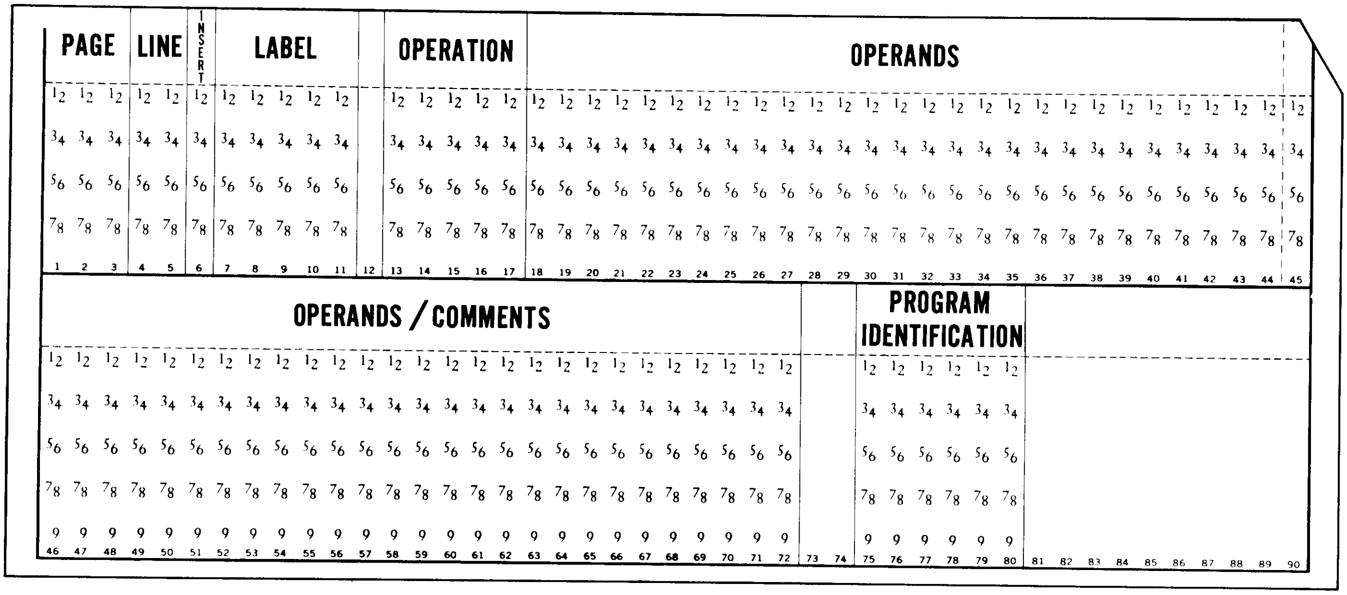


Figure 5-4. PAL 90-Column Source Card.

Not all expressions need to be written in some cases. For example, a symbol written as the M expression on an instruction line may also define the length of the field addressed. In this case, the L portion of the instruction line may be omitted.

Some possible forms for the OPERANDS field are:

OPERATION 13 18	OPERANDS 19
	M, L, X
	M, , X
	M, L
	M
	, L
	M, S, X
	M, S
	M, I, X
	M, I
	M,N, X
	M, N
	M, C
	M, T
	F, D, U, X
	F, , , X

Note that if the last expression which might appear on a line is omitted, the comma which would have preceded it is omitted.

Additional Rules

The remaining rules for the use of the PAL coding form are:

1. COMMENTS is shown for character positions 46 through 72. This is an artificial division recommended for standard positioning of comments on the printed listing. COMMENTS consists of notes or remarks concerning the program to be processed by the PAL Assembly System. These notes do not affect the processing of a line and are not reflected in object coding. COMMENTS can appear on any line following the last expression on the line and separated from it by a blank. (The last expression on a line is terminated by a blank. Every line except a COMMENTS line must have at least one expression.)
2. COMMENTS in addition to those made on normal lines of coding can be introduced at any point in the symbolic coding by use of a COMMENTS line. The COMMENTS line consists of a period in the leftmost character position of the label field followed by a blank followed by the comments. A COMMENTS line produces no output coding, but does produce a printed line on the symbolic listing.
3. If an alphabetic character appears in the leftmost position of the LABEL field, the field is assumed to contain a label of up to five characters.
4. If there is no period or alphabetic character in the leftmost position of the LABEL field, the field is assumed to be blank.
5. Character position 12 is normally left blank.

SYMBOLIC INSTRUCTIONS

In order to use symbolic instructions correctly and efficiently, the programmer must have a basic understanding of how instructions and data are stored and addressed, even though he will not be required to specify these details in PAL instructions. Bit positions referred to in the following discussion are numbered from right to left; the least significant bit is number 0.

REPRESENTATION OF INFORMATION IN CORE STORAGE

Data which is to be processed by the UNIVAC 1050 System is read from an external medium such as punched cards or magnetic tape, and placed in coded form in the core storage of the Processor. The data to be stored can consist of alphanumeric or special characters.

Numbers are represented in storage by two different methods. Certain values, such as an address or index register designation, are stored in a true binary format. Numerical data is usually stored in a character format, as are alphabetic and special characters. Each character in its coded form occupies six bit positions. A seventh bit position exists for a check or parity bit. Every time a character containing an even number of one bits is read into the system, a check (parity) bit is produced automatically and stored with the character. No parity bit is produced for a character containing an odd number of one bits. Thus, a character will always consist of an odd number of one bits stored in seven bit positions.

Since the user is never directly concerned with the parity bit (it is not accessible to the programmer), it will not be discussed further. When reference is made to a 6-bit character, it will be understood that a parity bit may also be present. Table 5-1 gives the complete character set, showing the equivalent 6-bit binary coding for each of the external media, together with the collating (ordering) sequence.

The 6–bit character format is further divided into two portions; the zone portion (bits 5-4) and the numeric portion (bits 3-0). All positive numeric data stored in the character format has zero zone bits. A field is negative when its least significant character has a 1-bit in position 5. Thus,

+4 is represented as 000111 and
−4 is represented as 100111

Data Fields

The standard unit of storage is a single 6-bit character which occupies one position of storage. Each field of data, whether a 1-character code or a 30-character descriptive name, occupies exactly the number of storage positions it requires. This organization permits the greatest economy in the assignment of storage to fit data fields of different lengths. In addition to space economy, the execution time of an instruction is directly related to the length of its associated OPERANDS field.

Representations of Instructions in Core Storage

Instructions, as well as data, occupy core storage. Each instruction occupies five consecutive character positions, or 30 bits, and successive instructions are placed one after the other, starting in any arbitrarily selected character position. The instruction control counter (CC) keeps track of the address of the most significant character of each instruction, and as each instruction is completed, the counter is incremented by 5. The control counter will step through storage in steps of 5 as the instructions are executed until some action occurs which overrides this automatic sequencing and sets the control counter to some other address.

Every instruction is divided into a number of fields. The general format is as follows (see I/O for exceptions.)

OP CODE (Bits 29-25) indicates the form of the operation to be performed.

X (Bits 24-22) indicates the index register to be used, if any, for address modification.

(Bit 21 disregarded.)

M (Bits 20-6) represents the address of the least significant digit of the field addressed, except for the block transfer, zero suppress and sequence control instructions; also multiply, divide and I/O instruction, which do not address a field.

C (Bits 5-0) have a variety of uses depending upon the operation to be performed; details are given with each instruction description.

ADDRESSING

Core Storage

Core storage is organized into modules, each capable of storing 4096 characters. A minimum of two (8192 characters) modules and a maximum of eight (32,768 characters) modules are permissible. Regardless of the size of storage the low order position always has an address of 0, the next position 1, and so forth. Memory can be considered as a contiguous sequence of position. The last (high order) position has an address one less than the total number of positions. Thus, in a system with 16,384 positions of storage, the last position has an address of 16,383.

The minimum number of binary bits required to specify 32,768 positions is 15; hence the 15-bit address in the instruction format. For ease in presentation, octal numbering is used to represent machine address; Appendix A presents a Decimal to octal conversion table. If the storage contains fewer than 32,768 positions, an instruction which addresses an area outside the actual limits of core storage will result in

- A parity error, if the instruction calls for reading information from the addressed area.
- No action, for portions of an instruction execution cycle which attempt to write in such an area.

Normally the user need not be concerned with position addressing. A LABEL can be given to a field and this LABEL equated to a particular character position and field length. Each reference to this field can be the LABEL assigned, thus relieving the programmer of the task of writing the particular character position and field length.

CARD CODES		BINARY CODE (Machine Collating Sequence)	HIGH-SPEED PRINTER CHARACTER		OCTAL	NUMBER
80 COLUMN	90 COLUMN		STANDARD	OPTIONAL		
NO PUNCH	NO PUNCH	000000	Space (Non-Printing)		00	0
11-5-8	1-3-5-7	000001	]		01	1
11	0-3-5-7	000010	– (minus or hyphen)		02	2
0	0	000011	0		03	3
1	1	000100	1		04	4
2	1-9	000101	2		05	5
3	3	000110	3		06	6
4	3-9	000111	4		07	7
5	5	001000	5		10	8
6	5-9	001001	6		11	9
7	7	001010	7		12	10
8	7-9	001011	8		13	11
9	9	001100	9		14	12
0-6-8	0-1-3-7-9	001101	\		15	13
11-6-8	1-3-5-7-9	001110	;		16	14
12-5-8	0-5-7-9	001111	[		17	15
12	0-1-3-5-7	010000	+	&	20	16
5-8	1-3-7-9	010001	: (colon)		21	17
12-3-8	1-3-5-9	010010	. (period)		22	18
12-0	0-1-3	010011	?		23	19
12-1	1-5-9	010100	A		24	20
12-2	1-5	010101	B		25	21
12-3	0-7	010110	C		26	22
12-4	0-3-5	010111	D		27	23
12-5	0-3	011000	E		30	24
12-6	1-7-9	011001	F		31	25
12-7	5-7	011010	G		32	26
12-8	3-7	011011	H		33	27
12-9	3-5	011100	I		34	28
3-8	0-1-5-7	011101	=	#	35	29
12-6-8	0-1-5-9	011110	<		36	30
12-7-8	0-1-3-5-7-9	011111	#		37	31
7-8	0-1-5-7-9	100000	@	' (apostrophe)	40	32
11-4-8	0-1	100001	*		41	33
11-3-8	0-1-3-5-9	100010	$		42	34
11-0	0-3-7-9	100011	!		43	35
11-1	1-3-5	100100	J		44	36
11-2	3-5-9	100101	K		45	37
11-3	0-9	100110	L		46	38
11-4	0-5	100111	M		47	39
11-5	0-5-9	101000	N		50	40
11-6	1-3	101001	O		51	41
11-7	1-3-7	101010	P		52	42
11-8	3-5-7	101011	Q		53	43
11-9	1-7	101100	R		54	44
0-5-8	0-1-9	101101	%	(	55	45
4-8	0-1-3-7	101110	' (apostrophe)	@	56	46
11-7-8	0-1-7	101111	△		57	47
0-2-8	0-1-7-9	110000	≠		60	48
0-4-8	0-1-5	110001	(	%	61	49
0-3-8	0-3-5-9	110010	, (comma)		62	50
2-8	1-5-7-9	110011	&	+	63	51
0-1	3-5-7-9	110100	/		64	52
0-2	1-5-7	110101	S		65	53
0-3	3-7-9	110110	T		66	54
0-4	0-5-7	110111	U		67	55
0-5	0-3-9	111000	V		70	56
0-6	0-3-7	111001	W		71	57
0-7	0-7-9	111010	X		72	58
0-8	1-3-9	111011	Y		73	59
0-9	5-7-9	111100	Z		74	60
12-4-8	0-1-3-9	111101	)	⌑	75	61
6-8	0-3-5-7-9	111110	>		76	62
0-7-8	0-1-3-5	111111	⌑	)	77	63

***NOTE:** Only the characters that differ from the standard are listed for the optional print drum.

TABLE 5–1. UNIVAC 1050 System Character Set

TETRADS

The first (low order) 256 positions of storage can be considered as 4-character position groups. These 4-position groups are called Tetrads, many of which have special functions. The special functions are discussed in the detailed instruction description.

Tetrads are numbered 0 through 63. In Tetrad instructions, the T expression contains the address of the tetrad involved. In general the T expression will be a LABEL, which must be equated to the address of the Tetrad involved. Similarly the X expression contains an octal or decimal number representing the index address of the index register involved. Generally the X expression will contain a LABEL, which must be equated to the storage address of the index register involved.

A list of suggested LABELS for the commonly used Tetrads is presented in table 5-2. The definitions of these labels are available in the form of prepunched cards which can be included with the input to the PAL Assembler.

INDICATORS

Many testable indicators exist in the UNIVAC 1050 System. These indicators are addressed as decimal numbers 0 through 63 (octal 0 through 077). The testing of these indicators is discussed in detail with the sequence control instructions; the setting of these indicators is discussed with the instructions which will affect the indicators. Normally the indicators will be addressed using a label, which is equated to the indicator number. Table 5-2 also lists the more commonly used indicators and their suggested labels.

LABEL	OPERATION	OPERAND		
		OCTAL	DECIMAL	
AR1	EQU	017	15	Arithmetic Register 1
AR2	EQU	037	31	Arithmetic Register 2
X1	EQU	047	39	indeX Register 1
X2	EQU	053	43	indeX Register 2
X3	EQU	057	47	indeX Register 3
X4	EQU	063	51	indeX Register 4
X5	EQU	067	55	indeX Register 5
X6	EQU	073	59	indeX Register 6
X7	EQU	077	63	indeX Register 7
DST	EQU	0103	67	DeStination address for Transfer from (TFR, TFI)
ORG	EQU	0107	71	ORiGin address from Transfer To (TTR, TTI)
TRO	EQU	0110	72	Translate table ROw address
ZCT	EQU	0111	73	Number of characters suppressed
TCT	EQU	0113	75	Number of characters to be Transferred
MLR	EQU	0127	87	MuLtiplieR
QTN	EQU	0127	87	QuoTieNt
INDICATORS — NOT IN CORE STORAGE				
KNO	EQU	040	32	No operation
KHI	EQU	041	33	High indicator
KEQ	EQU	042	34	Equal indicator
KUQ	EQU	043	35	Unequal indicator
KLO	EQU	044	36	Low indicator
KZR	EQU	045	37	Indicator of arithmetic result zero
KM	EQU	046	38	Indicator of decimal arithmetic result minus
KNB	EQU	047	39	Indicator of no overflow in last binary arithmetic
KDF	EQU	050	40	Decimal overflow indicator

TABLE 5-2. SUGGESTED STANDARD EQUALITY STATEMENTS

INDEXING

There are seven index registers in the UNIVAC 1050 System. These index registers are numbered 1 through 7, and occupy Tetrads 9 through 15 respectively. Index registers allow the modification of instruction storage addresses without changing the instruction in storage. There are no signs in the index register so that all indexing is by a positive increment. Thus, an instruction addressing a field labeled DATA as modified by the index register containing a 20 would be executed as if it were written DATA +20. The instruction in storage addresses the field DATA. Index registers may be incremented thus allowing a single instruction to operate on many fields, such as DATA, DATA +20, DATA +40 and so forth.

SYMBOLIC ADDRESSING

The following rules apply to the definition of values for labels and the relationship of these values to the final machine instructions produced:

1. A label appearing on a line containing a mne-' monic operation, constant, or field definition is assigned a value equal to the address of the least significant character of that line.
2. For the sake of consistency, the operand address of Sequence Control instructions will be automatically modified during assembly. Thus, although these instructions do not actually address the least significant character of the next instruction, they can be written as though they did.
3. A symbol appearing in the LABEL field of a line containing an AREA directive is assigned a value equal to the address of the most significant character of the area. This will reduce the number of times that this label has to be used with a modifier.

Expressions

An expression can consist of one of the items shown in the chart at the bottom of the page.

An expression can have a leading + or – sign, to denote a positive or a negative quantity. The value of an expression is a binary integer. Any negative integer value of an expression is represented by a two's complement and thus is always a positive value. However, if the operation field is –n (see *Data Generation*), and the expression is alphabetic or alphanumeric, the sign bit of the expression value is reversed.

An expression can also consist of two or three of the above items connected by:

+ (meaning an arithmetic summation is to be performed).

– (meaning an arithmetic difference is to be derived).

TYPE	ABBREVIATION	FORM	VALUE	EXAMPLE
Symbol	S	one to five alphanumeric characters beginning with an alphabetic character.	value assigned to the symbol as a result of an EQU directive or of appearance in the LABEL field.	L TAP02 COST
Location	L	$	current value of location counter, namely the address of the least significant character of the line in which the item $ appears.	$
Octal	O	Zero followed by octal (0-7) digits.	value interpreted as base 8 and converted to binary.	017 has the value 001111
Decimal to Binary	D	non-zero digit followed by decimal (0-9) digits.	value interpreted as base 10 and converted to binary.	17 has the value 010001
Alphanumeric	A	' followed by any characters except ' followed by '.	value of each character in corresponding position right justified (6-bit representation).	'ABC' has the value, 010100 010101 010110 '17' has the value, 000100 001010

The following are the meaningful groups of items which can be connected, taken in any order. Any other groups are unacceptable to the PAL assembler.

SS
SL
SO
SD
LO
LD
SSS
SSL
SSO
SSD
SLO
SLD

Some examples of meaningful expressions:

OPERATION 13 18	19 OPERANDS
	JOE − SAM + 1
	$ + 023
	SAM + JOE − MAC
	$ + JOE − 7
	PAT + 077

1050 INSTRUCTION REPERTOIRE

To explain the operation of each instruction, and its possible variations, the 1050 instruction repertoire has been divided into functional classes as follows:

Data Transfer Instructions

Arithmetic Instructions

Comparison Instructions

Sequence Control Instructions

Conversion and Edit Instructions

Block Transfer Instructions

Logical Instructions

Input-Output Instructions

As a further visual aid the chart reproduced for ready cross reference on each pair of facing pages of this section shows the requirements of each instruction in a condensed form.

In this chart the Operation Code and all other numerical values and ranges are given in octal notation. Operation codes have been ordered numerically (Table 5-3) and alphabetically (Table 5-4).

The bit configuration of an operation code consists of five bits, but a sixth bit is implied to convert the code into an octal expression for readability on the assembler listing. In the UNIVAC 1050 System, the sixth bit is implied in the least significant (rightmost) position of the operation code. Thus, a bit structure of

10111

is referred to octally as 56(10111$\underline{0}$)

Abbreviations used in the chart, and in the detailed explanation of each instruction, are:

SYMBOLIC CODING	MACHINE INSTRUCTION FORMAT	EXPLANATION
M	M	Memory Address, bits 20-6
L	L	Length, bits 3-0 (5-0 for TR)
X	X	Index Register or Channel designation, bits 24-22
C	C	Character or Constant, bits 5-0
T	T	Tetrad designation, bits 5-0
I	I	Indicator designation, bits 5-0
S	S	Shift count, bits 2-0
N	N	Loop counter, bits 5-0
a		Arithmetic Register designation, bit 4
n		Number of Characters involved in shift, bits 4-3
	OP	Operation Code, bits 29-25
	Bit 5	Operation Code expansion, bit 5
U	U	Unit designation (in External Function instruction), bits 21-18
F	F	Function designation (in External Function instruction), bits 17-12
D	D	Detail designation (in External Function instruction), bits 11-0

NOTE: The mnemonic operation code represents a combination of the OP and bit 5. The n and supplemental codes represent bits 4-3 or bit 4 respectively.

To illustrate the use of these abbreviations, the Bring Decimal to Arithmetic Register instruction is represented as:

OPERATION 13 18	19 OPERANDS
B D a	M, L, X

OCTAL OP CODE	MNEMONIC	DESCRIPTION
00	–	(Unassigned)
02	–	"
04	–	"
06	–	"
10	JR	Jump Return
12	TR	TRanslate
14	LC	Logical Comparison
16	BCn	Bit Circulate
16	BSn	Binary Shift
20	FT	Fix Tetrad
22	ZS*	ZeroSuppress with asterisk fill
22	ZS$	Zero Suppress with floating dollar sign
22	ZS	Zero Suppress with no floating dollar sign
24	TFI	Transfer From memory, Increment destination address
24	TFR	Transfer From memory, Reset destination address
24	TTI	Transfer To memory, Increment origin address
24	TTR	Transfer To memory, Reset origin address
26	PD	PaD blanks
26	PD0	PaD decimal zeros
26	CDa	Compare Decimal
30	JC	Jump Conditionally
32	JL	Jump Loop
34	CC	Compare Character
36	–	(Unassigned)
40	XF	eXternal Function
42	ST	Store Tetrad
44	SC	Store Character
46	BT	Bring to Tetrad
50	DV	DiVide
50	MPC	MultiPly Cumulative
50	MPN	MultiPly Noncumulative
52	ED	EDit
52	SAa	Store Arithmetic register
52	SAR	Store both Arithmetic Registers
54	LP	Logical Product
56	BAa	Bring Alphanumeric
56	BDa	Bring Decimal
60	AC	Add Character
62	SMa	Subtract from Memory
64	LS	Logical Sum
66	ADa	Add Decimal
66	SDa	Subtract Decimal
70	CBa	Compare Binary
72	ABa	Add Binary
74	CT	Compare Tetrad
76	AT	Add to Tetrad

TABLE 5-3. MNEMONIC OPERATIONS ORDERED BY OPERATION CODE

MNEMONIC INSTRUCTION	OCTAL OPCODE	DESCRIPTION
ABa	72	Add Binary
AC	60	Add Character
ADa	66	Add Decimal
AMa	62	Add to Memory
AT	76	Add to Tetrad
BAa	56	Bring Alphanumeric
BCn	16	Bit Circulate
BDa	56	Bring Decimal
BSn	16	Binary Shift
BT	46	Bring to Tetrad
CBa	70	Compare Binary
CC	34	Compare Character
CDa	26	Compare Decimal
CT	74	Compare Tetrad
DV	50	DiVide
ED	52	EDit
FT	20	Fix Tetrad
JC	30	Jump Conditionally
JL	32	Jump Loop
JR	10	Jump Return
LC	14	Logical Comparison
LP	54	Logical Product
LS	64	Logical Sum
MPC	50	MultiPly Cumulative
MPN	50	MultiPly Noncumulative
PD	26	PaD blanks
PD0	26	PaD decimal zeros
SAa	52	Store Arithmetic register
SAR	52	Store both Arithmetic Registers
SBa	72	Subtract Binary
SC	44	Store Character
SDa	66	Subtract Decimal
SMa	62	Subtract from Memory
ST	42	Store Tetrad
TFI	24	Transfer From memory, Increment destination address
TFR	24	Transfer From memory, Reset destination address
TR	12	TRanslate
TTI	24	Transfer To memory, Increment origin address
TTR	24	Transfer To memory, Reset origin address
ZS*	22	Zero Suppress with asterisk fill
ZS$	22	Zero Suppress with floating dollar sign
ZS	22	Zero Suppress with no floating dollar sign
XF	40	eXternal Function

TABLE 5-4. MNEMONIC OPERATIONS ORDERED ALPHABETICALLY

The symbol a is coded as 1 or 2 to specify one of the two Arithmetic Registers; in the machine instruction format this is represented by the absence or presence of a one bit in position 4.

Timing is shown in decimal microseconds with no indexing specified; indexing adds 13.5 microseconds. Additional abbreviations used in timing formulas are:

L_m L or the length of the field in ARa, whichever is greater

L_c The number of character positions that a carry is propagated

B The number of characters transferred (Tetrad 18)

K Length of multiplicand or divisor

E The number of characters inserted by the Edit Instruction or suppressed by the ZS Instructions

In the symbolic coding examples, the symbols used generally represent the address of the least significant character of the named field; other items in the operands field are decimal numbers, octal numbers, or alphanumeric characters. The contents of storage and arithmetic registers are shown before and after the execution of the instruction. The vertical arrow indicates the least significant character of the field. The field is underlined to set it off from unaffected character positions, which are indicated by lower case x.

The abbreviation M_x is used, in describing the operation of the various instructions, to indicate the *effective* character or field position in memory—that is, the address assigned to M as further modified by the contents of Index Register X (if called for).

Arithmetic Registers are addressable as part of storage. Therefore, when the text states that the Arithmetic Registers are not changed, this is the case, except when the AR's are addressed as a result field by M_x or T.

The PAL Assembly System will automatically convert the written elements of an instruction to the internal form required for execution. Normally the programmer need not be concerned with these transformations. Table 5-5 lists the acceptable forms for writing portions of PAL instructions, together with the internal forms produced in assembly.

ITEM	ACCEPTABLE INPUT	TRANSLATED TO (OCTAL)
M	0-32767	0-77777
	0-077777	0-77777
	Label or Expression	Value Assigned
X	X1 through X7	1-7
	1 through 7	1-7
	0	0
	(blank)	0
ARa	AR1	17
	AR2	37
a	1	0
	2	1
n(BCn,BSn)	1	1
	2	2
	3	3
	4	0
S(BCn,BSn)	0-7	0-7
L(MP, DV)	1-7	1-7
	8	0
L(TR)	1-63	1-77
	64	0
L(others)	1-15	1-17
	16	0
I(jumps)	Label (Defined)	0-77
	(blank)	0
	1-63	1-77
N(JL)	EXPRESSION (Defined)	0-77
	0-63	0-77
C	Specific Graphic	0-77
	Decimal Value 0-63	0-77
	Octal Value 0-077	0-77
T	Label (Defined)	0-77
	0-63	0-77
	0-077	0-77

TABLE 5-5. ABBREVIATIONS IN WRITING INSTRUCTIONS

	WRITTEN		INTERNAL FORM (OCTAL) BIT POSITIONS							M Addr.	
			29-25	24-22	21	20....6	5	4	3 2 1 0	LSD.▲	TIMING
PAGE	OPERATION	OPERANDS	OPCODE*	X		M			L	MSD.■	
5-33	ZS$	M,L,X	22	0-7	RESERVED	0-77777	0	0	0-17	■	49.5 + 9E
5-33	ZS		22				1	0		■	45 + 9E
5-33	ZS*		22				1	1		■	45 + 9E
5-34	PD		26				0	0		▲	27 + 4.5L
5-34	PDO		26				0	1		▲	27 + 4.5L
5-32	ED		52				1	0		▲	36 + 13.5L + 9E
5-27	CDa		26				1	0/1		▲	36 + 13.5Lm
5-15	SAa		52				0			▲	27 + 9L
5-15	BAa		56				0			▲	27 + 9L
5-15	BDa		56				1			▲	31.5 + 9L
5-21	AMa		62				0			▲	49.5 + 13.5L
5-21	SMa		62				1			▲	49.5 + 13.5L
5-18	ADa		66				0			▲	49.5 + 13.5(L + Lc)
5-20	SDa		66				1			▲	49.5 + 13.5(L + Lc)
5-26	CBa		70				1			▲	27 + 13.5L
5-22	ABa		72				0			▲	27 + 13.5L
5-23	SBa		72				1			▲	27 + 13.5L
5-35	TFR	M,,X	24				0	0	0	■	90 + 9B
5-35	TFI		24				0	1		■	103.5 + 9B
5-35	TTR		24				1	0		■	90 + 9B
5-35	TTI		24				1	1		■	103.5 + 9B
5-15	SAR		52				1	1		▲	315
5-24	MPN	,L	50	0		0	0	0	0 0-7		L(33.75K + 27) + 54
5-24	MPC		50				0	1			L(33.75K + 27) − 27
5-25	DV		50				1	0			4.5L(74.25K + 13.75) + 54
				X		M		n	S		
5-37	BCn	M,S,X	16	0-7		0-77777	1	0-3	0-7	▲	40.5 + S(9 + 18n)
5-37	BSn	M,S,X	16	0-7		0-77777	0	0-3	0-7	▲	40.5 + S(9 + 18n)
				X		M					
5-31	JR	M,I,X	10	0-7		0-77777		0-77	(I)	†	45
5-29	JC	M,I,X	30						(I)	■	31.5
5-30	JL	M,N,X	32						(N)	■	40.5
5-28	LC	M,C,X	14						(C)	▲	40.5
5-27	CC		34							▲	40.5
5-16	SC		44							▲	40.5
5-36	LP		54							▲	40.5
5-23	AC		60							▲	45 + 13.5Lc
5-36	LS		64							▲	40.5
5-17	FT	M,T,X	20						(T)	▲	81
5-16	ST		42							▲	63
5-16	BT		46							▲	63
5-28	CT		74							▲	81
5-23	AT		76							▲	81
5-31	TR	M,L,X	12						(L)	▲	36 + 13.5L
			29-25	24-22	21-18	17-12			11 - 0		
			OPCODE	X	U	F			D		
5-63	XF	F,D,U,X	40	0-7	0-17	0-77			0-7777		72**

*Operation Codes 00, 02, 04, 06 and 36 are unassigned.

**Not including loading print buffer.

†LSD of address portion of instruction referenced.

Figure 5-6. Instruction Repertoire.

DATA TRANSFER INSTRUCTIONS

Data transfer instructions are used to transfer information from one part of internal storage to another. This group includes instructions which operate on single characters, variable length character fields, and tetrads. In these multicharacter transfer operations, the M position addresses the least significant character of the field. The transfer begins with this character and proceeds from right to left.

All data transfer instructions are indexable.

BRING DECIMAL — BDa M, L, X

OPERATION Brings L (from 1 to 16) characters from M_x to ARa (Arithmetic Register a). The zone bits of each character except the least significant character, which contains the sign, are changed to zeros.

A sentinel character (110011 = &) is inserted to the left of the last character transferred into ARa, if the specified length is less than 16 characters. Characters in ARa to the left of the sentinel are not affected.

EXAMPLE Bring decimal the five characters stored at ALPHA into AR1.

OPERATION 13 18	OPERANDS 19
B D 1	ALPHA, 5

	ALPHA	AR1
Before:	...xx76B2P	..xxxxxxxxxx
After:	(Same)	..xxx&7622P
Timing:	31.5 + 9 L	

BRING ALPHANUMERIC — BAa M, L, X

OPERATION Brings L (from 1 to 16) characters from M_x to ARa. All six (zone and numeric) bits of each character are transferred.

No sentinel character is inserted.

EXAMPLE Bring the 16 alphanumeric characters stored at BETA as indexed by Index Register 3, into AR2.

OPERATION 13 18	OPERANDS 19
B A 2	BETA, 16, 3

	BETA + (X3)
Before:	...xBCDEFGHIJKLM#56Qx...
After:	(Same)
	AR1
Before:	xxxxx xxxxx xxxxxx
After:	BCDEFGHIJKLM#56Q
Timing:	27 + 9 L

Since indexing is used in the example, the time for the operation is increased by 13.5 for a total of 27 + 13.5 + 9(16) = 184.5 microseconds.

STORE ARITHMETIC REGISTER — SAa M, L, X

OPERATION Stores L (from 1 to 16) characters from ARa into L positions of M_x.

EXAMPLE Store 8 characters from AR2 into TOTAL.

OPERATION 13 18	OPERANDS 19
S A 2	TOTAL, 8

	AR2	TOTAL
Before:	...x&00518436	xxxxxxxxxxxxxx
After:	(Same)	xxxx00518436xx
Timing:	27 + 9 L	

STORE BOTH ARITHMETIC REGISTERS — SAR M, ,X

OPERATION Stores the contents of AR1 and AR2 into 32 positions of M_x. The L field is not used by this operation.

EXAMPLE Store the contents of the Arithmetic Registers in TEMP.

OPERATION 13 18	OPERANDS 19
S A R	TEMP

INSTRUCTION REPERTOIRE

PAGE 5-	WRITTEN		INTERNAL FORM (OCTAL) BIT POSITIONS							M Addr.	TIMING
			29-25	24-22	21	20....6	5	4	3 2 1 0	LSD. ▲	
	OP-ERA-TION	OPER-ANDS	Op-code*	X		M			L	MSD. ■	
33	ZS$	M,L,X	22	0-7	RESERVED	0-77777	0	0	0-17	■	49.5 + 9E
33	ZS	↓	22	↓		↓	1	0	↓	■	45 + 9E
33	ZS✱	↓	22	↓		↓	1	1	↓	■	45 + 9E
34	PD	↓	26	↓		↓	0	0	↓	▲	27 + 4.5L
34	PDO	↓	26	↓		↓	0	1	↓	▲	27 + 4.5L
32	ED	↓	52	↓		↓	1	0	↓	▲	36 + 13.5L + 9E
27	CDa	↓	26	↓		↓	1	0/1	↓	▲	36 + 13.5Lm
15	SAa	↓	52	↓		↓	0	↓	↓	▲	27 + 9L
15	BAa	↓	56	↓		↓	0	↓	↓	▲	27 + 9L
15	BDa	↓	56	↓		↓	1	↓	↓	▲	31.5 + 9L
21	AMa	↓	62	↓		↓	0	↓	↓	▲	49.5 + 13.5L
21	SMa	↓	62	↓		↓	1	↓	↓	▲	49.5 + 13.5L
18	ADa	↓	66	↓		↓	0	↓	↓	▲	49.5 + 13.5(L + Lc)
20	SDa	↓	66	↓		↓	1	↓	↓	▲	49.5 + 13.5(L + Lc)
26	CBa	↓	70	↓		↓	1	↓	↓	▲	27 + 13.5L
22	ABa	↓	72	↓		↓	0	↓	↓	▲	27 + 13.5L
23	SBa	↓	72	↓		↓	1	↓	↓	▲	27 + 13.5L
35	TFR	M,,X	24	↓		↓	0	0	0	■	90 + 9B
35	TFI	↓	24	↓		↓	0	1	↓	■	103.5 + 9B
35	TTR	↓	24	↓		↓	1	0	↓	■	90 + 9B
35	TTI	↓	24	↓		↓	1	1	↓	■	103.5 + 9B
15	SAR	↓	52	↓		↓	1	1	↓	▲	315
24	MPN	,L	50	0		0	0	0	0 0-7		L(33.75K + 27) + 54
24	MPC	↓	50	↓		↓	0	1	↓ ↓		L(33.75K + 27)—27
25	DV	↓	50	↓		↓	1	0	↓ ↓		4.5L(74.25K + 13.75) + 54
				X		M		n	S		
37	BCn	M,S,X	16	0-7		0-77777	1	0-3	0-7	▲	40.5 + S(9 + 18n)
37	BSn	M,S,X	16	0-7		0-77777	0	0-3	0-7	▲	40.5 + S(9 + 18n)
				X		M					
31	JR	M,I,X	10	0-7		0-77777	0-77		(I)	†	45
29	JC	M,I,X	30	↓		↓	↓		(I)	■	31.5
30	JL	M,N,X	32	↓		↓	↓		(N)	■	40.5
28	LC	M,C,X	14	↓		↓	↓		(C)	▲	40.5
27	CC	↓	34	↓		↓	↓		↓	▲	40.5
16	SC	↓	44	↓		↓	↓		↓	▲	40.5
36	LP	↓	54	↓		↓	↓		↓	▲	40.5
23	AC	↓	60	↓		↓	↓		↓	▲	45 + 13.5Lc
36	LS	↓	64	↓		↓	↓		↓	▲	40.5
17	FT	M,T,X	20	↓		↓	↓		(T)	▲	81
16	ST	↓	42	↓		↓	↓		↓	▲	63
16	BT	↓	46	↓		↓	↓		↓	▲	63
28	CT	↓	74	↓		↓	↓		↓	▲	81
23	AT	↓	76	↓		↓	↓		↓	▲	81
31	TR	M,L,X	12	↓		↓	↓		(L)	▲	36 + 13.5L
			29-25	24-22	21-18	17-12	11		0		
			Op-Code	X	U	F	D				
63	XF	F,D,U,X	40	0-7	0-17	0-77	0-7777				72**

*Operation Codes 00, 02, 04, 06 and 36 are unassigned.
**Not including loading print buffer.
† LSD of address portion of instruction referenced.

	AR1	AR2
Before:	1234ABCDEFGHIJKL	MNOPQRSTUVWXYZ78
After:	(Same)	(Same)
		TEMP ↓
Before:	xxx ...	... xxx

After: ...xx1234ABCDEFGHIJKLMNOPQRSTUVWXYZ78x...

Timing: 315

BRING TO TETRAD — BT M, T, X

OPERATION Brings four contiguous characters from M_x into the Tetrad specified by T. The Arithmetic Registers are not involved in this operation, and their contents are not changed.

EXAMPLE Bring FILDA to Tetrad 14.

OPERATION	OPERANDS
13 18	19
BT	FILDA,14

	FILDA	Tetrad 14
Before:	...x2345x...	xxxx
After:	(Same)	2345
Timing:	63	

STORE TETRAD — ST M, T, X

OPERATION Stores the four characters of Tetrad T into four continguous positions at M_x. The Arithmetic Registers are not involved in this operation, and their contents are not changed.

EXAMPLE Store the contents of Tetrad 11 (Index Register 3) into CONT3.

OPERATION	OPERANDS
13 18	19
ST	CONT3,11
O	R
ST	CONT3, X3

	Tetrad 11	CONT3
Before:	2047	...xxxxxxxx...
After:	"	...xxx2047x...
Timing:	63	

STORE CHARACTER — SC M, C, X

OPERATION Stores the character represented by C into M_x. The Arithmetic Registers are not involved in this operation, and their contents are not changed.

EXAMPLE Store a decimal 3 at TAGA.

OPERATION 13 18	19 OPERANDS
SC	TAGA, '3'

	TAGA ↓
Before:	...xxx...
After:	...x3x...
Timing:	40.5

FIX TETRAD — FT M,T,X

OPERATION Place 15 bits specified by the actual value of M_x into the designated Tetrad. Note that M_x is *not* an address where this value is stored; it *is* the value. This value is placed into bits 14-0 of the Tetrad, bits 17-15 are set to zero, and the remainder of the tetrad, bits 23-18, is not affected.

If indexing is specified, the 15-bit contents of the index register specified by X are added to the 15 bits of M before M is stored in the Tetrad. Carries past the 15 position are ignored.

This instruction is useful in address modification since the 15 bits affected by it correspond to the length of the M portion of an instruction.

The FT sequence of operation is as follows:

1. The contents of CC (the Instruction Control Counter containing the address of the next instruction) are stored in Tetrad 19.
2. The 15 bits of the M_x portion of the FT instruction are stored in CC.
3. The contents of CC (i.e., the M address) are stored in the Tetrad specified by T.
4. The contents of Tetrad 19 are returned to CC and normal sequencing of the program is resumed.

If T specifies Tetrad 19, this instruction can be used to perform a jump operation.

NOTE: In the examples below, the contents of the tetrads are shown as 8 octal digits to represent the 24 bits. Lower case y's represent octal digits which do not affect the operation.

EXAMPLE 1 Fix Tetrad 14 (Index Register 6) to the value 02010. In symbolic coding this value has been equated to the label MAIN.

OPERATION 13 18	19 OPERANDS
FT	MAIN, 14

	Tetrad 14
Before:	y y y y y y y y
After:	y y y 0 2 0 1 0

EXAMPLE 2 Increment Tetrad 10 (Index Register 2) by the value INCRT. INCRT has been equated to a constant with a value of 03320.

OPERATION 13 18	19 OPERANDS
FT	INCRT, 10, 2

	Tetrad 10
Before:	y y y 0 0 2 0 0
After:	y y y 0 3 5 2 0

EXAMPLE 3 Use the FT instruction to change the contents of CC to NEXT, which has a value of 012300.

OPERATION 13 18	19 OPERANDS
FT	NEXT, 19

	Tetrad 19	CC
Before:	y y y y y y y y	y y y y y
After:	y y y 1 2 3 0 0	1 2 3 0 0
Timing:	81	

INSTRUCTION REPERTOIRE

P A G E 5-	WRITTEN		INTERNAL FORM (OCTAL) BIT POSITIONS							M Addr.	TIMING
	OP-ERA-TION	OPER-ANDS	29-25	24-22	21	20....6	5	4	3 2 1 0	LSD. ▲	
			Op-code*	X		M			L	MSD. ■	
33	ZS$	M,L,X	22	0-7		0-77777	0	0	0-17	■	49.5+9E
33	ZS	↓	22	↓		↓	1	0	↓	■	45+9E
33	ZS*	↓	22	↓		↓	1	1	↓	■	45+9E
34	PD	↓	26	↓		↓	0	0	↓	▲	27+4.5L
34	PDO	↓	26	↓		↓	0	1	↓	▲	27+4.5L
32	ED	↓	52	↓		↓	1	0	↓	▲	36+13.5L+9E
27	CDa	↓	26	↓		↓	1	0/1	↓	▲	36+13.5Lm
15	SAa	↓	52	↓		↓	0	↓	↓	▲	27+9L
15	BAa	↓	56	↓		↓	0	↓	↓	▲	27+9L
15	BDa	↓	56	↓		↓	1	↓	↓	▲	31.5+9L
21	AMa	↓	62	↓	RESERVED	↓	0	↓	↓	▲	49.5+13.5L
21	SMa	↓	62	↓		↓	1	↓	↓	▲	49.5+13.5L
18	ADa	↓	66	↓		↓	0	↓	↓	▲	49.5+13.5(L+Lc)
20	SDa	↓	66	↓		↓	1	↓	↓	▲	49.5+13.5(L+Lc)
26	CBa	↓	70	↓		↓	1	↓	↓	▲	27+13.5L
22	ABa	↓	72	↓		↓	0	↓	↓	▲	27+13.5L
23	SBa	↓	72	↓		↓	1	↓	↓	▲	27+13.5L
35	TFR	M,,X	24	↓		↓	0	0	0	■	90+9B
35	TFI	↓	24	↓		↓	0	1	↓	■	103.5+9B
35	TTR	↓	24	↓		↓	1	0	↓	■	90+9B
35	TTI	↓	24	↓		↓	1	1	↓	■	103.5+9B
15	SAR	↓	52	↓		↓	1	1	↓	▲	315
24	MPN	,L	50	0		0	0	0	0 0-7		L(33.75K+27)+54
24	MPC	↓	50	↓		↓	0	1	↓ ↓		L(33.75K+27)—27
25	DV	↓	50	↓		↓	1	0	↓ ↓		4.5L(74.25K+13.75)+54
				X		M		n	S		
37	BCn	M,S,X	16	0-7		0-77777	1	0-3	0-7	▲	40.5+S(9+18n)
37	BSn	M,S,X	16	0-7		0-77777	0	0-3	0-7	▲	40.5+S(9+18n)
				X		M					
31	JR	M,I,X	10	0-7		0-77777	0-77		(I)	†	45
29	JC	M,I,X	30	↓		↓	↓		(I)	■	31.5
30	JL	M,N,X	32	↓		↓	↓		(N)	■	40.5
28	LC	M,C,X	14	↓		↓	↓		(C)	▲	40.5
27	CC	↓	34	↓		↓	↓		↓	▲	40.5
16	SC	↓	44	↓		↓	↓		↓	▲	40.5
36	LP	↓	54	↓		↓	↓		↓	▲	40.5
23	AC	↓	60	↓		↓	↓		↓	▲	45+13.5Lc
36	LS	↓	64	↓		↓	↓		↓	▲	40.5
17	FT	M,T,X	20	↓		↓	↓		(T)	▲	81
16	ST	↓	42	↓		↓	↓		↓	▲	63
16	BT	↓	46	↓		↓	↓		↓	▲	63
28	CT	↓	74	↓		↓	↓		↓	▲	81
23	AT	↓	76	↓		↓	↓		↓	▲	81
31	TR	M,L,X	12	↓		↓	↓		(L)	▲	36+13.5L
			29-25	24-22	21-18	17-12	11 - 0				
			Op-Code	X	U	F	D				
63	XF	F,D,U,X	40	0-7	0-17	0-77	0-7777				72**

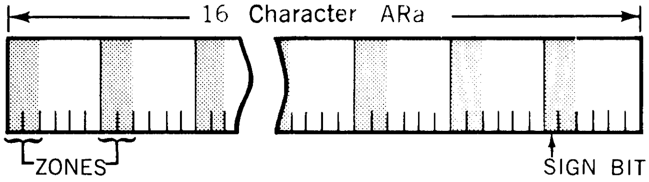
*Operation Codes 00, 02, 04, 06 and 36 are unassigned.
**Not including loading print buffer.
† LSD of address portion of instruction referenced.

ARITHMETIC INSTRUCTIONS

Both decimal and binary arithmetic operations can be performed by the UNIVAC 1050 System. Results are stored in either the Arithmetic Registers or designated storage locations, as defined by the individual instructions.

Carries are propagated normally up to the limits set by either the capacity of the Arithmetic Register, an AR length sentinel, or the conditions of the instruction. In a decimal operation, an attempted carry beyond these limits causes an automatic overflow interrupt, and KDF (Indicator 40) is set to 1. An attempted carry beyond these limits causes KNB (Indicator 39) to be set to 0 in a binary add and to 1 in a binary subtract.

Except for the sentinel bit and the sign bit, which is positioned in the AR as shown, zone bits in both AR and storage fields are ignored in all decimal add and subtract operations, and do not appear in the sum.



A decimal zero sum is signed positive, except in the single case (−0) + (−0) = −0. A false zero sum resulting when an overflow occurs in a decimal add or subtract operation will carry the sign of the full correct result.

In decimal operations, the four characters (blank, +, @, ≠) having internal form xx0000 will be converted to zeros (000011) before the operation. Decimal operations should not be performed with any of the following invalid numeric digits:

Binary Value	Source Characters
xx0001	] : * (
xx0010	– . $,
xx1101	\ = %)
xx1110	; < ’ >
xx1111	[# △ ⌑

In using the timing formulas given with each instruction, remember that actual times are functions of such variable factors as number of carries, number of blanks, a zero sum, complementation and recomplementation of negative values, and indexing, and hence the formulas are not exact in all cases.

All arithmetic instructions except Multiply and Divide are indexable.

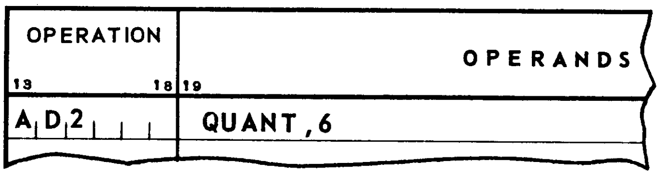
ADD DECIMAL—ADa M, L, X

OPERATION Performs a decimal algebraic addition of L (from 1 to 16) digits from M_x to the contents of ARa. The sum is stored in ARa. The length of the field in ARa is specified by the existing sentinel.

The following rules govern the operation of this instruction:

- The field length (including leading zeros) of the resulting sum will be equal to L or the length of the field in ARa, whichever is greater.
- If the length of the field in ARa is equal to or greater than L:
  - The operation is terminated when L characters from M_x have been added, except for the propagation of carries or presence of a zero sum.
  - Carries are propagated up to, but not including, the sentinel position. An attempted carry into the sentinel position generates a decimal overflow, setting KDF (Indicator 40) to 1.
- If the field in ARa is shorter than L:
  - The length of the field in ARa is increased to L by inserting as many zeros as necessary in the most significant character positions of ARa. An attempted carry into the sentinel position generates a decimal overflow, setting KDF (Indicator 40) to 1.
  - A new sentinel is inserted in ARa unless the new length is 16.

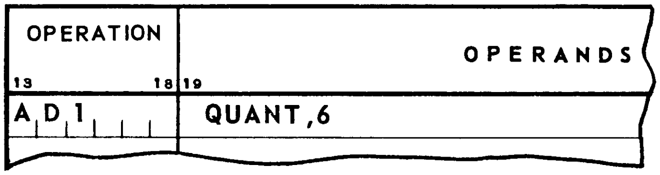
EXAMPLE 1 Add 6 digits at QUANT to AR2. The length of the field in AR2 is 7 characters.



OPERATION	OPERANDS
13 18	19
AD2	QUANT,6

	QUANT	AR2
Before:	...xx375826x...	...xx&6504226
After:	(Same)	...xx&6880052

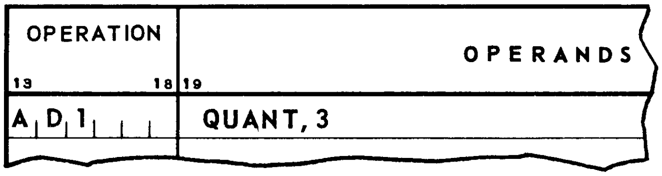
EXAMPLE 2 Add 6 digits at QUANT to AR1. The length of the field in AR1 is 4 characters.



OPERATION	OPERANDS
13 18	19
AD1	QUANT,6

	QUANT	AR1
Before:	xx375826x	...xxxx&0435
After:	(Same)	...xx&376261

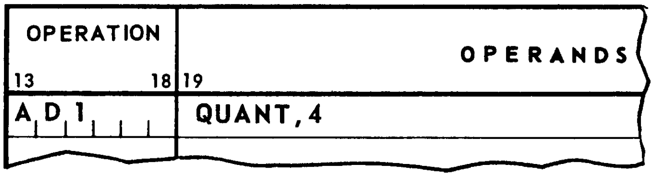
EXAMPLE 3 Add 3 digits at QUANT to AR1.



OPERATION	OPERANDS
13 18	19
AD1	QUANT,3

	QUANT	AR1
Before:	xx422xx	xx&7321P (−73217)
After:	(Same)	xx&7279N (−72795)

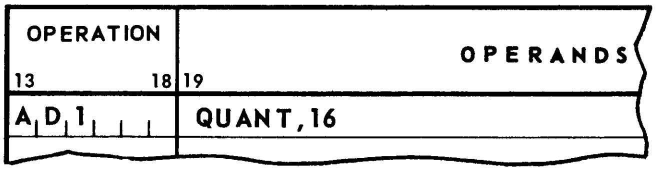
EXAMPLE 4 Add 4 digits at QUANT to AR1.



OPERATION	OPERANDS
13 18	19
AD1	QUANT,4

	QUANT	AR1
Before:	xx3175xx	xx&9471
After:	(Same)	xx&2646 with decimal overflow

EXAMPLE 5 Add 16 digits at QUANT to AR1.



OPERATION	OPERANDS
13 18	19
AD1	QUANT,16

	QUANT	AR1
Before:	xx3725478912543421xx	...xx&34721
After:	(Same)	3725478912578142

INSTRUCTION REPERTOIRE

PAGE	WRITTEN		INTERNAL FORM (OCTAL) BIT POSITIONS							M Addr.	TIMING
			29-25	24-22	21	20....6	5	4	3 2 1 0	LSD. ▲	
5-	OP-ERA-TION	OPER-ANDS	Op-code*	X		M			L	MSD. ■	
33	ZS$	M,L,X	22	0-7	RESERVED	0-77777	0	0	0-17	■	49.5+9E
33	ZS	↓	22	↓		↓	1	0	↓	■	45+9E
33	ZS✱	↓	22	↓		↓	1	1	↓	■	45+9E
34	PD	↓	26	↓		↓	0	0	↓	▲	27+4.5L
34	PDO	↓	26	↓		↓	0	1	↓	▲	27+4.5L
32	ED	↓	52	↓		↓	1	0	↓	▲	36+13.5L+9E
27	CDa	↓	26	↓		↓	1	0/1	↓	▲	36+13.5Lm
15	SAa	↓	52	↓		↓	0	↓	↓	▲	27+9L
15	BAa	↓	56	↓		↓	0	↓	↓	▲	27+9L
15	BDa	↓	56	↓		↓	1	↓	↓	▲	31.5+9L
21	AMa	↓	62	↓		↓	0	↓	↓	▲	49.5+13.5L
21	SMa	↓	62	↓		↓	1	↓	↓	▲	49.5+13.5L
18	ADa	↓	66	↓		↓	0	↓	↓	▲	49.5+13.5(L+Lc)
20	SDa	↓	66	↓		↓	1	↓	↓	▲	49.5+13.5(L+Lc)
26	CBa	↓	70	↓		↓	1	↓	↓	▲	27+13.5L
22	ABa	↓	72	↓		↓	0	↓	↓	▲	27+13.5L
23	SBa	↓	72	↓		↓	1	↓	↓	▲	27+13.5L
35	TFR	M,,X	24	↓		↓	0	0	0	■	90+9B
35	TFI	↓	24	↓		↓	0	1	↓	■	103.5+9B
35	TTR	↓	24	↓		↓	1	0	↓	■	90+9B
35	TTI	↓	24	↓		↓	1	1	↓	■	103.5+9B
15	SAR	↓	52	↓		↓	1	1	↓	▲	315
24	MPN	,L	50	0		0	0	0	0 0-7		L(33.75K+27)+54
24	MPC	↓	50	↓		↓	0	1	↓ ↓		L(33.75K+27)—27
25	DV	↓	50	↓		↓	1	0	↓ ↓		4.5L(74.25K+13.75)+54
				X		M		n	S		
37	BCn	M,S,X	16	0-7		0-77777	1	0-3	0-7	▲	40.5+S(9+18n)
37	BSn	M,S,X	16	0-7		0-77777	0	0-3	0-7	▲	40.5+S(9+18n)
				X		M					
31	JR	M,I,X	10	0-7		0-77777	0-77		(I)	†	45
29	JC	M,I,X	30	↓		↓	↓		(I)	■	31.5
30	JL	M,N,X	32	↓		↓	↓		(N)	■	40.5
28	LC	M,C,X	14	↓		↓	↓		(C)	▲	40.5
27	CC	↓	34	↓		↓	↓		↓	▲	40.5
16	SC	↓	44	↓		↓	↓		↓	▲	40.5
36	LP	↓	54	↓		↓	↓		↓	▲	40.5
23	AC	↓	60	↓		↓	↓		↓	▲	45+13.5Lc
36	LS	↓	64	↓		↓	↓		↓	▲	40.5
17	FT	M,T,X	20	↓		↓	↓		(T)	▲	81
16	ST	↓	42	↓		↓	↓		↓	▲	63
16	BT	↓	46	↓		↓	↓		↓	▲	63
28	CT	↓	74	↓		↓	↓		↓	▲	81
23	AT	↓	76	↓		↓	↓		↓	▲	81
31	TR	M,L,X	12	↓		↓	↓		(L)	▲	36+13.5L
			29-25	24-22	21-18	17-12	11 - 0				
			Op-Code	X	U	F	D				
63	XF	F,D,U,X	40	0-7	0-17	0-77	0-7777				72**

*Operation Codes 00, 02, 04, 06 and 36 are unassigned.
**Not including loading print buffer.
† LSD of address portion of instruction referenced.

EXAMPLE 6 Add 15 digits at QUANT to AR1.

OPERATION 13 – 18	OPERANDS 19
A D 1	QUANT, 15

	QUANT	AR1
Before:	xx999999994792183xx	0000008491233467
After:	(Same)	1000008486025650

Timing: $49.5+13.5\ (L+L_c)$ or $49.5+27L_m$ if the magnitude of the quantity at M_x exceeds that of the quantity in ARa and the signs of the two quantities are different. L_c represents the number of carries beyond L, and L_m the length of the larger field.

NOTE: The following indicators are set as a result of this operation.

	KZR 37	KM 38
If sum is positive nonzero	0	0
If sum is negative nonzero	0	1
If sum is + 0	1	0
If sum is − 0	1	1

SUBTRACT DECIMAL — SDa M, L, X

OPERATION Performs a decimal algebraic subtraction of L (from 1 to 16) digits of M_x from the contents of ARa. The difference is stored in ARa. The length of the field in ARa is specified by the existing sentinel.

The rules governing the operation of this instruction are the same as those for the Add Decimal Instruction, reading difference in place of sum, and subtracted in place of added.

EXAMPLE Subtract 6 digits at DIFER from AR2. The length of the field in AR2 is 6 characters.

OPERATION 13 – 18	OPERANDS 19
S D 2	DIFER, 6

	DIFER	AR2
Before:	...x000325x...	...xx&013496
After:	(Same)	...xx&013171

Timing: $49.5+13.5\ (L+L_c)$ or $49.5+27L_m$ if the magnitude of the quantity at M_x exceeds that of the quantity in ARa and the signs are the same. L_c represents the number of carries beyond L and L_m the length of the larger field. The following indicators are set as a result of this operation:

	KZR 37	KM 38
If difference is positive nonzero	0	0
If difference is negative nonzero	0	1
If difference is + 0	1	0
If difference is − 0	1	1

ADD TO MEMORY—AMa M, L, X

OPERATION Performs a decimal algebraic addition of ARa and L (from 1 to 16) digits of M_x. The sum is stored in M_x. The contents of ARa are not changed as a result of this operation.

The following rules govern the operation of this instruction:

- The length of the field in ARa must be less than or equal to L; if it is larger, the excess high order characters in ARa will not be used.
- If the length of the field in ARA is less than L, the length of the ARa field is effectively increased to L by inserting as many imaginary zeros as necessary in the most significant character positions of ARa. Actually ARa, including the sentinel and sign, is left unchanged at the end of the operation.
- An attempted carry from the most significant digit of the M_x field terminates the addition and causes a decimal overflow, setting KDF (Indicator 40) to 1.

EXAMPLE Add the contents of AR2 to the 6-digit field at QNTY.

OPERATION 13 18	OPERANDS 19
AM2	QNTY,6

	AR2	QNTY
Before:	...xxx&014	...xx563048xx...
After:	(Same)	...xx563062xx...

Timing: 49.5 + 13.5L or 49.5 + 31.5L if the magnitude of the quantity in ARa exceeds that of the quantity at M_x and the signs of the two quantities are different.

NOTE: The following indicators are set as a result of this operation:

	KZR 37	KM 38
If sum is positive nonzero	0	0
If sum is negative nonzero	0	1
If sum is + 0	1	0
If sum is − 0	1	1

SUBTRACT FROM MEMORY—SMa M, L, X

OPERATION Performs a decimal algebraic subtraction of ARa from L (from 1 to 16) digits from M_x. The difference is stored in M_x. The contents of ARa are not changed as a result of this operation.

This instruction effectively reverses the sign of ARa and then operates as the Add to Memory instruction, and the rules governing its operation are the same.

EXAMPLE Subtract the contents of AR2 from the 5 digits at BLNCE.

OPERATION 13 18	OPERANDS 19
SM2	BLNCE,5

	AR2	BLNCE
Before:	...xxx&40263	..xx87947xx....
After:	(Same)	..xx47684xx....
Before:	...&773M(−7734)	..xx94234xx....
After:	(Same)	..xx01968xx.... (decimal overflow) (Indicator 40 set to 1)

The following indicators are set as a result of this operation:

	KZR 37	KM 38
If difference is positive nonzero	0	0
If difference is negative nonzero	0	1
If difference is + 0	1	0
If difference is − 0	1	1

Timing: 49.5 + 13.5L or 49.5 + 31.5L if the magnitude of the quantity in ARa exceeds that of the quantity at M_x and the signs of the two quantities are the same.

INSTRUCTION REPERTOIRE

P A G E 5 –	WRITTEN		INTERNAL FORM (OCTAL) BIT POSITIONS							M Addr.	TIMING
	OP-ERA-TION	OPER-ANDS	29-25	24-22	21	20....6	5	4	3 2 1 0	LSD. ▲	
			Op-code*	X		M			L	MSD. ■	
33	ZS$	M,L,X	22	0-7	RESERVED	0-77777	0	0	0-17	■	49.5+9E
33	ZS	↓	22	↓		↓	1	0	↓	■	45+9E
33	ZS✱	↓	22	↓		↓	1	1	↓	■	45+9E
34	PD	↓	26	↓		↓	0	0	↓	▲	27+4.5L
34	PDO	↓	26	↓		↓	0	1	↓	▲	27+4.5L
32	ED	↓	52	↓		↓	1	0	↓	▲	36+13.5L+9E
27	CDa	↓	26	↓		↓	1	0/1	↓	▲	36+13.5Lm
15	SAa	↓	52	↓		↓	0	↓	↓	▲	27+9L
15	BAa	↓	56	↓		↓	0	↓	↓	▲	27+9L
15	BDa	↓	56	↓		↓	1	↓	↓	▲	31.5+9L
21	AMa	↓	62	↓		↓	0	↓	↓	▲	49.5+13.5L
21	SMa	↓	62	↓		↓	1	↓	↓	▲	49.5+13.5L
18	ADa	↓	66	↓		↓	0	↓	↓	▲	49.5+13.5(L+Lc)
20	SDa	↓	66	↓		↓	1	↓	↓	▲	49.5+13.5(L+Lc)
26	CBa	↓	70	↓		↓	1	↓	↓	▲	27+13.5L
22	ABa	↓	72	↓		↓	0	↓	↓	▲	27+13.5L
23	SBa	↓	72	↓		↓	1	↓	↓	▲	27+13.5L
35	TFR	M,,X	24	↓		↓	0	0	0	■	90+9B
35	TFI	↓	24	↓		↓	0	1	↓	■	103.5+9B
35	TTR	↓	24	↓		↓	1	0	↓	■	90+9B
35	TTI	↓	24	↓		↓	1	1	↓	■	103.5+9B
15	SAR	↓	52	↓		↓	1	1	↓	▲	315
24	MPN	,L	50	0		0	0	0	0 0-7		L(33.75K+27)+54
24	MPC	↓	50	↓		↓	0	1	↓ ↓		L(33.75K+27)—27
25	DV	↓	50	↓		↓	1	0	↓ ↓		4.5L(74.25K+13.75)+54
				X		M		n	S		
37	BCn	M,S,X	16	0-7		0-77777	1	0-3	0-7	▲	40.5+S(9+18n)
37	BSn	M,S,X	16	0-7		0-77777	0	0-3	0-7	▲	40.5+S(9+18n)
				X		M					
31	JR	M,I,X	10	0-7		0-77777		0-77	(I)	†	45
29	JC	M,I,X	30	↓		↓		↓	(I)	■	31.5
30	JL	M,N,X	32	↓		↓		↓	(N)	■	40.5
28	LC	M,C,X	14	↓		↓		↓	(C)	▲	40.5
27	CC	↓	34	↓		↓		↓	↓	▲	40.5
16	SC	↓	44	↓		↓		↓	↓	▲	40.5
36	LP	↓	54	↓		↓		↓	↓	▲	40.5
23	AC	↓	60	↓		↓		↓	↓	▲	45+13.5Lc
36	LS	↓	64	↓		↓		↓	↓	▲	40.5
17	FT	M,T,X	20	↓		↓		↓	(T)	▲	81
16	ST	↓	42	↓		↓		↓	↓	▲	63
16	BT	↓	46	↓		↓		↓	↓	▲	63
28	CT	↓	74	↓		↓		↓	↓	▲	81
23	AT	↓	76	↓		↓		↓	↓	▲	81
31	TR	M,L,X	12	↓		↓		↓	(L)	▲	36+13.5L
			29-25	24-22	21-18	17-12	11 - 0				
			Op-Code	X	U	F	D				
63	XF	F,D,U,X	40	0-7	0-17	0-77	0-7777				72**

*Operation Codes 00, 02, 04, 06 and 36 are unassigned.
**Not including loading print buffer.
† LSD of address portion of instruction referenced.

ADD BINARY — ABa M, L, X,

OPERATION Performs a binary addition of ARa and L (from 1 to 16) characters of M_x. The sum is stored in M_x. The Arithmetic Registers are unchanged as a result of this operation.

Note that the instruction specifies L *characters;* therefore the number of bits involved is always 6L.

No algebraic signs are associated with the operands; all six bits of each character position are treated as data bits.

Carries are not propagated beyond the most significant bit; an attempted carry beyond this bit position will cause a binary overflow condition, and indicator 39 is set to 0. A "no overflow" sets indicator 39 to 1.

EXAMPLE 1 Add, in binary, three characters from AR2 to TOTAL.

OPERATION 13 18	OPERANDS 19
A B 2	TOTAL, 3

	AR2	TOTAL
Before:	. . . x 1 1 0 (Octal 040403)	. . . x x A 5 W x x . . . (Octal 241071)
After:	(Same)	. . . x x E 9 Z x x . . . (Octal 301474)

EXAMPLE 2 Add, in binary, two characters from AR2 to TOTAL.

OPERATION 13 18	OPERANDS 19
A B 2	TOTAL, 2

	AR2	TOTAL
Before:	x x 4 7 (Octal 0712)	x x Q 4 x x (Octal 5307)
After:	(Same)	x x , : x x (Octal 6221)
Timing:	27 + 13.5L	

NOTE: The following indicators are set as a result of this operation:

	KZR 37	KNB 39
If sum is not zero, and binary overflow occurs	0	0
If sum is not bero, and there is no binary overflow	0	1
If sum is zero, and binary overflow occurs	1	0
If sum is zero, and there is no binary overflow	1	1

SUBTRACT BINARY— SBa M, L, X,

OPERATION Performs a binary subtraction of ARa from L (from 1 to 16) characters of M_x. The difference is stored in M_x. The Arithmetic Registers are unchanged as a result of this operation.

Note that the instruction specifies L *characters;* therefore the number of bits involved in always 6L.

No algebraic signs are associated with the operands; all six bits of each character are treated as data bits.

This instruction adds the 2's complement of ARa to the L characters of M_x. The normal execution of this instruction with a positive result will attempt a carry, and Indicator 39 is set to 1. If no carry is attempted, the result will be the complement of the true result, and indicator 39 is set to 0.

EXAMPLE Subtract, in binary, three characters in AR2 from QNTY.

OPERATION 13 18	OPERANDS 19
SB2	QNTY, 3

	AR2	QNTY
Before:	. . . x 6 5 3 . . . (Octal 111006)	. . . x x 86K x . . . (Octal 131145)
After:	(Same)	. . . x x -]# x . . . (Octal 020137)
Timing:	27 + 13.5 L	

NOTE: The following indicators are set as a result of this operation:

	KZR 37	KNB 39
If difference is not zero, and no carry is attempted	0	0
If difference is not zero, and carry is attempted	0	1
If difference is 0, and no carry is attempted	1	0
If difference is 0, and carry is attempted	1	1

ADD TO TETRAD—AT M, T, X

OPERATION Performs a binary addition of the contents of a four-character field at M_x to the contents of the Tetrad specified by T. The sum is stored in the Tetrad. The contents of ARa are not changed as a result of this operation.

Note that both M_x and T consist of four *characters;* therefore the operation always involves 24 bit positions, including leading zeros.

No algebraic signs are associated with the operands; all six bits of each character are treated as data bits.

EXAMPLE Add the contents of LOCNA to the contents of Tetrad 10 (X2).

OPERATION 13 18	OPERANDS 19
AT	LOCNA, X2

	LOCNA	Tetrad 10
Before:	. . . x x x 1 6 3 8 x x x . . .	4 9 3 8
After:	(Same)	8 B 9 C

Carries are propagated up to the most significant bit of the Tetrad. An attempt to carry beyond this bit will set overflow indicator KNB (39) to 0; no overflow sets the indicator to 1. An actual overflow does not take place.

Timing: 81

ADD CHARACTER—AC M, C, X

OPERATION Performs a binary addition of the 6-bit characters of C and M_x, and stores the sum in M_x. The contents of ARa are not changed as a result of this operation. No algebraic signs are associated with the operands; all six bits of each character are treated as data bits.

Carries are allowed to propogate as far as necessary. A carry to the character at $(M_x - 1)$ sets Indicator 39 to 0; if there is no carry beyond the character at M_x, Indicator 39 is set to 1.

INSTRUCTION REPERTOIRE

PAGE	WRITTEN		INTERNAL FORM (OCTAL) BIT POSITIONS							M Addr.	TIMING
			29-25	24-22	21	20....6	5	4	3 2 1 0	LSD. ▲ MSD. ■	
5-	OP-ERA-TION	OPER-ANDS	Op-code*	X		M			L		
33	ZS$	M,L,X	22	0-7	RESERVED	0-77777	0	0	0-17	■	49.5+9E
33	ZS		22				1	0		■	45+9E
33	ZS✱		22				1	1		■	45+9E
34	PD		26				0	0		▲	27+4.5L
34	PDO		26				0	1		▲	27+4.5L
32	ED		52				1	0		▲	36+13.5L+9E
27	CDa		26				1	0/1		▲	36+13.5Lm
15	SAa		52				0			▲	27+9L
15	BAa		56				0			▲	27+9L
15	BDa		56				1			▲	31.5+9L
21	AMa		62				0			▲	49.5+13.5L
21	SMa		62				1			▲	49.5+13.5L
18	ADa		66				0			▲	49.5+13.5(L+Lc)
20	SDa		66				1			▲	49.5+13.5(L+Lc)
26	CBa		70				1			▲	27+13.5L
22	ABa		72				0		↓	▲	27+13.5L
23	SBa	↓	72				1	↓		▲	27+13.5L
35	TFR	M,,X	24				0	0	0	■	90+9B
35	TFI		24				0	1		■	103.5+9B
35	TTR		24				1	0		■	90+9B
35	TTI		24				1	1		■	103.5+9B
15	SAR	↓	52	↓		↓	1	1	↓	▲	315
24	MPN	,L	50	0		0	0	0	0 0-7		L(33.75K+27)+54
24	MPC		50				0	1			L(33.75K+27)—27
25	DV	↓	50	↓		↓	1	0	↓ ↓		4.5L(74.25K+13.75)+54
				X		M		n	S		
37	BCn	M,S,X	16	0-7		0-77777	1	0-3	0-7	▲	40.5+S(9+18n)
37	BSn	M,S,X	16	0-7		0-77777	0	0-3	0-7	▲	40.5+S(9+18n)
				X		M					
31	JR	M,I,X	10	0-7		0-77777		0-77	(I)	†	45
29	JC	M,I,X	30						(I)	■	31.5
30	JL	M,N,X	32						(N)	■	40.5
28	LC	M,C,X	14						(C)	▲	40.5
27	CC		34							▲	40.5
16	SC		44							▲	40.5
36	LP		54							▲	40.5
23	AC		60							▲	45+13.5Lc
36	LS	↓	64						↓	▲	40.5
17	FT	M,T,X	20						(T)	▲	81
16	ST		42							▲	63
16	BT		46							▲	63
28	CT		74							▲	81
23	AT	↓	76						↓	▲	81
31	TR	M,L,X	12	↓		↓		↓	(L)	▲	36+13.5L
			29-25	24-22	21-18	17-12		11 - 0			
			Op-Code	X	U	F		D			
63	XF	F,D,U,X	40	0-7	0-17	0-77		0-7777			72**

*Operation Codes 00, 02, 04, 06 and 36 are unassigned.
**Not including loading print buffer.
† LSD of address portion of instruction referenced.

EXAMPLE Change the character '1' (000100) at CODEA to character 'A' (010100) by adding the special character '+' (010000).

OPERATION 13 18	OPERANDS 19
A C	CODEA,'+'

CODEA

Before: ...xx1x...

After: ...xxAx...

Timing: 45 + 13.5L_c

where L_c represents the number of character positions a carry is propagated.

MULTIPLY NONCUMULATIVE — MPN ,L
MULTIPLY CUMULATIVE — MPC ,L

OPERATION Performs a decimal algebraic multiplication of the contents of AR2 (multiplicand) and L (from 1 to 8) digits of MLR (multiplier, Tetrads 20-21). Both factors must be positioned by previous instructions. The product is stored in AR1 with leading zeros to fill out the register. This form is appropriate for subsequent use in further cumulative multiplication, or for subsequent addition or subtraction; the effective length is 16.

The length of the multiplicand is determined by the length sentinel in AR2; a maximum length of 15 is possible. If a sentinel appears in the least significant position of AR2, the multiplicand is considered as zero. L is the length of the multiplier. The total number of digits in both multiplier and multiplicand must not exceed 16; otherwise overflow may occur, setting indicator 40 to 1 and causing an overflow interrupt. The following are permissible combinations in multiplication:

L	Allowable length of multiplicand in AR2
1	1-15
2	1-14
3	1-13
4	1-12
5	1-11
6	1-10
7	1-9
8	1-8

In these instructions, the product is always added to the contents of AR1. The MPN instruction first clears AR1 to zeros, and for this reason is called noncumulative. MPC does not clear AR1, and any sentinel is treated as decimal zero.

The sign of the product is governed by normal algebraic rules. Like signs produce a positive product, and unlike signs produce a negative product. Care must be taken with the sign of the field in AR1, for correct cumulative multiplication is possible only if the sign of AR1 before multiplication is the same as the sign of the product. The sign of AR1 is assumed to be the same as that of the product. For example, regardless of whether AR1 contains +3 or −3:

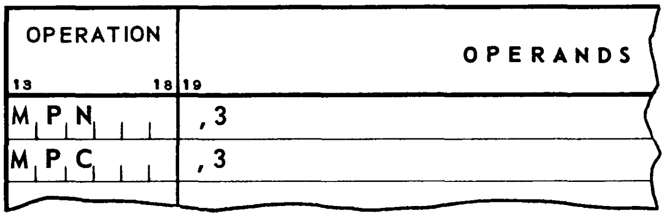
$(\ \ 3 \times 4)+AR1=+15$

$(-3 \times 4)+AR1=-15$

$(-3 \times -4)+AR1=+15$

Multiplication destroys MLR but leaves the multiplicand in AR2 unaltered. This instruction is not indexable.

EXAMPLE Multiply a 5-digit multiplicand by a 3-digit multiplier. AR1 contains the value 2054.



OPERATION 13 18	OPERANDS 19
MPN	,3
MPC	,3

	AR1	AR2	MLR
Before:	..xx00002054	...xx&13256	xxxxx182
After MPN:	..xx02412592	Same	xxxxxxxx
After MPC:	..xx02414646	Same	xxxxxxxx

The following indicators are set as a result of the MPN operations:

	KZR 37	KM 38
If (cumulative) result is positive nonzero	0	0
If (cumulative) result is negative nonzero	0	1
If (cumulative) result is +0	1	0
If (cumulative) result is −0	1	1

The following indicator is set as a result of the MPC operation:

	KM 38
If (cumulative) result is positive nonzero	0
If (cumulative) result is negative nonzero	1
If (cumulative) result is +0	0
If (cumulative) result is −0	1

Timing:	(33.75K + 27) L − 27	for MPC
	(33.75K + 27) L + 45	for MPN

where K is the number of digits in the multiplicand.

DIVIDE — DV ,L

OPERATION Performs a decimal algebraic division of the contents of AR1 (dividend) by the contents of AR2 (divisor), and stores the quotient in QTN (Tetrads 20-21). The length of the quotient is specified by L, and a maximum of 8 digits is possible. The remainder, if any, is stored in AR1, and has the sign of the original dividend.

The dividend must be stored in AR1 by a previous instruction. If the length of the dividend is less than the length of the divisor + the length of the quotient, it must be extended to that length by padding zeros. PD0 to the left of the dividend after it is loaded into AR1. The divisor must be stored in AR2 by a previous instruction, with a sentinel. An undetected improper division occurs if:

- the length of the dividend field is greater than the length of the quotient + the length of the divisor; or
- The length of the quotient + the length of the divisor is greater than 16.

Decimal overflow will occur, and Indicator 40 will be set to 1, if the absolute value of the divisor, shifted L digits to the left, is smaller than the absolute value of the dividend, and the condition for improper division does not occur.

The divisor in AR2 is not changed as a result of this operation.

INSTRUCTION REPERTOIRE

PAGE 5-	WRITTEN		INTERNAL FORM (OCTAL) BIT POSITIONS							M Addr.	TIMING
			29-25	24-22	21	20....6	5	4	3 2 1 0	LSD. ▲	
	OP-ERA-TION	OPER-ANDS	Op-code*	X		M			L	MSD. ■	
33	ZS$	M,L,X	22	0-7		0-77777	0	0	0-17	■	49.5+9E
33	ZS	↓	22	↓		↓	1	0	↓	■	45+9E
33	ZS✱	↓	22	↓		↓	1	1	↓	■	45+9E
34	PD	↓	26	↓		↓	0	0	↓	▲	27+4.5L
34	PDO	↓	26	↓		↓	0	1	↓	▲	27+4.5L
32	ED	↓	52	↓		↓	1	0	↓	▲	36+13.5L+9E
27	CDa	↓	26	↓		↓	1	0/1	↓	▲	36+13.5Lm
15	SAa	↓	52	↓		↓	0	↓	↓	▲	27+9L
15	BAa	↓	56	↓		↓	0	↓	↓	▲	27+9L
15	BDa	↓	56	↓		↓	1	↓	↓	▲	31.5+9L
21	AMa	↓	62	↓		↓	0	↓	↓	▲	49.5+13.5L
21	SMa	↓	62	↓		↓	1	↓	↓	▲	49.5+13.5L
18	ADa	↓	66	↓	RESERVED	↓	0	↓	↓	▲	49.5+13.5(L+Lc)
20	SDa	↓	66	↓		↓	1	↓	↓	▲	49.5+13.5(L+Lc)
26	CBa	↓	70	↓		↓	1	↓	↓	▲	27+13.5L
22	ABa	↓	72	↓		↓	0	↓	↓	▲	27+13.5L
23	SBa	↓	72	↓		↓	1	↓	↓	▲	27+13.5L
35	TFR	M,,X	24	↓		↓	0	0	0	■	90+9B
35	TFI	↓	24	↓		↓	0	1	↓	■	103.5+9B
35	TTR	↓	24	↓		↓	1	0	↓	■	90+9B
35	TTI	↓	24	↓		↓	1	1	↓	■	103.5+9B
15	SAR	↓	52	↓		↓	1	1	↓	▲	315
24	MPN	,L	50	0		0	0	0	0 0-7		L(33.75K+27)+54
24	MPC	↓	50	↓		↓	0	1	↓ ↓		L(33.75K+27)—27
25	DV	↓	50	↓		↓	1	0	↓ ↓		4.5L(74.25K+13.75)+54
				X		M		n	S		
37	BCn	M,S,X	16	0-7		0-77777	1	0-3	0-7	▲	40.5+S(9+18n)
37	BSn	M,S,X	16	0-7		0-77777	0	0-3	0-7	▲	40.5+S(9+18n)
				X		M					
31	JR	M,I,X	10	0-7		0-77777	0-77		(I)	†	45
29	JC	M,I,X	30	↓		↓	↓		(I)	■	31.5
30	JL	M,N,X	32	↓		↓	↓		(N)	■	40.5
28	LC	M,C,X	14	↓		↓	↓		(C)	▲	40.5
27	CC	↓	34	↓		↓	↓		↓	▲	40.5
16	SC	↓	44	↓		↓	↓		↓	▲	40.5
36	LP	↓	54	↓		↓	↓		↓	▲	40.5
23	AC	↓	60	↓		↓	↓		↓	▲	45+13.5Lc
36	LS	↓	64	↓		↓	↓		↓	▲	40.5
17	FT	M,T,X	20	↓		↓	↓		(T)	▲	81
16	ST	↓	42	↓		↓	↓		↓	▲	63
16	BT	↓	46	↓		↓	↓		↓	▲	63
28	CT	↓	74	↓		↓	↓		↓	▲	81
23	AT	↓	76	↓		↓	↓		↓	▲	81
31	TR	M,L,X	12	↓		↓	↓		(L)	▲	36+13.5L
			29-25	24-22	21-18	17-12	11 - 0				
			Op-Code	X	U	F	D				
63	XF	F,D,U,X	40	0-7	0-17	0-77	0-7777				72**

*Operation Codes 00, 02, 04, 06 and 36 are unassigned.
**Not including loading print buffer.
† LSD of address portion of instruction referenced.

The sign of the quotient is governed by the normal algebraic rules.

This instruction is not indexable.

EXAMPLE Divide a 5-digit field by a 2-digit field, and store a 4-digit quotient.

OPERATION 13 18	OPERANDS 19
DV	,4

	AR1	AR2	QTN
Before:	. . . xx007230	. . . xx&17	xxxxxxxx
After:	. . . xxxxxx05	. . . xx&17	xxxx0425

Timing: 4.5L(74.25K + 13.75) + 54

Where L is the length of the quotient, and K is the length of the divisor.

NOTE: The following indicators are set as a result of this operation.

	KZR 37	KM 38
If quotient is positive and remainder is nonzero	0	0
If quotient is negative and remainder is nonzero	0	1
If quotient is positive and remainder is zero	1	0
If quotient is negative and remainder is zero	1	1

COMPARISON

These instructions compare two values, and the result of the comparison is represented by the settings of the following testable indicators:

	INDICATOR	
	NAME	NUMBER
High	KHI	33
Equal	KEQ	34
Unequal	KUQ	35
Low	KLO	36

The indicators can be tested by the program and appropriate action can be taken.

Neither of the two fields tested is changed as a result of a comparison.

These instructions are indexable.

COMPARE BINARY — CBa M, L, X

OPERATION Performs a binary comparison of L (from 1 to 16) characters at M_x with L characters of ARa. The length sentinel in ARa is treated as another 6-bit character and may be involved in the comparison.

Since L specifies a length in *characters*, the number of bits involved in the comparison will be 6L.

Comparison is on the basis of the absolute binary value of the compared fields; each bit of each character is treated as a data bit. The result of the comparison is recorded in the testable indicators as shown below:

	KHI 33	KEQ 34	KUQ 35	KLO 36
If (AR) < (M_x)	0	0	1	1
If (AR) = (M_x)	0	1	0	0
If (AR) > (M_x)	1	0	1	0

EXAMPLE Compare the two characters at CODEX with the two least significant characters of AR1.

OPERATION 13 18	19 OPERANDS
C B 1	CODEX, 2

```
CODEX               AR1
...B3...            ...xxxB2
```

The bit configuration of the two characters at M is 010101 000110. The bit configuration of the two characters in AR1 is 010101 000101. The absolute binary value in AR1 is less than that in M, and indicators 35 and 36 are set to 1.

Timing: 27 + 13.5L

COMPARE CHARACTER — CC M, C, X

OPERATION Performs an absolute binary comparison of the bits in the character in M_x with the bits of the character represented by C.

The following indicators are set as a result of this instruction:

	KHI 33	KEQ 34	KUQ 35	KLO 36
If C < (M_x)	0	0	1	1
If C = (M_x)	0	1	0	0
If C > (M_x)	1	0	1	0

EXAMPLE Compare character D with the character at TAGB.

OPERATION 13 18	19 OPERANDS
C C	TAGB, 'D'

```
TAGB
...xGx...
```

The bit configuration of the character D is 010111, which is equivalent to 27_8. Similarly the character G is equivalent to 32_8. The result is C < (M_x) and indicators 35 and 36 are set to 1's, while indicators 33 and 34 are set to 0's.

Timing: 40.5

COMPARE DECIMAL — CDa M, L, X

OPERATION Performs a decimal algebraic comparison of L (from 1 to 16) characters at M_x with the contents of ARa. The length of the field in ARa is determined by the length sentinel.

The following rules govern the operation of this instruction:

1. If the signs of the two fields are unlike, the comparison is immediately terminated, and the result is recorded in the testable indicators as shown below:

	KHI 33	KEQ 34	KUQ 35	KLO 36
IF (AR) < (M_x)	0	0	1	1
IF (AR) = (M_x)	0	1	0	0
IF (AR) > (M_x)	1	0	1	0

2. If the fields being compared are of unequal lengths, the shorter field will be lengthened by the insertion of as many imaginary decimal zeros as necessary in the most significant character positions. This generation of zeros takes place only in the comparator circuits, and the actual contents of either ARa or M_x are not changed.

EXAMPLE Compare decimal the 5-character field at CONST with the 7-character field in AR2.

OPERATION 13 18	19 OPERANDS
C D 2	CONST, 5

INSTRUCTION REPERTOIRE

PAGE 5-	WRITTEN		INTERNAL FORM (OCTAL) BIT POSITIONS							M Addr.	TIMING
	OP-ERA-TION	OPER-ANDS	29-25 Op-code*	24-22 X	21	20....6 M	5	4	3 2 1 0 L	LSD. ▲ MSD. ■	
33	ZS$	M,L,X	22	0-7	RESERVED	0-77777	0	0	0-17	■	49.5+9E
33	ZS	↓	22	↓		↓	1	0	↓	■	45+9E
33	ZS✱	↓	22	↓		↓	1	1	↓	■	45+9E
34	PD	↓	26	↓		↓	0	0	↓	▲	27+4.5L
34	PDO	↓	26	↓		↓	0	1	↓	▲	27+4.5L
32	ED	↓	52	↓		↓	1	0	↓	▲	36+13.5L+9E
27	CDa	↓	26	↓		↓	1	0/1	↓	▲	36+13.5Lm
15	SAa	↓	52	↓		↓	0	↓	↓	▲	27+9L
15	BAa	↓	56	↓		↓	0	↓	↓	▲	27+9L
15	BDa	↓	56	↓		↓	1	↓	↓	▲	31.5+9L
21	AMa	↓	62	↓		↓	0	↓	↓	▲	49.5+13.5L
21	SMa	↓	62	↓		↓	1	↓	↓	▲	49.5+13.5L
18	ADa	↓	66	↓		↓	0	↓	↓	▲	49.5+13.5(L+Lc)
20	SDa	↓	66	↓		↓	1	↓	↓	▲	49.5+13.5(L+Lc)
26	CBa	↓	70	↓		↓	1	↓	↓	▲	27+13.5L
22	ABa	↓	72	↓		↓	0	↓	↓	▲	27+13.5L
23	SBa	↓	72	↓		↓	1	↓	↓	▲	27+13.5L
35	TFR	M,,X	24	↓		↓	0	0	0	■	90+9B
35	TFI	↓	24	↓		↓	0	1	↓	■	103.5+9B
35	TTR	↓	24	↓		↓	1	0	↓	■	90+9B
35	TTI	↓	24	↓		↓	1	1	↓	■	103.5+9B
15	SAR	↓	52	↓		↓	1	1	↓	▲	315
24	MPN	,L	50	0		0	0	0	0 0-7		L(33.75K+27)+54
24	MPC	↓	50	↓		↓	0	1	↓ ↓		L(33.75K+27)—27
25	DV	↓	50	↓		↓	1	0	↓ ↓		4.5L(74.25K+13.75)+54
				X		M		n	S		
37	BCn	M,S,X	16	0-7		0-77777	1	0-3	0-7	▲	40.5+S(9+18n)
37	BSn	M,S,X	16	0-7		0-77777	0	0-3	0-7	▲	40.5+S(9+18n)
				X		M					
31	JR	M,I,X	10	0-7		0-77777		0-77	(I)	†	45
29	JC	M,I,X	30	↓		↓		↓	(I)	■	31.5
30	JL	M,N,X	32	↓		↓		↓	(N)	■	40.5
28	LC	M,C,X	14	↓		↓		↓	(C)	▲	40.5
27	CC	↓	34	↓		↓		↓	↓	▲	40.5
16	SC	↓	44	↓		↓		↓	↓	▲	40.5
36	LP	↓	54	↓		↓		↓	↓	▲	40.5
23	AC	↓	60	↓		↓		↓	↓	▲	45+13.5Lc
36	LS	↓	64	↓		↓		↓	↓	▲	40.5
17	FT	M,T,X	20	↓		↓		↓	(T)	▲	81
16	ST	↓	42	↓		↓		↓	↓	▲	63
16	BT	↓	46	↓		↓		↓	↓	▲	63
28	CT	↓	74	↓		↓		↓	↓	▲	81
23	AT	↓	76	↓		↓		↓	↓	▲	81
31	TR	M,L,X	12	↓		↓		↓	(L)	▲	36+13.5L
			29-25	24-22	21-18	17-12	11 - 0				
			Op-Code	X	U	F	D				
63	XF	F,D,U,X	40	0-7	0-17	0-77	0-7777				72**

*Operation Codes 00, 02, 04, 06 and 36 are unassigned.
**Not including loading print buffer.
† LSD of address portion of instruction referenced.

	CONST	AR2
Before:	x x x 1 3 5 8 2 x x	. . . x&0014400
Treated as:	x 0 0 1 3 5 8 2 x x	. . . x&0014400
After:	x x x 1 3 5 8 2 x x	. . . x&0014400

Since the contents of AR2 arc greater than the contents of the specified positions, indicators 33 and 35 are set to 1.

Timing: $36 + 13.5L_m$ (Like Signs)

36 (Unlike Signs)

Where L_m is the length of the longer of the two fields.

COMPARE TETRAD — CT M, T, X

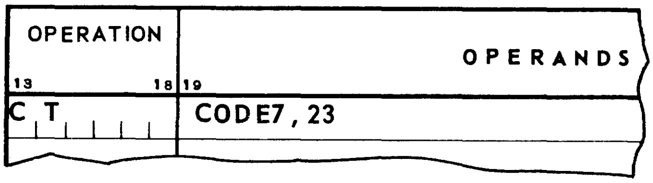
OPERATION Performs a binary comparison of the contents of the Tetrad specified by T and the four characters at M_x. Four full characters (24 bits) are always involved in the comparison.

Algebraic signs are not considered; each bit of each character is treated as a data bit.

The comparison is on the basis of the absolute binary value of the fields compared. The result of the comparison is stored in the testable indicators as shown below:

	KHI 33	KEQ 34	KUQ 35	KLO 36
If (T) < (M_x)	0	0	1	1
If (T) = (M_x)	0	1	0	0
If (T) > (M_x)	1	0	1	0

EXAMPLE Compare the contents of Tetrad 23 with the 4 characters of CODE7.



OPERATION 13 18	OPERANDS 19
C T	CODE7, 23

Tetrad 23	CODE7
1234	. . . xx1234xx . . .

Binary bit configurations for both fields are the same, and indicators are set accordingly.

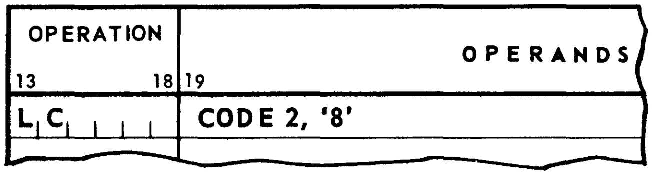
Timing: 81

LOGICAL COMPARE — LC M, C, X

OPERATION Performs a comparison of the bits of the character at M_x with the bits of the character represented by C. The comparison takes place only for those bit positions in M_x which correspond to 1 bits in C. If *all* bit positions in M_x which correspond to 1 bits in C are also 1 bits, the result is equality, and indicators are set as shown in the table below. When inequality occurs, the character specified by C is considered to be higher in value.

at the 1 positions in mask	KHI 33	KEQ 34	KUQ 35	KLO 36
If C = (M_x)	0	1	0	0
If C ≠ (M_x)	1	0	1	0

EXAMPLE Compare the 1 bits of the character '8' with the 1 bits of the character at CODE2.



OPERATION 13 18	OPERANDS 19
LC	CODE 2, '8'

Since '8' is represented internally as 001011, equality is established at CODE2 if bits 0, 1 and 3 of the character are also 1 bits. Therefore the following characters are considered equal to the character '8' in this example:

8 = 001011 (same as mask)

H = 011011

Q = 101011

Y = 111011

[= 001111

# = 011111

△ = 101111

¤ = 111111

Timing: 40.5

SEQUENCE CONTROL INSTRUCTIONS

Normally the instructions in a program are stored in five consecutive character positions, and are executed sequentially. When the conditions of a program require a break in this sequence, the sequence control instructions are used. They can test the status of specified indicators, and as a result of the test transfer control to one of two program paths, that is, the program will either execute the next instruction in sequence, or "jump" to the location designated by M_x.

All of the sequence control instructions are indexable.

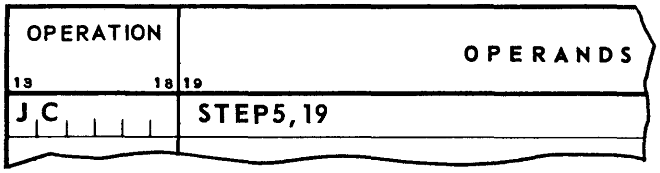
JUMP CONDITIONAL—JC M, I, X

OPERATION Performs a jump to the instruction stored at M_x if the indicator specified by I is set to 1. If the indicator is set to 0, the next sequential instruction is executed.

The following rules govern the operation of this instruction:

- An indicator is set by a previous operation, and remains set until another instruction which sets the indicator is executed.
- Testing of an indicator does not affect the setting except in the case of decimal overflow (Indicator 40) operator interrupt (Indicator 44) and Trace Indicator (Indicator 57) which are reset to 0 when tested.
- Arithmetic Registers are not affected as a result of the jump instructions.
- Indicators 00-31 are unconditional jumps; they perform a specified operation (See Appendix B) and jump to M_x. Indicators 32 and 56 are "no operation" instructions; they have no other functions.
- Indicators 32-63 (except 32, 41, 42, and 56) are conditional jumps; if the indicator is set to 1, the program jumps to M_x. Indicators 32 and 56 are "no operation" instructions; they have no other functions. Indicators 41 and 42 store and restore Indicators 33-40 in memory position M_x, they do not cause a jump.

EXAMPLE 1 Set Sense Indicator 2 to 1, and jump to the instruction with the label STEP5.



OPERATION 13 18	OPERANDS 19
JC	STEP5, 19

INSTRUCTION REPERTOIRE

PAGE 5-	WRITTEN OPERATION	WRITTEN OPERANDS	INTERNAL FORM (OCTAL) BIT POSITIONS 29-25 Op-code*	24-22 X	21	20....6 M	5	4	3 2 1 0 L	M Addr. LSD. ▲ MSD. ■	TIMING
33	ZS$	M,L,X	22	0-7	RESERVED	0-77777	0	0	0-17	■	49.5+9E
33	ZS	↓	22	↓		↓	1	0	↓	■	45+9E
33	ZS✱	↓	22	↓		↓	1	1	↓	■	45+9E
34	PD	↓	26	↓		↓	0	0	↓	▲	27+4.5L
34	PDO	↓	26	↓		↓	0	1	↓	▲	27+4.5L
32	ED	↓	52	↓		↓	1	0	↓	▲	36+13.5L+9E
27	CDa	↓	26	↓		↓	1	0/1	↓	▲	36+13.5Lm
15	SAa	↓	52	↓		↓	0	↓	↓	▲	27+9L
15	BAa	↓	56	↓		↓	0	↓	↓	▲	27+9L
15	BDa	↓	56	↓		↓	1	↓	↓	▲	31.5+9L
21	AMa	↓	62	↓		↓	0	↓	↓	▲	49.5+13.5L
21	SMa	↓	62	↓		↓	1	↓	↓	▲	49.5+13.5L
18	ADa	↓	66	↓		↓	0	↓	↓	▲	49.5+13.5(L+Lc)
20	SDa	↓	66	↓		↓	1	↓	↓	▲	49.5+13.5(L+Lc)
26	CBa	↓	70	↓		↓	1	↓	↓	▲	27+13.5L
22	ABa	↓	72	↓		↓	0	↓	↓	▲	27+13.5L
23	SBa		72	↓		↓	1	↓		▲	27+13.5L
35	TFR	M,,X	24	↓		↓	0	0	0	■	90+9B
35	TFI	↓	24	↓		↓	0	1	↓	■	103.5+9B
35	TTR	↓	24	↓		↓	1	0	↓	■	90+9B
35	TTI	↓	24	↓		↓	1	1	↓	■	103.5+9B
15	SAR	↓	52	↓		↓	1	1	↓	▲	315
24	MPN	,L	50	0		0	0	0	0 0-7		L(33.75K+27)+54
24	MPC	↓	50	↓		↓	0	1	↓ ↓		L(33.75K+27)—27
25	DV	↓	50	↓		↓	1	0	↓ ↓		4.5L(74.25K+13.75)+54
				X		M		n	S		
37	BCn	M,S,X	16	0-7		0-77777	1	0-3	0-7	▲	40.5+S(9+18n)
37	BSn	M,S,X	16	0-7		0-77777	0	0-3	0-7	▲	40.5+S(9+18n)
				X		M					
31	JR	M,I,X	10	0-7		0-77777	0-77		(I)	†	45
29	JC	M,I,X	30	↓		↓	↓		(I)	■	31.5
30	JL	M,N,X	32	↓		↓	↓		(N)	■	40.5
28	LC	M,C,X	14	↓		↓	↓		(C)	▲	40.5
27	CC	↓	34	↓		↓	↓		↓	▲	40.5
16	SC	↓	44	↓		↓	↓		↓	▲	40.5
36	LP	↓	54	↓		↓	↓		↓	▲	40.5
23	AC	↓	60	↓		↓	↓		↓	▲	45+13.5Lc
36	LS	↓	64	↓		↓	↓		↓	▲	40.5
17	FT	M,T,X	20	↓		↓	↓		(T)	▲	81
16	ST	↓	42	↓		↓	↓		↓	▲	63
16	BT	↓	46	↓		↓	↓		↓	▲	63
28	CT	↓	74	↓		↓	↓		↓	▲	81
23	AT	↓	76	↓		↓	↓		↓	▲	81
31	TR	M,L,X	12	↓		↓	↓		(L)	▲	36+13.5L
			29-25	24-22	21-18	17-12	11 - 0				
			Op-Code	X	U	F	D				
63	XF	F,D,U,X	40	0-7	0-17	0-77	0-7777				72**

*Operation Codes 00, 02, 04, 06 and 36 are unassigned.
**Not including loading print buffer.
† LSD of address portion of instruction referenced.

EXAMPLE 2 If the result of the last arithmetic operation was zero, jump to the routine starting with the instruction labeled CALCX. A zero result would have set indicator 37 to 1.

OPERATION 13 18	OPERANDS 19
JC	CALCX ,37

Timing: 31.5

JUMP LOOP—JL M, N, X

OPERATION This instruction is normally the last of a series of instructions which are to be executed N times. N can be a maximum of 63.

Each time this instruction is executed, the binary value of N contained in the instruction is first decremented by 1. If this result is zero, the next instruction in sequence will be executed. If the result is not zero, the address of the next instruction is M_x The value in N is never decremented past 000000. Thus if N has the initial value of 0 or 1, the loop will be executed (this initial time). The Arithmetic Registers and indicators are not changed as a result of this instruction.

EXAMPLE Execute the loop beginning at SUBRT 24 times.

The preferred method of coding this example is:

LABEL 7 11	OPERATION 13 18	OPERANDS 19
SUBRT		
		LOOP TO BE
		EXECUTED
	JL	SUBRT,24

An alternate to the above coding is:

LABEL 7 11	OPERATION 13 18	OPERANDS 19
SUBRT	JL	$ + 10, 25
	JC	NEXT
		LOOP TO BE
		EXECUTED
	JC	SUBRT
NEXT		

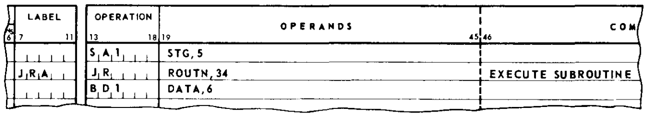
Timing: 40.5

JUMP RETURN — JR M, I, X

OPERATION Stores the contents of the control counter in the address portion of the instruction at M_x and performs a jump to the instruction *following* the one at M_x. Both the storing of the control counter and the jump are conditional upon the indicator I. The rules governing the operation of the JC instruction also apply to the JR instruction. The symbolic form of the JR instruction allows the instruction at M_x to be addressed by its label. This is translated by PAL into the address which is required in the machine form of the JR instruction.

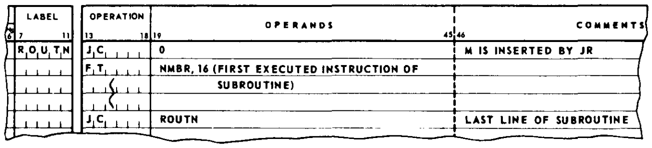
EXAMPLE If the result of a previous compare operation had set Indicator 34 to 1, jump to a subroutine beginning at ROUTN. The JR instruction is stored in JRA. At the conclusion of the subroutine return to the instruction at JRA + 5 (the instruction following the JR).

The main chain of coding could be:



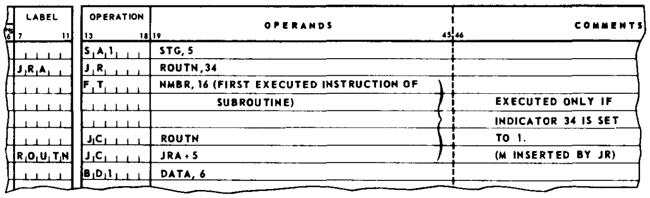
LABEL 7–11	OPERATION 13–18	OPERANDS 19–45	COM 46
	SA1	STG, 5	
JRA	JR	ROUTN, 34	EXECUTE SUBROUTINE
	BD1	DATA, 6	

The subroutine, appearing elsewhere in the coding, has the form:



LABEL 7–11	OPERATION 13–18	OPERANDS 19–45	COMMENTS 46
ROUTN	JC	0	M IS INSERTED BY JR
	FT	NMBR, 16 (FIRST EXECUTED INSTRUCTION OF	
	}	SUBROUTINE)	
	}		
	JC	ROUTN	LAST LINE OF SUBROUTINE

The order of execution of instructions in the above example is:



LABEL 7–11	OPERATION 13–18	OPERANDS 19–45	COMMENTS 46
	SA1	STG, 5	
JRA	JR	ROUTN, 34	
	FT	NMBR, 16 (FIRST EXECUTED INSTRUCTION OF	
		SUBROUTINE)	EXECUTED ONLY IF
			INDICATOR 34 IS SET
	JC	ROUTN	TO 1.
ROUTN	JC	JRA + 5	(M INSERTED BY JR)
	BD1	DATA, 6	

Timing: 45

CONVERSION AND EDIT INSTRUCTIONS

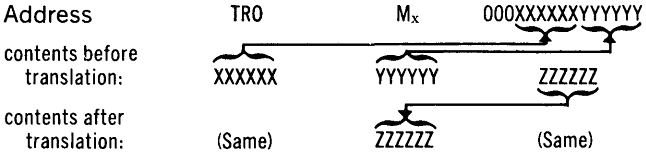
These instructions are used to change the format of information in memory. Four types of modification are possible:

1. Change the bit configuration *(Translate)*.
2. Insert punctuation and other symbols *(Edit)*.
3. Suppress zeros and replace with asterisks or a floating dollar sign *(Zero Suppress)*.
4. Place zeros or blank characters in any position of memory (Pad zeros or blanks).

TRANSLATE — TR M, L, X

OPERATION Replaces the L (from 1 to 64) characters at M_x with characters selected from a translate table. The first position of the translate table contains the character which is the replacement for the character with bit configuration 000000, the second position contains the replacement for the character with configuration 000001, the third position contains the replacement for the character with bit configuration 000010, etc. The table occupies one row of memory, i.e., 64 consecutive positions starting at an address which is a multiple of 64. The table must be contained in the first 4096 positions of memory.

The character at M_x provides bits 5-0 of the address of its replacement. TRO (the leftmost position of Tetrad 18) contains bits 11-6 of the address and the remaining bits of the address are zero. The replacement of the character at M_x may be pictured, showing all six bits of each character, as



After replacing the character at M_x the replacement for the character at M_x-1 is obtained in the same manner and so on until L characters have been replaced.

INSTRUCTION REPERTOIRE

PAGE 5-	WRITTEN OP-ERA-TION	WRITTEN OPER-ANDS	INTERNAL FORM (OCTAL) BIT POSITIONS 29-25 Op-code*	24-22 X	21	20....6 M	5	4	3 2 1 0 L	M Addr. LSD. ▲ MSD. ■	TIMING
33	ZS$	M,L,X	22	0-7	RESERVED	0-77777	0	0	0-17	■	49.5+9E
33	ZS	↓	22	↓		↓	1	0	↓	■	45+9E
33	ZS✱	↓	22	↓		↓	1	1	↓	■	45+9E
34	PD	↓	26	↓		↓	0	0	↓	▲	27+4.5L
34	PDO	↓	26	↓		↓	0	1	↓	▲	27+4.5L
32	ED	↓	52	↓		↓	1	0	↓	▲	36+13.5L+9E
27	CDa	↓	26	↓		↓	1	0/1	↓	▲	36+13.5Lm
15	SAa	↓	52	↓		↓	0	↓	↓	▲	27+9L
15	BAa	↓	56	↓		↓	0	↓	↓	▲	27+9L
15	BDa	↓	56	↓		↓	1	↓	↓	▲	31.5+9L
21	AMa	↓	62	↓		↓	0	↓	↓	▲	49.5+13.5L
21	SMa	↓	62	↓		↓	1	↓	↓	▲	49.5+13.5L
18	ADa	↓	66	↓		↓	0	↓	↓	▲	49.5+13.5(L+Lc)
20	SDa	↓	66	↓		↓	1	↓	↓	▲	49.5+13.5(L+Lc)
26	CBa	↓	70	↓		↓	1	↓	↓	▲	27+13.5L
22	ABa	↓	72	↓		↓	0	↓	↓	▲	27+13.5L
23	SBa	↓	72	↓		↓	1	↓	↓	▲	27+13.5L
35	TFR	M,,X	24	↓		↓	0	0	0	■	90+9B
35	TFI	↓	24	↓		↓	0	1	↓	■	103.5+9B
35	TTR	↓	24	↓		↓	1	0	↓	■	90+9B
35	TTI	↓	24	↓		↓	1	1	↓	■	103.5+9B
15	SAR	↓	52	↓		↓	1	1	↓	▲	315
24	MPN	,L	50	0		0	0	0	0 0-7		L(33.75K+27)+54
24	MPC	↓	50	↓		↓	0	1	↓ ↓		L(33.75K+27)—27
25	DV	↓	50	↓		↓	1	0	↓ ↓		4.5L(74.25K+13.75)+54
				X		M		n	S		
37	BCn	M,S,X	16	0-7		0-77777	1	0-3	0-7	▲	40.5+S(9+18n)
37	BSn	M,S,X	16	0-7		0-77777	0	0-3	0-7	▲	40.5+S(9+18n)
				X		M					
31	JR	M,I,X	10	0-7		0-77777		0-77	(I)	†	45
29	JC	M,I,X	30	↓		↓		↓	(I)	■	31.5
30	JL	M,N,X	32	↓		↓		↓	(N)	■	40.5
28	LC	M,C,X	14	↓		↓		↓	(C)	▲	40.5
27	CC	↓	34	↓		↓		↓	↓	▲	40.5
16	SC	↓	44	↓		↓		↓	↓	▲	40.5
36	LP	↓	54	↓		↓		↓	↓	▲	40.5
23	AC	↓	60	↓		↓		↓	↓	▲	45+13.5Lc
36	LS	↓	64	↓		↓		↓	↓	▲	40.5
17	FT	M,T,X	20	↓		↓		↓	(T)	▲	81
16	ST	↓	42	↓		↓		↓	↓	▲	63
16	BT	↓	46	↓		↓		↓	↓	▲	63
28	CT	↓	74	↓		↓		↓	↓	▲	81
23	AT	↓	76	↓		↓		↓	↓	▲	81
31	TR	M,L,X	12	↓		↓		↓	(L)	▲	36+13.5L
			29-25	24-22	21-18	17-12		11 - 0			
			Op-Code	X	U	F		D			
63	XF	F,D,U,X	40	0-7	0-17	0-77		0-7777			72**

*Operation Codes 00, 02, 04, 06 and 36 are unassigned.
**Not including loading print buffer.
† LSD of address portion of instruction referenced.

EXAMPLE At ADRS is a 16 character field which it is desired to print. The field is in FIELDATA code and must be translated to XS3 code for printing. In locations 1024 to 1087 are contained 64 characters:

bbbbbbABCDEFGHIJKLMNOPQRSTUVWXYZ)—+<=>&$*(b:bb;b
0123456789'b/.bb

(b represents a blank)

OPERATION 13 18	19 OPERANDS
TR	ADRS, 16

	TRO	ADRS ↓
Before:	x+xx	&#(b5;F<b3H7bbbb
After:	(Same)	301bCITYbAVEbbbb

TIMING: 36+13.5L.

EDIT — ED M, L, X

OPERATION Provides a simple method of inserting symbols, numbers, and letters into the body of a data field. The data field must be stored in AR1 by a previous operation, and the controlling field or mask (containing the editing and control symbols) must similarly be stored in AR2. The length of the field to be edited is specified by L, the resulting edited field is stored in M_x. The area in storage which is to receive the edited field must be large enough to permit storage of the data characters, the inserted symbols, and a sign if desired. The maximum number of characters which can be edited is 16 minus the number of inserted characters.

The mask in AR2 is a picture of the resultant edited field. An '@' (100000) character appears in each position in AR2 which corresponds to a position in the edited field which is to receive a character from AR1.

The least significant character of the mask may be a '-' (000010), in which case the rightmost position of the edited field will contain a blank if the field in AR1 is positive and a '-' if the field in AR1 is negative.

If the least significant character of the mask is a ⧖ (111 111), the least significant character of AR1 is transferred to the edited field without its zone bits.

There must be at least L '@' characters in the mask, except when the least significant character of the mask is ⧖. In this case, there must be at least L-1 '@' characters in the mask.

EXAMPLE Edit the 6-character field at FTBED by inserting a — if the field is negative, inserting a decimal point to separate dollars and cents and inserting a comma to indicate thousands of dollars. Store the result at TAG3.

The sequence of instructions is:

OPERATION 13 18	OPERANDS 19
BA2	MASK,9
BD1	FTBED,6
ED	TAG3,6

After execution of the first two instructions, AR1 & AR2 contains:

```
       AR1                    AR2 (MASK)
...xx&123456           ...xx@,@@@.@@—
```

After execution of the third (ED) instruction, storage contains

```
              TAG3
               ↓
...xx1,234.56  xx...
```

If the field to be edited (FTBED) had been negative, storage would contain

```
              TAG3
               ↓
...xx1,234.56—xx...
```

In most cases, since the edit mask is not destroyed when used, the first instruction (BA2 | MASK, 9) is not necessary for each item of a string of items to be edited. Also the second instructions (BD1 | FTBED, 6) is not necessary if the field to be edited is a result of a prior arithmetic operation performed in AR1.

Timing: 36 + 13.5 L + 9 E

where E is the number of characters inserted.

ZERO SUPPRESS ZS M, L, X

ZS$ M, L, X

ZS* M, L, X

OPERATION The Zero Suppress instruction performs any of three operations:

ZS Clears all leading commas, zeros and blanks of the field at M_x to blanks

ZS$ Clears all leading commas, zeros and blanks of the field at M_x to blanks, and inserts a dollar sign to the left of the remaining digits of the field.

ZS* Replaces all leading commas, zeros and blanks of the field at M_x with asterisks.

The character at position M_x is tested to see if it is blank, decimal zero, or a comma; if so, it is replaced as specified in the instruction. This process continues from left to right, for a maximum of L. (from 1 to 16) characters until a character is encountered which is neither a blank, a decimal zero, nor a comma, at which time the suppression ceases. In a ZS$ operation, the $ is then inserted in the position immediately to the left of the first character encountered that is not a blank, decimal zero or comma. A binary count (from 0 to 16) of the number of characters suppressed is developed and stored in ZCT (the second character position of Tetrad 18). The Arithmetic Registers are not altered by this instruction.

EXAMPLE Zero suppress the field NET showing the coding and results for the three variations.

OPERATION 13 18	OPERANDS 19
ZS	NET – 7, 8
ZS$	NET – 7, 8
ZS*	NET – 7, 8

```
                             NET           ZCT
                              ↓             ↓
Before:      ...xxx0003257 4xxx...        xxxx
After ZS:    ...xxx     32574xxx...       x0xx
After ZS$:   ...xxx    $32574xxx...       x0xx
After ZS*:   ...xxx***32574xxx...         x0xx
```

INSTRUCTION REPERTOIRE

PAGE 5-	WRITTEN OP-ERA-TION	 OPER-ANDS	INTERNAL FORM (OCTAL) BIT POSITIONS 29-25 Op-code*	 24-22 X	 21	 20....6 M	 5	 4	 3 2 1 0 L	M Addr. LSD. ▲ MSD. ■	TIMING
33	ZS$	M,L,X	22	0-7	RESERVED	0-77777	0	0	0-17	■	49.5+9E
33	ZS	↓	22	↓		↓	1	0	↓	■	45+9E
33	ZS✱	↓	22	↓		↓	1	1	↓	■	45+9E
34	PD	↓	26	↓		↓	0	0	↓	▲	27+4.5L
34	PDO	↓	26	↓		↓	0	1	↓	▲	27+4.5L
32	ED	↓	52	↓		↓	1	0	↓	▲	36+13.5L+9E
27	CDa	↓	26	↓		↓	1	0/1	↓	▲	36+13.5Lm
15	SAa	↓	52	↓		↓	0	↓	↓	▲	27+9L
15	BAa	↓	56	↓		↓	0	↓	↓	▲	27+9L
15	BDa	↓	56	↓		↓	1	↓	↓	▲	31.5+9L
21	AMa	↓	62	↓		↓	0	↓	↓	▲	49.5+13.5L
21	SMa	↓	62	↓		↓	1	↓	↓	▲	49.5+13.5L
18	ADa	↓	66	↓		↓	0	↓	↓	▲	49.5+13.5(L+Lc)
20	SDa	↓	66	↓		↓	1	↓	↓	▲	49.5+13.5(L+Lc)
26	CBa	↓	70	↓		↓	1	↓	↓	▲	27+13.5L
22	ABa	↓	72	↓		↓	0	↓	↓	▲	27+13.5L
23	SBa	↓	72	↓		↓	1	↓	↓	▲	27+13.5L
35	TFR	M,,X	24	↓		↓	0	0	0	■	90+9B
35	TFI	↓	24	↓		↓	0	1	↓	■	103.5+9B
35	TTR	↓	24	↓		↓	1	0	↓	■	90+9B
35	TTI	↓	24	↓		↓	1	1	↓	■	103.5+9B
15	SAR	↓	52	↓		↓	1	1	↓	▲	315
24	MPN	,L	50	0		0	0	0	0 0-7		L(33.75K+27)+54
24	MPC	↓	50	↓		↓	0	1	↓ ↓		L(33.75K+27)—27
25	DV	↓	50	↓		↓	1	0	↓ ↓		4.5L(74.25K+13.75)+54
				X		M		n	S		
37	BCn	M,S,X	16	0-7		0-77777	1	0-3	0-7	▲	40.5+S(9+18n)
37	BSn	M,S,X	16	0-7		0-77777	0	0-3	0-7	▲	40.5+S(9+18n)
				X		M					
31	JR	M,I,X	10	0-7		0-77777		0-77	(I)	†	45
29	JC	M,I,X	30	↓		↓		↓	(I)	■	31.5
30	JL	M,N,X	32	↓		↓		↓	(N)	■	40.5
28	LC	M,C,X	14	↓		↓		↓	(C)	▲	40.5
27	CC	↓	34	↓		↓		↓	↓	▲	40.5
16	SC	↓	44	↓		↓		↓	↓	▲	40.5
36	LP	↓	54	↓		↓		↓	↓	▲	40.5
23	AC	↓	60	↓		↓		↓	↓	▲	45+13.5Lc
36	LS	↓	64	↓		↓		↓	↓	▲	40.5
17	FT	M,T,X	20	↓		↓		↓	(T)	▲	81
16	ST	↓	42	↓		↓		↓	↓	▲	63
16	BT	↓	46	↓		↓		↓	↓	▲	63
28	CT	↓	74	↓		↓		↓	↓	▲	81
23	AT	↓	76	↓		↓		↓	↓	▲	81
31	TR	M,L,X	12	↓		↓		↓	(L)	▲	36+13.5L
			29-25	24-22	21-18	17-12	11	-	0		
			Op-Code	X	U	F		D			
63	XF	F,D,U,X	40	0-7	0-17	0-77		0-7777			72**

*Operation Codes 00, 02, 04, 06 and 36 are unassigned.
**Not including loading print buffer.
† LSD of address portion of instruction referenced.

Timing: 45 + 9E

Where E is the number of suppressed zeros. Add 4.5 if a dollar sign is inserted. Subtract 4.5 is the field length is 16.

PAD ZEROS — PDO M, L, X
PAD BLANKS — PD M, L, X

OPERATION Places decimal zeros or blanks, as indicated, into L storage positions whose least significant position is at M_x. The Arithmetic Registers are not involved in this operation, and their contents are not changed as a result of it.

EXAMPLE Place 3 zeros to the left of the most significant positions of a 5-character field named TOTAL.

OPERATION	OPERANDS
13 18	19
PDO	TOTAL - 5, 3

TOTAL

Before: ...xxxx52842xxx...

After: ...x00052842xxx...

Timing: 27 + 4.5 L

BLOCK TRANSFER INSTRUCTIONS

These instructions provide the facility to move the contents of a block of consecutive memory positions to another block of consecutive memory positions. The number of characters to be transferred (block size) is specified by the binary value of the ten least significant bits of TCT (Tetrad 18).

Two forms of the instruction are available: a Transfer From and a Transfer To. M_x addresses the most significant character of the origin or destination block. If M_x addresses the most significant character of the origin block, the binary value of the fifteen least significant bits of DST (Tetrad 16) specify the most significant character of the destination block. Similarly if M_x addresses the most significant character of the destination block ORG (Tetrad 17) specifies the first character address of the origin block. The transfer is from left to right.

Tetrads 16 (or 17) and 18 must be loaded by previous instructions. The contents of Tetrads 16 or 17, whichever is called for by the instruction, can be reset to the original address value, or set at an address greater by 1 than the last character transferred. The block size in Tetrad 18 is unaltered by any of these instructions.

During the Block Transfer, Tetrad 19 is used to store the contents of the instruction control counter (CC). At the end of the Block Transfer operation, the contents of Tetrad 19 are returned to CC and the normal sequence of instructions is resumed from CC.

The Block Transfer instructions are indexable.

TRANSFER BLOCK FROM MEMORY AND RESET—TFR M, ,X

OPERATION Transfer a block of characters beginning with the character at M_x to the block whose first character position is addressed by the contents of DST (Tetrad 16). The block size in binary is designated by the contents of TCT (Tetrad 18). After the execution of this instruction, Tetrad 16 is reset to its original value.

EXAMPLE Transfer a block of 75 (octal 113) characters, starting with GAMMA + 1 to DELTA (octal 07641).

OPERATION 13 18	OPERANDS 19
TFR	GAMMA + 1

			Tetrads		
	GAMMA + 1	DELTA	DST 16	ORG 17	TCT 18
Before:	135 . . . 732	xxx . . . xxx	00007641	xxxx . .	00000113
After:	(Same)	135 . . . 732	(Same)	(Same)	(Same)

NOTE: Contents of tetrads are shown in octal.

Timing: 90 + 9B

Where B is the number of characters transferred (Tetrad 18).

TRANSFER BLOCK FROM MEMORY AND INCREMENT—TFI M, ,X

OPERATION Identical to TFR except that after execution, Tetrad 16 is set to a value one greater than the address of the last character transferred.

EXAMPLE Transfer a block of 75 (octal 113) characters starting with GAMMA + 1 to DELTA (octal 7641).

OPERATION 13 18	OPERANDS 19
TFI	GAMMA + 1

			Tetrads		
	GAMMA + 1	DELTA	DST 16	ORG 17	TCT 18
Before:	135 . . . 732	xxx . . . xxx	00007641	xxxx	00000113
After:	(Same)	135 . . . 732	00007754	xxxx	(Same)

Timing: 103.5 + 9B

Where B is the number of characters transferred (Tetrad 18).

TRANSFER BLOCK TO MEMORY AND RESET—TTR M, ,X

OPERATION Transfers a block of characters beginning with the character addressed in ORG (Tetrad 17) to the block whose first character position is addressed by M_x. The block size in binary is specified by the contents of TCT (Tetrad 18). After the execution of this instruction, Tetrad 17 is reset to its original value.

EXAMPLE Transfer a block of 30 characters from ZETA (octal 1166) to DELTA.

OPERATION 13 18	OPERANDS 19
TTR	DELTA

			Tetrads		
	ZETA	DELTA	DST 16	ORG 17	TCT 18
Before:	JJS . . . MAH	xxx . . . xxx	xxxx	00001166	00000036
After:	(Same)	JJS . . . MAH	(Same)	(Same)	(Same)

Timing: 90 + 9 B

Where B is the number of characters transferred (Tetrad 18).

TRANSFER BLOCK TO MEMORY AND INCREMENT—TTI M, ,X

OPERATION Identical to TTR, except that after the execution, Tetrad 17 is set to a value one greater than the address of the last character transferred.

EXAMPLE Transfer a block of 30 characters from ZETA (octal 1166) to DELTA.

INSTRUCTION REPERTOIRE

PAGE 5-	WRITTEN		INTERNAL FORM (OCTAL) BIT POSITIONS							M Addr.	TIMING
			29-25	24-22	21	20....6	5	4	3 2 1 0	LSD. ▲	
	OP-ERA-TION	OPER-ANDS	Op-code*	X		M			L	MSD. ■	
33	ZS$	M,L,X	22	0-7	RESERVED	0-77777	0	0	0-17	■	49.5+9E
33	ZS	↓	22	↓		↓	1	0	↓	■	45+9E
33	ZS*	↓	22	↓		↓	1	1	↓	■	45+9E
34	PD	↓	26	↓		↓	0	0	↓	▲	27+4.5L
34	PDO	↓	26	↓		↓	0	1	↓	▲	27+4.5L
32	ED	↓	52	↓		↓	1	0	↓	▲	36+13.5L+9E
27	CDa	↓	26	↓		↓	1	0/1	↓	▲	36+13.5Lm
15	SAa	↓	52	↓		↓	0	↓	↓	▲	27+9L
15	BAa	↓	56	↓		↓	0	↓	↓	▲	27+9L
15	BDa	↓	56	↓		↓	1	↓	↓	▲	31.5+9L
21	AMa	↓	62	↓		↓	0	↓	↓	▲	49.5+13.5L
21	SMa	↓	62	↓		↓	1	↓	↓	▲	49.5+13.5L
18	ADa	↓	66	↓		↓	0	↓	↓	▲	49.5+13.5(L+Lc)
20	SDa	↓	66	↓		↓	1	↓	↓	▲	49.5+13.5(L+Lc)
26	CBa	↓	70	↓		↓	1	↓	↓	▲	27+13.5L
22	ABa	↓	72	↓		↓	0	↓	↓	▲	27+13.5L
23	SBa	↓	72	↓		↓	1	↓	↓	▲	27+13.5L
35	TFR	M,,X	24	↓		↓	0	0	0	■	90+9B
35	TFI	↓	24	↓		↓	0	1	↓	■	103.5+9B
35	TTR	↓	24	↓		↓	1	0	↓	■	90+9B
35	TTI	↓	24	↓		↓	1	1	↓	■	103.5+9B
15	SAR	↓	52	↓		↓	1	1	↓	▲	315
24	MPN	,L	50	0		0	0	0	0 | 0-7		L(33.75K+27)+54
24	MPC	↓	50	↓		↓	0	1	↓ | ↓		L(33.75K+27)—27
25	DV	↓	50	↓		↓	1	0	↓ | ↓		4.5L(74.25K+13.75)+54
				X		M		n	S		
37	BCn	M,S,X	16	0-7		0-77777	1	0-3	0-7	▲	40.5+S(9+18n)
37	BSn	M,S,X	16	0-7		0-77777	0	0-3	0-7	▲	40.5+S(9+18n)
				X		M					
31	JR	M,I,X	10	0-7		0-77777	0-77		(I)	†	45
29	JC	M,I,X	30	↓		↓	↓		(I)	■	31.5
30	JL	M,N,X	32	↓		↓	↓		(N)	■	40.5
28	LC	M,C,X	14	↓		↓	↓		(C)	▲	40.5
27	CC	↓	34	↓		↓	↓		↓	▲	40.5
16	SC	↓	44	↓		↓	↓		↓	▲	40.5
36	LP	↓	54	↓		↓	↓		↓	▲	40.5
23	AC	↓	60	↓		↓	↓		↓	▲	45+13.5Lc
36	LS	↓	64	↓		↓	↓		↓	▲	40.5
17	FT	M,T,X	20	↓		↓	↓		(T)	▲	81
16	ST	↓	42	↓		↓	↓		↓	▲	63
16	BT	↓	46	↓		↓	↓		↓	▲	63
28	CT	↓	74	↓		↓	↓		↓	▲	81
23	AT	↓	76	↓		↓	↓		↓	▲	81
31	TR	M,L,X	12	↓		↓	↓		(L)	▲	36+13.5L
			29-25	24-22	21-18	17-12	11 - 0				
			Op-Code	X	U	F	D				
63	XF	F,D,U,X	40	0-7	0-17	0-77	0-7777				72**

*Operation Codes 00, 02, 04, 06 and 36 are unassigned.
**Not including loading print buffer.
† LSD of address portion of instruction referenced.

OPERATION 13 18	OPERANDS 19
TTI	DELTA

	ZETA	DELTA	Tetrads DST 16	ORG 17	TCT 18
Before:	JJS . . . MAH	xxx . . . xxx	xxxx	00001166	00000036
After:	(Same)	JJS . . . MAH	(Same)	00001224	(Same)
Timing:	103.5 + 9 B				

Where B is the number of characters transferred (Tetrad 18).

LOGICAL INSTRUCTIONS

Logical instructions examine two characters on a bit by bit basis and generate a new character in accordance with the specified truth table.

In logical instructions carries are not propagated.

The logical instructions are indexable.

LOGICAL SUM—LS M, C, X

OPERATION Performs a logical addition of the bits represented in C to the bits in the character specified by M_x. The result of the addition is stored in M_x.

Arithmetic Registers and indicators are not changed as a result of this operation.

The logical sum, also called the Inclusive OR, is defined by the table:

C \ M	0	1
0	0	1
1	1	1

EXAMPLE Form the logical sum of character 0 (000011) and the character at TAGA.

OPERATION 13 18	OPERANDS 19
LS	TAGA, '0 '

	TAGA
Before:	010101
After:	010111
Timing:	40.5

LOGICAL PRODUCT—LP M, C, X

OPERATION Performs a logical multiplication of the bits represented in C and the bits in the character specified by M_x. The result is stored in M_x.

Arithmetic Registers and indicators are not changed as a result of this operation. The logical product, also called logical AND, is defined by the following table:

C \ M	0	1
0	0	0
1	0	1

EXAMPLE Form the logical product of the character '&' (110011) and the character at TAGB.

OPERATION 13 18	19 OPERANDS
LP	TAGB, '&'

	TAGB
Before:	101111
After:	100011
Timing:	40.5

SHIFT INSTRUCTIONS

The shift instructions cause the left shift of a field of n (1-4) characters.

S represents the number of *bit* positions to be shifted. A maximum shift of 7 is possible.

The shift instructions are indexable.

BINARY SHIFT — BSn M, S, X

OPERATION Shifts n (from 1 to 4) characters at M_x. S *bit* positions from right to left.

The shifting may be considered as taking place in a theoretical shift register whose length is variable, and will be 6, 12, 18, or 24 bits as determined by n. Any bits which are shifted left beyond the most significant bit position of this register are dropped, and zeros replace the bits shifted out of the least significant positions of the register. After the shift, the result is stored in M_x.

EXAMPLE Perform a 3-bit shift of the 2-character field DATA3.

OPERATION 13 18	19 OPERANDS
BS2	DATA3, 3

		DATA3
Before:	110101	001111
After:	101001	111000

NOTE: Examples show binary representation of characters.

Timing: 40.5 + S (9 + 18 n)

BIT CIRCULATE — BCn M, S, X

OPERATION Shifts n (from 1-4) at M_x. S *bit* positions from right to left.

The shifting may be considered as taking place in a theoretical circular shift register whose length is variable, and will be 6, 12, 18 or 24 bits as determined by n. Any bits which are shifted left beyond the most significant bit position of this register are *not* dropped, but are entered into the least significant positions of the register. After the shift, the result is stored in M.

EXAMPLE Perform a 5-bit shift of the contents of the 3-character field DATA4.

OPERATION 13 18	19 OPERANDS
BC3	DATA4, 5

			DATA4
Before:	100110	110101	001111
After:	011010	100111	110011
Timing:	40.5 + S(9 + 18 n)		

ASSEMBLER DIRECTIVES

Assembler directives supply information to the PAL Assembly System. They are not assembled as object program instructions. The eight assembler directives are:

BEGIN **AREA**
END **PROC**
ORIG **NAME**
EQU **DO**

Each assembler directive requires at least one expression in the operands field; these are discussed with the individual directives. Where a label is referred to as an operands field entry, it must be

INSTRUCTION REPERTOIRE

PAGE 5-	WRITTEN OP-ERA-TION	WRITTEN OPER-ANDS	29-25 Op-code*	24-22 X	21	20....6 M	5	4	3 2 1 0 L	M Addr. LSD. ▲ MSD. ■	TIMING
33	ZS$	M,L,X	22	0-7	RESERVED	0-77777	0	0	0-17	■	49.5+9E
33	ZS		22				1	0		■	45+9E
33	ZS✱		22				1	1		■	45+9E
34	PD		26				0	0		▲	27+4.5L
34	PDO		26				0	1		▲	27+4.5L
32	ED		52				1	0		▲	36+13.5L+9E
27	CDa		26				1	0/1		▲	36+13.5Lm
15	SAa		52				0			▲	27+9L
15	BAa		56				0			▲	27+9L
15	BDa		56				1			▲	31.5+9L
21	AMa		62				0			▲	49.5+13.5L
21	SMa		62				1			▲	49.5+13.5L
18	ADa		66				0			▲	49.5+13.5(L+Lc)
20	SDa		66				1			▲	49.5+13.5(L+Lc)
26	CBa		70				1			▲	27+13.5L
22	ABa		72				0			▲	27+13.5L
23	SBa		72				1			▲	27+13.5L
35	TFR	M,,X	24				0	0	0	■	90+9B
35	TFI		24				0	1		■	103.5+9B
35	TTR		24				1	0		■	90+9B
35	TTI		24				1	1		■	103.5+9B
15	SAR		52				1	1		▲	315
24	MPN	,L	50	0		0	0	0	0 0-7		L(33.75K+27)+54
24	MPC		50				0	1			L(33.75K+27)—27
25	DV		50				1	0			4.5L(74.25K+13.75)+54
				X		M		n	S		
37	BCn	M,S,X	16	0-7		0-77777	1	0-3	0-7	▲	40.5+S(9+18n)
37	BSn	M,S,X	16	0-7		0-77777	0	0-3	0-7	▲	40.5+S(9+18n)
				X		M					
31	JR	M,I,X	10	0-7		0-77777		0-77	(I)	†	45
29	JC	M,I,X	30						(I)	■	31.5
30	JL	M,N,X	32						(N)	■	40.5
28	LC	M,C,X	14						(C)	▲	40.5
27	CC		34							▲	40.5
16	SC		44							▲	40.5
36	LP		54							▲	40.5
23	AC		60							▲	45+13.5Lc
36	LS		64							▲	40.5
17	FT	M,T,X	20						(T)	▲	81
16	ST		42							▲	63
16	BT		46							▲	63
28	CT		74							▲	81
23	AT		76							▲	81
31	TR	M,L,X	12						(L)	▲	36+13.5L
			29-25	24-22	21-18	17-12		11 - 0			
			Op-Code	X	U	F		D			
63	XF	F,D,U,X	40	0-7	0-17	0-77		0-7777			72**

*Operation Codes 00, 02, 04, 06 and 36 are unassigned.
**Not including loading print buffer.
† LSD of address portion of instruction referenced.

an already defined label (that is, it must have appeared previously as a label field entry).

Hereafter each entry in the symbolic coding form will be represented according to the type of expression that may be written for that entry:

n — an expression that must consist of at most, a single octal or decimal integer.

s — an expression that may consist of:

- ■ a single octal or decimal integer
- ■ the location counter or a symbol with a constant modifier (octal or decimal integer).

e — an arbitrary expression.

c — a single alphanumeric or special character.

t — a single character representing the type of information to be contained in an area. It must be one of the following:

A for alphanumeric information

B for binary information

I for instructions

p — may have the value 4 or 64.

Begin

Every program to be assembled must have BEGIN in the operation field of its first line.

The label field of this line should contain the program name. (This may require the use of column 12, which is accessible.)

The operands field must contain a single octal or decimal integer. The value of the integer determines the action taken by the assembler as:

INTEGER

DECIMAL	OCTAL	INTERPRETATION
1	01	The address of the first program line can be anywhere in computer memory.
2	02	The address of the first program line must be a multiple of four.
3	03	The address of the first program line must be a multiple of sixty-four.
any other non-negative integer		This expression represents the absolute address for the first line of the program.

As examples:

SEQUENCE PAGE 3	LINE 4 5	INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19
001	01		READ1	BEGIN	01

SEQUENCE PAGE 3	LINE 4 5	INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19
001	01		TAX	BEGIN	3

In the first example, the address assigned to the first object program line could be anywhere in the computer memory.

In the second example, the address of the first object program line would be a multiple of sixty-four.

End

Every program to be assembled must have END in the operation field of the last line to be assembled. END is also used with the PROC directive, see below.

For program end, the operands field must contain the label identifying (or the absolute address specifying) the first object program instruction that is to be executed; that is, the starting point of the program.

As an example:

OPERATION	OPERANDS
13 18	19
END	START

Here, the first program instruction had been labeled START. The address assigned to START is the address at which object program execution will begin.

Orig

When a program is divided into sections each of which is to be assembled relative to a different origin, an ORIG assembly directive is used. The occurrence of ORIG in the operations field of a line indicates the beginning of a section. The entry in the operands field indicates the address which is assigned to the first character of the section.

The operands field entry must be in one of the following forms:

OPERATION	OPERANDS
13 18	19
ORIG	s, p

OPERATION	OPERANDS
13 18	19
ORIG	s

where s is:

- a previously defined label field entry (this expression can include a constant modifier), in which case the first character of the section is assigned an address equal to the value of the expression.
- $, the current value of the location counters, is to be used for the address of the first character of the section.
- A decimal or octal integer to specify the absolute address in memory which the first character of the sections assigned.

p is either 4 or 64.

If p is used, the address for the first character of the section is the smallest number, not less than s, which is a multiple of p.

For example:

OPERATION	OPERANDS
13 18	19
ORIG	SUB01, 4

Here, the first line of the section will be assigned the lowest address which is a multiple of 4 and is not less than the address assigned to the line containing SUB01 in its label field.

Equ

An EQU assembler directive line must have in the label field an entry which is to be equated to the value of the expression or expressions in the operands field.

	LABEL	OPERATION	OPERANDS
S 6	7 11	13 18	19
	TAG01	EQU	s

	LABEL	OPERATION	OPERANDS
S 6	7 11	13 18	19
	TAG01	EQU	s, n

INSTRUCTION REPERTOIRE

PAGE 5-	WRITTEN OP-ERA-TION	 OPER-ANDS	INTERNAL FORM (OCTAL) BIT POSITIONS 29-25 Op-code*	 24-22 X	 21	 20....6 M	 5	 4	 3 2 1 0 L	M Addr. LSD. ▲ MSD. ■	TIMING
33	ZS$	M,L,X	22	0-7	RESERVED	0-77777	0	0	0-17	■	49.5+9E
33	ZS	↓	22	↓		↓	1	0	↓	■	45+9E
33	ZS✱	↓	22	↓		↓	1	1	↓	■	45+9E
34	PD	↓	26	↓		↓	0	0	↓	▲	27+4.5L
34	PDO	↓	26	↓		↓	0	1	↓	▲	27+4.5L
32	ED	↓	52	↓		↓	1	0	↓	▲	36+13.5L+9E
27	CDa	↓	26	↓		↓	1	0/1	↓	▲	36+13.5Lm
15	SAa	↓	52	↓		↓	0	↓	↓	▲	27+9L
15	BAa	↓	56	↓		↓	0	↓	↓	▲	27+9L
15	BDa	↓	56	↓		↓	1	↓	↓	▲	31.5+9L
21	AMa	↓	62	↓		↓	0	↓	↓	▲	49.5+13.5L
21	SMa	↓	62	↓		↓	1	↓	↓	▲	49.5+13.5L
18	ADa	↓	66	↓		↓	0	↓	↓	▲	49.5+13.5(L+Lc)
20	SDa	↓	66	↓		↓	1	↓	↓	▲	49.5+13.5(L+Lc)
26	CBa	↓	70	↓		↓	1	↓	↓	▲	27+13.5L
22	ABa	↓	72	↓		↓	0	↓	↓	▲	27+13.5L
23	SBa	↓	72	↓		↓	1	↓	↓	▲	27+13.5L
35	TFR	M,,X	24	↓		↓	0	0	0	■	90+9B
35	TFI	↓	24	↓		↓	0	1	↓	■	103.5+9B
35	TTR	↓	24	↓		↓	1	0	↓	■	90+9B
35	TTI	↓	24	↓		↓	1	1	↓	■	103.5+9B
15	SAR	↓	52	↓		↓	1	1	↓	▲	315
24	MPN	,L	50	0		0	0	0	0 0-7		L(33.75K+27)+54
24	MPC	↓	50	↓		↓	0	1	↓ ↓		L(33.75K+27)—27
25	DV	↓	50	↓		↓	1	0	↓ ↓		4.5L(74.25K+13.75)+54
				X		M		n	S		
37	BCn	M,S,X	16	0-7		0-77777	1	0-3	0-7	▲	40.5+S(9+18n)
37	BSn	M,S,X	16	0-7		0-77777	0	0-3	0-7	▲	40.5+S(9+18n)
				X		M					
31	JR	M,I,X	10	0-7		0-77777	0-77		(I)	†	45
29	JC	M,I,X	30	↓		↓	↓		(I)	■	31.5
30	JL	M,N,X	32	↓		↓	↓		(N)	■	40.5
28	LC	M,C,X	14	↓		↓	↓		(C)	▲	40.5
27	CC	↓	34	↓		↓	↓		↓	▲	40.5
16	SC	↓	44	↓		↓	↓		↓	▲	40.5
36	LP	↓	54	↓		↓	↓		↓	▲	40.5
23	AC	↓	60	↓		↓	↓		↓	▲	45+13.5Lc
36	LS	↓	64	↓		↓	↓		↓	▲	40.5
17	FT	M,T,X	20	↓		↓	↓		(T)	▲	81
16	ST	↓	42	↓		↓	↓		↓	▲	63
16	BT	↓	46	↓		↓	↓		↓	▲	63
28	CT	↓	74	↓		↓	↓		↓	▲	81
23	AT	↓	76	↓		↓	↓		↓	▲	81
31	TR	M,L,X	12	↓		↓	↓		(L)	▲	36+13.5L
			29-25	24-22	21-18	17-12	11 - 0				
			Op-Code	X	U	F	D				
63	XF	F,D,U,X	40	0-7	0-17	0-77	0-7777				72**

*Operation Codes 00, 02, 04, 06 and 36 are unassigned.
**Not including loading print buffer.
† LSD of address portion of instruction referenced.

where s is ■ a previously defined label field entry (this expression can include a constant modifier).

■ $, to designate the value of the location counter.

■ A decimal or octal integer to specify an absolute address.

n is an octal or decimal integer specifying a field length.

Normally, s denotes an address and n the length of a field whose least significant character is assigned that address.

The label field entry of an EQU line containing two expressions in the operands field has the value of those expressions supplied to any instruction that references that entry and does not contain an explicit statement of the length of the addressed field. Thus, given the EQU line:

	LABEL	OPERATION	OPERANDS
6	7 11	13 18	19
	DOG	EQU	CAT + 6, 3

the instruction

OPERATION	OPERANDS
13 18	19
BD1	DOG

is assembled as though it had been written

OPERATION	OPERANDS
13 18	19
BD1	CAT + 6, 3

Area

A storage area within a program can be defined by the use of the AREA assembler directive. The label field entry is assigned the address of the area's leftmost character. The operands field entry must be in one of the following forms:

OPERATION	OPERANDS
13 18	19
AREA	n
AREA	n, t
AREA	n, t, c

where **n** is a decimal or octal number specifying the size of the area.

t is the type of area and can be one of the following characters:

A for an alphanumeric data area.

B for a binary data area.

I for an area to contain instructions.

(Note: if the t *expression is omitted, the area is assumed to be alphanumeric.)*

c is any character except a comma (,) the value of which is to be placed in the area when the object program is loaded. That is, c represents a presetting for an area. The presetting character must occur in the first character position following the comma which terminates the t expression.

(Note: the area is represented on external storage only if the c *expression is used.)*

The Operands entry may also be:

OPERATION 13 18	OPERANDS 19
AREA	n, t, c, x
AREA	n, t, , x

The latter form is used if no presetting is required. x is a number from 1 to 7 inclusive and represents an Index Register. Each reference to a label of the area defined by this directive or to a label of a field within this area will cause the number specified by x to be inserted in the Index Register portion of the instruction containing the reference. This insertion can be prevented in any instruction by including an expression for the desired Index Register on the reference line.

If one writes:

LABEL 7 11	OPERATION 13 18	OPERANDS 19
SPACE	AREA	15,,,3
	BA1	SPACE, 1
	BA1	SPACE, 1, 0

The second line would result in a line with 3 in the index portion, and the third line would result in a line with 0 in the index portion. Thus, writing a symbol in the storage address portion of an instruction may cause the assembly routine to supply information to the storage address, length, and index portions.

Fields within an area can be defined in the lines immediately following an AREA line. This is accomplished by placing the continuation character in the operation field of the lines and using one of the following entries in the operands field:

OPERATION 13 18	OPERANDS 19
–	n_1
–	n_1, n_2

where $\mathbf{n}_1$ is a decimal or octal number specifying the length of the field.

$\mathbf{n}_2$ is the position of the field's rightmost character.

(Note: when a field's rightmost character is exactly n_1 *positions to the right of the last defined field,* n_2 *is not required.)*

A label field entry in a field defining line is assigned the address of the rightmost character of the field and specifies the field length.

As an example:

To define a 15-character area consisting of three fields as shown,

<table>
<tr><td colspan="6">F1</td><td colspan="9"></td></tr>
<tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td></tr>
<tr><td colspan="5"></td><td colspan="5">F2</td><td colspan="5">F3</td></tr>
</table>

the coding could be:

LABEL 7 11	OPERATION 13 18	OPERANDS 19
SPACE	AREA	15
F1	–	6
F2	–	5, 10
F3	–	5

Here, **SPACE** would be assigned the address of the leftmost character of the area being defined.

F1 would be assigned the address of the sixth character from the leftmost

INSTRUCTION REPERTOIRE

PAGE	WRITTEN		INTERNAL FORM (OCTAL) BIT POSITIONS							M Addr.	TIMING
			29-25	24-22	21	20....6	5	4	3 2 1 0	LSD. ▲	
5-	OP-ERATION	OPERANDS	Op-code*	X		M			L	MSD. ■	
33	ZS$	M,L,X	22	0-7	RESERVED	0-77777	0	0	0-17	■	49.5+9E
33	ZS	↓	22	↓		↓	1	0	↓	■	45+9E
33	ZS✱	↓	22	↓		↓	1	1	↓	■	45+9E
34	PD	↓	26	↓		↓	0	0	↓	▲	27+4.5L
34	PDO	↓	26	↓		↓	0	1	↓	▲	27+4.5L
32	ED	↓	52	↓		↓	1	0	↓	▲	36+13.5L+9E
27	CDa	↓	26	↓		↓	1	0/1	↓	▲	36+13.5Lm
15	SAa	↓	52	↓		↓	0	↓	↓	▲	27+9L
15	BAa	↓	56	↓		↓	0	↓	↓	▲	27+9L
15	BDa	↓	56	↓		↓	1	↓	↓	▲	31.5+9L
21	AMa	↓	62	↓		↓	0	↓	↓	▲	49.5+13.5L
21	SMa	↓	62	↓		↓	1	↓	↓	▲	49.5+13.5L
18	ADa	↓	66	↓		↓	0	↓	↓	▲	49.5+13.5(L+Lc)
20	SDa	↓	66	↓		↓	1	↓	↓	▲	49.5+13.5(L+Lc)
26	CBa	↓	70	↓		↓	1	↓	↓	▲	27+13.5L
22	ABa	↓	72	↓		↓	0	↓	↓	▲	27+13.5L
23	SBa	↓	72	↓		↓	1	↓	↓	▲	27+13.5L
35	TFR	M,,X	24	↓		↓	0	0	0	■	90+9B
35	TFI	↓	24	↓		↓	0	1	↓	■	103.5+9B
35	TTR	↓	24	↓		↓	1	0	↓	■	90+9B
35	TTI	↓	24	↓		↓	1	1	↓	■	103.5+9B
15	SAR	↓	52	↓		↓	1	1	↓	▲	315
24	MPN	,L	50	0		0	0	0	0 0-7		L(33.75K+27)+54
24	MPC	↓	50	↓		↓	0	1	↓ ↓		L(33.75K+27)—27
25	DV	↓	50	↓		↓	1	0	↓ ↓		4.5L(74.25K+13.75)+54
				X		M		n	S		
37	BCn	M,S,X	16	0-7		0-77777	1	0-3	0-7	▲	40.5+S(9+18n)
37	BSn	M,S,X	16	0-7		0-77777	0	0-3	0-7	▲	40.5+S(9+18n)
				X		M					
31	JR	M,I,X	10	0-7		0-77777	0-77		(I)	†	45
29	JC	M,I,X	30	↓		↓	↓		(I)	■	31.5
30	JL	M,N,X	32	↓		↓	↓		(N)	■	40.5
28	LC	M,C,X	14	↓		↓	↓		(C)	▲	40.5
27	CC	↓	34	↓		↓	↓		↓	▲	40.5
16	SC	↓	44	↓		↓	↓		↓	▲	40.5
36	LP	↓	54	↓		↓	↓		↓	▲	40.5
23	AC	↓	60	↓		↓	↓		↓	▲	45+13.5Lc
36	LS	↓	64	↓		↓	↓		↓	▲	40.5
17	FT	M,T,X	20	↓		↓	↓		(T)	▲	81
16	ST	↓	42	↓		↓	↓		↓	▲	63
16	BT	↓	46	↓		↓	↓		↓	▲	63
28	CT	↓	74	↓		↓	↓		↓	▲	81
23	AT	↓	76	↓		↓	↓		↓	▲	81
31	TR	M,L,X	12	↓		↓	↓		(L)	▲	36+13.5L
			29-25	24-22	21-18	17-12	11		0		
			Op-Code	X	U	F	D				
63	XF	F,D,U,X	40	0-7	0-17	0-77	0-7777				72**

*Operation Codes 00, 02, 04, 06 and 36 are unassigned.
**Not including loading print buffer.
† LSD of address portion of instruction referenced.

of the area (the rightmost of this field) and a field length of 6.

F2 would be assigned the address of the tenth character from the leftmost of the area and a field length of 5.

F3 would be assigned the address of the fifteenth character from the leftmost and a field length of 5.

Proc

In the **PAL** Assembly System a list of basic macro-instructions has been provided. These allow some functions of general utility to be included in an object program with maximum ease. However, it is likely that other general functions will be recognized (perhaps peculiar to an installation) which it is desirable to incorporate in the program with similar ease.

To facilitate the process of incorporating such functions, the **PAL** Assembly System includes the ability to handle procedures. A procedure is a piece of symbolic coding with possible variations and with one or more names by which it can be referenced and inserted in the object program. The coding generated by a procedure is placed in the object program at the place corresponding to a place where a reference to the procedure appears in the source program.

The procedure begins with a line of the form:

	LABEL		OPERATION		OPERANDS
NS 6	7	11	13	18	19
	LBL		PROC		PAR

The operation field of this line contains the assembler directive PROC. This indicates that the coding that follows up to the next END line is part of a procedure. These lines are to be included, conditionally, when the procedure is called. The label field of the PROC line contains an acceptable name by which the procedure may be called. This symbol is also used to denote the parameters written in the reference line. Thus the fifth parameter in the call for LBL would be denoted by LBL(5). A procedure can be called by more than one name (see NAME below). The operand field of the PROC line contains a value that will be supplied as parameter 0 if the procedure is called by the name in the label field. Thus if the procedure with the PROC line shown above were called by the name LBL, LBL(0) written in the coding within the procedure would be replaced by the value PAR. LBL written in the coding would be replaced by the number of parameters in the calling line. A procedure must not contain a macro reference.

Name

Immediately following the PROC line of a procedure there can appear one or more lines with NAME in the operation field. The label field of each such line contains an acceptable name by which the procedure may be called. The operand portion contains a parameter to be supplied as parameter zero when the routine is called by this name. Thus a procedure to calculate either the maximum or the minimum of a set of numbers might begin:

NS	LABEL	OPERATION	OPERANDS
6	7 11	13 18	19
	MAX	PROC	0
	MIN	NAME	1

If this routine is called by a line with MAX in the operation field, the value of the parameter MAX(0) is 0. If it is called by a line with MIN in the operation field, the value of MAX(0) is 1. In either case, the value of MAX is the number of parameters written on the reference line.

The last of these lines is followed by the assembly directive END.

Do

A set of lines can be generated several times in succession. This is accomplished by writing:

NS	LABEL	OPERATION	OPERANDS
6	7 11	13 18	19
	L	DO	e_1, e_2

This line instructs the assembler to put out the following e_2 lines, e_1 times. If the DO statement has a label, the label has no relation to the location of the DO statement in the program. For the lines of coding in the range of the DO statement (that is, the next e_2 lines), the symbol appearing as the label of the DO statement has the value i in the ith repetition of the coding pattern. There can be no unconditional DO's within the lines specified by e_2.

Thus a line of coding of the form:

OPERATION	OPERANDS
13 18	19
BA1	S (L + 2)

where S(L+2) represent the (L+2)th parameter of the procedure in which the above line appears and L is the label of the DO statement generating the above line, would be equivalent to:

OPERATION	OPERANDS
13 18	19
BA1	S (3)

in the first copy of the coding:

OPERATION	OPERANDS
13 18	19
BA1	S (4)

in the second copy, and so on.

The first operand (e_1) in a DO statement can involve another type of expression. A line containing this expression has the form

OPERATION	OPERANDS
13 18	19
DO	e_3Re_4, e_2

where e_3 and e_4 are expressions and R is one of the following:

$>$ (e_3 greater than e_4)
$<$ (e_3 less than e_4)
$=$ (e_3 equal to e_4)
$\neq$ (e_3 unequal to e_4)

The value of the expression is 1 if the relation is satisfied, 0 otherwise. e_3 and e_4 must be expressions involving only octal, decimal or alphanumeric items and parameters of the procedure.

As an example, for a procedure to produce coding to place in arithmetic register 1 either the smallest or the largest of an arbitrary number of decimal values. The reference line has the form

OPERATION	OPERANDS
13 18	19
MAX	$n, P_1, \ldots, P_k$

INSTRUCTION REPERTOIRE

PAGE 5-	WRITTEN		INTERNAL FORM (OCTAL) BIT POSITIONS							M Addr.	TIMING
			29-25	24-22	21	20....6	5	4	3 2 1 0	LSD. ▲ MSD. ■	
	OP-ERA-TION	OPER-ANDS	Op-code*	X		M			L		
33	ZS$	M,L,X	22	0-7	RESERVED	0-77777	0	0	0-17	■	49.5+9E
33	ZS	↓	22	↓		↓	1	0	↓	■	45+9E
33	ZS✱	↓	22	↓		↓	1	1	↓	■	45+9E
34	PD	↓	26	↓		↓	0	0	↓	▲	27+4.5L
34	PDO	↓	26	↓		↓	0	1	↓	▲	27+4.5L
32	ED	↓	52	↓		↓	1	0	↓	▲	36+13.5L+9E
27	CDa	↓	26	↓		↓	1	0/1	↓	▲	36+13.5Lm
15	SAa	↓	52	↓		↓	0	↓	↓	▲	27+9L
15	BAa	↓	56	↓		↓	0	↓	↓	▲	27+9L
15	BDa	↓	56	↓		↓	1	↓	↓	▲	31.5+9L
21	AMa	↓	62	↓		↓	0	↓	↓	▲	49.5+13.5L
21	SMa	↓	62	↓		↓	1	↓	↓	▲	49.5+13.5L
18	ADa	↓	66	↓		↓	0	↓	↓	▲	49.5+13.5(L+Lc)
20	SDa	↓	66	↓		↓	1	↓	↓	▲	49.5+13.5(L+Lc)
26	CBa	↓	70	↓		↓	1	↓	↓	▲	27+13.5L
22	ABa	↓	72	↓		↓	0	↓	↓	▲	27+13.5L
23	SBa	↓	72	↓		↓	1	↓	↓	▲	27+13.5L
35	TFR	M,,X	24	↓		↓	0	0	0	■	90+9B
35	TFI	↓	24	↓		↓	0	1	↓	■	103.5+9B
35	TTR	↓	24	↓		↓	1	0	↓	■	90+9B
35	TTI	↓	24	↓		↓	1	1	↓	■	103.5+9B
15	SAR	↓	52	↓		↓	1	1	↓	▲	315
24	MPN	,L	50	0		0	0	0	0 0-7		L(33.75K+27)+54
24	MPC	↓	50	↓		↓	0	1	↓ ↓		L(33.75K+27)—27
25	DV	↓	50	↓		↓	1	0	↓ ↓		4.5L(74.25K+13.75)+54
				X		M		n	S		
37	BCn	M,S,X	16	0-7		0-77777	1	0-3	0-7	▲	40.5+S(9+18n)
37	BSn	M,S,X	16	0-7		0-77777	0	0-3	0-7	▲	40.5+S(9+18n)
				X		M					
31	JR	M,I,X	10	0-7		0-77777		0-77	(I)	†	45
29	JC	M,I,X	30	↓		↓		↓	(I)	■	31.5
30	JL	M,N,X	32	↓		↓		↓	(N)	■	40.5
28	LC	M,C,X	14	↓		↓		↓	(C)	▲	40.5
27	CC	↓	34	↓		↓		↓	↓	▲	40.5
16	SC	↓	44	↓		↓		↓	↓	▲	40.5
36	LP	↓	54	↓		↓		↓	↓	▲	40.5
23	AC	↓	60	↓		↓		↓	↓	▲	45+13.5Lc
36	LS	↓	64	↓		↓		↓	↓	▲	40.5
17	FT	M,T,X	20	↓		↓		↓	(T)	▲	81
16	ST	↓	42	↓		↓		↓	↓	▲	63
16	BT	↓	46	↓		↓		↓	↓	▲	63
28	CT	↓	74	↓		↓		↓	↓	▲	81
23	AT	↓	76	↓		↓		↓	↓	▲	81
31	TR	M,L,X	12	↓		↓		↓	(L)	▲	36+13.5L
			29-25	24-22	21-18	17-12	11 - 0				
			Op-Code	X	U	F	D				
63	XF	F,D,u,X	40	0-7	0-17	0-77	0-7777				72**

*Operation Codes 00, 02, 04, 06 and 36 are unassigned.
**Not including loading print buffer.
† LSD of address portion of instruction referenced.

or

OPERATION 13 18	OPERANDS 19
MIN	n, P_1, . . . , P_k

where it is desired to find the largest (or smallest, respectively) of the n-digit decimal numbers,

$P_1, \ldots . P_k$

The procedure is written as follows:

LABEL 7 11	OPERATION 13 18	OPERANDS 19
MAX	PROC	0
MIN	NAME	1
	BD1	MAX (2), MAX (1)
L	DO	MAX - 2, 4
	CD1	MAX (L + 2), MAX (1)
	DO	MAX (0) = 0, 1
	JC	$ + 10, 33
	DO	MAX (0) = 1, 1
	JC	$ + 10, 36
	BD1	MAX (L + 2), MAX (1)

If the reference line is:

OPERATION 13 18	OPERANDS 19
MAX	5, X, X + 5, Y

Then the following coding will be generated

OPERATION 13 18	OPERANDS 19
BD1	X,5
CD1	X + 5, 5
JC	$ + 10, 33
BD1	X + 5, 5
CD1	Y, 5
JC	$ + 10, 33
BD1	Y, 5

DATA GENERATION

Constant Data

A constant of specified length can be generated by the operation of field entries +n or −n where n is a decimal number. The label field of such a line can contain an entry. The operands field must contain a single expression specifying the content to be generated for the field length specified by +n or −n. If the value of the expression is an integer of less than n characters (for example, if n were 4 and the value of the expression was one character the value of the expression is stored right justified). Thus, given the coded line:

OPERATION 13 18	OPERANDS 19
+4	‘432’

the constant would be treated as alphanumeric and stored as:

CHARACTER 1	CHARACTER 2	CHARACTER 3	CHARACTER 4	Field defined in OPERATION field.
000000	000111	000110	000101	Constant stored

When the operands field expression is alphanumeric and the sign in the operation field is negative, the sign bit of the constant stored is reversed. Thus,

CONSTANT STORED

000111

OPERATION 13 18	OPERANDS 19
+1	‘4’

100111

OPERATION 13 18	OPERANDS 19
-1	‘4’

(This sign bit of a character is the most significant bit position. The sign of a field is found in the least significant character of the field.)

When the operand field expression is octal, or decimal to binary, and the sign in the operations field is negative, the two's complement of the expression value is stored. Thus, as further examples,

CONSTANT STORED

000101 000110 100111

OPERATION 13 18	OPERANDS 19
-3	‘234’

000000 000011 101010

OPERATION 13 18	OPERANDS 19
+3	234

000010 011100

OPERATION 13 18	OPERANDS 19
+2	0234

111101 100100

OPERATION 13 18	OPERANDS 19
-2	0234

When the operands field expression is a label (an entry that has occurred in a label field) unmodified or with a constant modifier, and n in the operation field is:

1 — the length portion of the label's definition is supplied.

3 — the address portion of the label's definition is supplied.

4 — both the address and length portion of the label is supplied.

Arbitrary Data

Data can be stored in any desired arbitrary form by the use of the assembler operator FORM in the operations field of a line. The operands field of this line contains the format in which the data is to be stored. The specified format is applied to the operands field entries of succeeding lines that refer to the FORM line.

The FORM line is written

OPERATION 13 18	OPERANDS 19
FORM	$n_1, n_2 . . . , n_k$

where $1 \leq k \leq 8$.

(That is, the operands field must contain one entry and can contain up to eight entries.)

where n_i is a decimal number not greater than 48, a T, or an X. The total of the decimal numbers, plus 6 for each T and 3 for each X used, must not exceed 96 and must be a multiple of 6.

INSTRUCTION REPERTOIRE

PAGE 5-	WRITTEN		INTERNAL FORM (OCTAL) BIT POSITIONS							M Addr.	TIMING
	OP-ERA-TION	OPER-ANDS	29-25	24-22	21	20....6	5	4	3 2 1 0	LSD. ▲	
			Op-code*	X		M			L	MSD. ■	
33	ZS$	M,L,X	22	0-7	RESERVED	0-77777	0	0	0-17	■	49.5+9E
33	ZS		22				1	0		■	45+9E
33	ZS*		22				1	1		■	45+9E
34	PD		26				0	0		▲	27+4.5L
34	PDO		26				0	1		▲	27+4.5L
32	ED		52				1	0		▲	36+13.5L+9E
27	CDa		26				1	0/1		▲	36+13.5Lm
15	SAa		52				0			▲	27+9L
15	BAa		56				0			▲	27+9L
15	BDa		56				1			▲	31.5+9L
21	AMa		62				0			▲	49.5+13.5L
21	SMa		62				1			▲	49.5+13.5L
18	ADa		66				0			▲	49.5+13.5(L+Lc)
20	SDa		66				1			▲	49.5+13.5(L+Lc)
26	CBa		70				1			▲	27+13.5L
22	ABa		72				0		↓	▲	27+13.5L
23	SBa	↓	72				1	↓		▲	27+13.5L
35	TFR	M,,X	24				0	0	0	■	90+9B
35	TFI		24				0	1		■	103.5+9B
35	TTR		24				1	0		■	90+9B
35	TTI	↓	24				1	1	↓	■	103.5+9B
15	SAR		52	↓		↓	1	1		▲	315
24	MPN	,L	50	0		0	0	0	0 0-7		L(33.75K+27)+54
24	MPC		50				0	1			L(33.75K+27)—27
25	DV	↓	50	↓		↓	1	0	↓ ↓		4.5L(74.25K+13.75)+54
				X		M		n	S		
37	BCn	M,S,X	16	0-7		0-77777	1	0-3	0-7	▲	40.5+S(9+18n)
37	BSn	M,S,X	16	0-7		0-77777	0	0-3	0-7	▲	40.5+S(9+18n)
				X		M					
31	JR	M,I,X	10	0-7		0-77777		0-77	(I)	†	45
29	JC	M,I,X	30						(I)	■	31.5
30	JL	M,N,X	32						(N)	■	40.5
28	LC	M,C,X	14						(C)	▲	40.5
27	CC		34							▲	40.5
16	SC		44							▲	40.5
36	LP		54							▲	40.5
23	AC		60							▲	45+13.5Lc
36	LS	↓	64						↓	▲	40.5
17	FT	M,T,X	20						(T)	▲	81
16	ST		42							▲	63
16	BT		46							▲	63
28	CT		74							▲	81
23	AT	↓	76						↓	▲	81
31	TR	M,L,X	12	↓		↓		↓	(L)	▲	36+13.5L
			29-25	24-22	21-18	17-12	11 - 0				
			Op-Code	X	U	F	D				
63	XF	F,D,U,X	40	0-7	0-17	0-77	0-7777				72**

*Operation Codes 00, 02, 04, 06 and 36 are unassigned.
**Not including loading print buffer.
† LSD of address portion of instruction referenced.

Each n_1 expression specifies how many bit positions are to be allocated to hold the value of a corresponding expression in a succeeding line. A succeeding line to which a FORM line is to apply must contain:

1. *The LABEL entry of the FORM line in its OPERATION field.*
2. *As many expressions in the OPERANDS field as are indicated by the number of expressions in the FORM line OPERANDS field.*

The characters T and X specify special evaluation of values. T specifies six bit positions; X specifies three bit positions. In addition, when the corresponding expression on a succeeding line is a label,

T specifies that the value of the defined label (if a symbol) is to have three (3) subtracted from it and the remainder divided by four (4) before storage in six bit positions.

X specifies that the value of the defined label (if a symbol) have thirty-five (35) subtracted from it and the remainder divided by four (4).

For example, it may be desired to communicate with a subroutine which is entered with a JR instruction and requires, in the line following the JR, a parameter occupying 5 characters with a tetrad address in the first character and an address in the next three characters and a length in the least significant three bits of the last character.

Writing references to such a subroutine could be simplified by writing a FORM line:

NS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19
	PARL	FORM	T, 18, 6
		JR	SUBR
		PARL	X1, COST, 5

Where X1 is assumed to have been elswhere defined as the address of index register 1. If COST has the octal value 236 the PARL line in the above example would result in the octal output

1100023605

since 11 is the octal value for the Tetrad address of index register 1.

MACRO-INSTRUCTIONS

A set of basic macro-instructions is provided with the **PAL** Assembly System. Each such macro represents a section of coding performing a function of general utility. This coding can be entered into the object program by writing the name of the macro in the operation field of a line at each place in the program at which it is desired to perform this function. The operands field of each such reference contains the parameters describing the particular form and function presently desired.

Input-Output Macro-Instructions

These macro-instructions represent linkages to the CALL routines which provide functions tailored to the specific file formats and processing indicated in the specifications to the CALL routines. They provide for:

a. *getting the next input record from a tape file,*
b. *writing the next output record on a tape file,*
c. *initializing and terminating the processing of a tape file.*

GET

The next input record to be processed is obtained by use of the macro-instruction GET. Use of the GET macro-instruction results in the automatic reading of another block of records of this file from tape whenever all the records in an input area have been processed. The GET macro-instruction has either of the following forms:

OPERATION 13 18	OPERANDS 19
GET	FLA

OPERATION 13 18	OPERANDS 19
GET	FLA, WKAR

The parameter FLA represents the name of an input file, WKAR is the name of working storage area large enough to hold the largest record of FLA. The first of the above two forms places the address of the next item of FLA in an Index Register as specified on the CALL form carrying the specifications of processing for FLA. The second form of the GET macro-instruction will move the next input record into the area named WKAR.

PUT

The next output record to be written is delivered to the output file by use of the macro-instruction PUT. If more than one record is to be included in an output block they are collected in an output area until enough records have been accumulated tơ make up an output block and then written on tape. This function is accomplished automatically through the use of the PUT macro-instruction. The instruction can be written in one of the following ways:

OPERATION 13 18	OPERANDS 19
PUT	FLA, FLB

The form above causes the current record in FLA, which must be an input file, to be included as the next record in FLB, which must be an output file. The next record of FLA must be obtained by use of the GET macro-instruction.

OPERATION 13 18	OPERANDS 19
PUT	WKAR, FLB

This form results in the inclusion of the record in the area labeled WKAR in the output file, FLB.

OPERATION 13 18	OPERANDS 19
PUT	FLB

This form is used when processing is done in the output area of FLB. The current record area in FLB is addressed through an Index Register given in the specifications for the processing of FLB. The PUT macro-instruction causes the setting of this Index Register to be altered to address the next record area. A block is written when appropriate.

OPEN

Before beginning to use an input or output file, the file must be initialized for processing by use of the macro-instruction OPEN. OPEN causes checking or writing of appropriate header labels.

OPERATION 13 18	OPERANDS 19
OPEN	FLA, FLB

INSTRUCTION REPERTOIRE

PAGE 5-	WRITTEN		INTERNAL FORM (OCTAL) BIT POSITIONS							M Addr.	TIMING
			29-25	24-22	21	20....6	5	4	3 2 1 0	LSD. ▲	
	OPERATION	OPERANDS	Op-code*	X		M			L	MSD. ■	
33	ZS$	M,L,X	22	0-7	RESERVED	0-77777	0	0	0-17	■	49.5+9E
33	ZS	↓	22	↓		↓	1	0	↓	■	45+9E
33	ZS✱	↓	22	↓		↓	1	1	↓	■	45+9E
34	PD	↓	26	↓		↓	0	0	↓	▲	27+4.5L
34	PDO	↓	26	↓		↓	0	1	↓	▲	27+4.5L
32	ED	↓	52	↓		↓	1	0	↓	▲	36+13.5L+9E
27	CDa	↓	26	↓		↓	1	0/1	↓	▲	36+13.5Lm
15	SAa	↓	52	↓		↓	0	↓	↓	▲	27+9L
15	BAa	↓	56	↓		↓	0	↓	↓	▲	27+9L
15	BDa	↓	56	↓		↓	1	↓	↓	▲	31.5+9L
21	AMa	↓	62	↓		↓	0	↓	↓	▲	49.5+13.5L
21	SMa	↓	62	↓		↓	1	↓	↓	▲	49.5+13.5L
18	ADa	↓	66	↓		↓	0	↓	↓	▲	49.5+13.5(L+Lc)
20	SDa	↓	66	↓		↓	1	↓	↓	▲	49.5+13.5(L+Lc)
26	CBa	↓	70	↓		↓	1	↓	↓	▲	27+13.5L
22	ABa	↓	72	↓		↓	0	↓	↓	▲	27+13.5L
23	SBa	↓	72	↓		↓	1	↓		▲	27+13.5L
35	TFR	M,,X	24	↓		↓	0	0	0	■	90+9B
35	TFI	↓	24	↓		↓	0	1	↓	■	103.5+9B
35	TTR	↓	24	↓		↓	1	0	↓	■	90+9B
35	TTI	↓	24	↓		↓	1	1	↓	■	103.5+9B
15	SAR	↓	52	↓		↓	1	1	↓	▲	315
24	MPN	,L	50	0		0	0	0	0 0-7		L(33.75K+27)+54
24	MPC	↓	50	↓		↓	0	1	↓ ↓		L(33.75K+27)—27
25	DV	↓	50	↓		↓	1	0	↓ ↓		4.5L(74.25K+13.75)+54
				X		M		n	S		
37	BCn	M,S,X	16	0-7		0-77777	1	0-3	0-7	▲	40.5+S(9+18n)
37	BSn	M,S,X	16	0-7		0-77777	0	0-3	0-7	▲	40.5+S(9+18n)
				X		M					
31	JR	M,I,X	10	0-7		0-77777	0-77		(I)	†	45
29	JC	M,I,X	30	↓		↓	↓		(I)	■	31.5
30	JL	M,N,X	32	↓		↓	↓		(N)	■	40.5
28	LC	M,C,X	14	↓		↓	↓		(C)	▲	40.5
27	CC	↓	34	↓		↓	↓		↓	▲	40.5
16	SC	↓	44	↓		↓	↓		↓	▲	40.5
36	LP	↓	54	↓		↓	↓		↓	▲	40.5
23	AC	↓	60	↓		↓	↓		↓	▲	45+13.5Lc
36	LS	↓	64	↓		↓	↓		↓	▲	40.5
17	FT	M,T,X	20	↓		↓	↓		(T)	▲	81
16	ST	↓	42	↓		↓	↓		↓	▲	63
16	BT	↓	46	↓		↓	↓		↓	▲	63
28	CT	↓	74	↓		↓	↓		↓	▲	81
23	AT	↓	76	↓		↓	↓		↓	▲	81
31	TR	M,L,X	12	↓		↓	↓		(L)	▲	36+13.5L
			29-25	24-22	21-18	17-12	11		0		
			Op-Code	X	U	F	D				
63	XF	F,D,U,X	40	0-7	0-17	0-77	0-7777				72**

*Operation Codes 00, 02, 04, 06 and 36 are unassigned.
**Not including loading print buffer.
† LSD of address portion of instruction referenced.

It does not deliver an input record. However, for an output file with processing in the output area, the address of the first area is delivered to the appropriate index register. A single OPEN can open several files. The operand portion of the instruction line contains the name of each of the files to be opened at this point in the program.

CLOSE

At the completion of an output file, the writing of the file is terminated by executing a CLOSE macro-instruction. This instruction causes the following action: any partial block of output data that has not yet been written is now written, a sentinel block (or trailer label) is written. A single CLOSE can close several files. The operand portion of the instruction line contains the name of each of the files to be closed at this point.

OPERATION 13 18	OPERANDS 19
CLOSE	FLA, FLB

Any of the above macro-instructions can have a symbol in the label field. This symbol will address the first line of coding generated by the macro.

Diagnostic Macro-instructions

SNAP

This macro-instruction provides entry to the routine DUMP to print a specified portion of storage in a specified mode.

DUMP

This macro-instruction causes inclusion in the object program of a routine to print a portion of storage. It can be entered from various points in the program. The form of the macro-instruction may be any of the following:

OPERATION 13 18	OPERANDS 19
DUMP	t,a,b

OPERATION 13 18	OPERANDS 19
DUMP	t

OPERATION 13 18	OPERANDS 19
DUMP	

In the first form, a and b represent the lowest and highest addresses to be included in the printing. All locations from a to b, inclusive, will be represented.

The mode of representation is indicated by the parameter, t, which is:

A *for alphanumeric, and*

B *if the area to be represented is binary. In this case, the interpretation will be in octal format.*

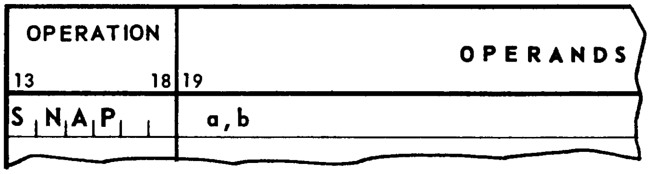
I *for instruction representation.*

The parameter t can be left blank in which case the information will be presented in both alphanumeric and octal formats.

The form of the SNAP macro-instruction used with this form of DUMP must have a blank operands field.

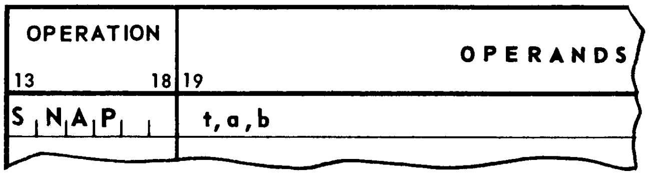
The second and third forms provide for printing any portion of memory as specified by the SNAP macro-instruction providing entrance to the DUMP routine.

The second form of DUMP reference provides information as to the form of the information to be printed. Thus, t may be A, B or I to provide for printing information in a single form, or it may be IA, IB, or AB to provide a facility for printing the corresponding kinds of information. If the ability to print all three kinds is desired the operand field can be left blank. If t is A, B or I, the form of the SNAP macro-instruction must be



OPERATION 13 18	OPERANDS 19
SNAP	a,b

If the third form of DUMP is used or if t is IA, IB, or AB, then the SNAP macro-instruction to be used has the form



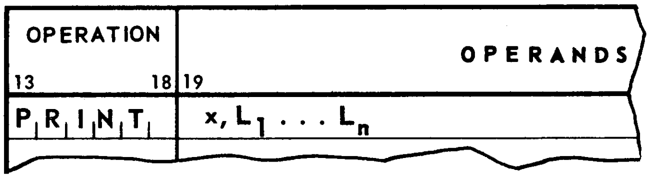
OPERATION 13 18	OPERANDS 19
SNAP	t,a,b

where t, a, and b have the same meaning as for the DUMP macro-instruction.

PRINT

This macro-instruction causes the printing of certain calculated results together with the label of the field containing each such result. The printing for each such field can be made conditional upon the result being different from an expected value.

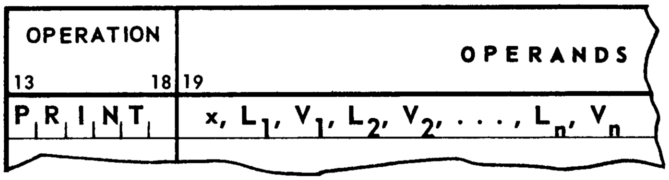
If it is desired to print every time this routine is entered, the form of the macro-instruction is



OPERATION 13 18	OPERANDS 19
PRINT	$x, L_1 \ldots L_n$

where x has the value E. The form of the output for each such field is L_i followed by the computed result.

If it is desired to print only those values which are different from an expected value, the form of the macro-instruction is

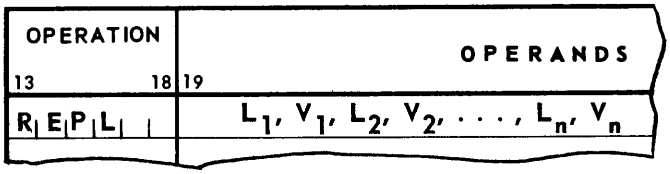


OPERATION 13 18	OPERANDS 19
PRINT	$x, L_1, V_1, L_2, V_2, \ldots, L_n, V_n$

where x has the value D. L_i is the expression representing the address of the ith field to be printed if the computed result differs from the expected value V_i. The information for each field is printed in the order L_i, computed result, V_i.

REPL

This macro-instruction allows the programmer to replace certain computed results with predicted results and thus proceed to test portions of a program beyond one that is not yet functioning properly. This enables one program test run to uncover several program errors and provides more efficient use of the computer for debugging. The form of the macro-instruction is:



OPERATION 13 18	OPERANDS 19
REPL	$L_1, V_1, L_2, V_2, \ldots, L_n, V_n$

where L_i is the expression representing the address of the ith field and V_i is the expected value of that field.

SYMBOLIC LISTING:

The printed output of the PAL Assembly System includes the following elements: memory map, symbol table, and symbolic listing. The memory map lists the address bounds of the various program areas by type (instructions, decimal data, or binary data). The listing of the symbol table includes the symbol, type of field represented, length (unless it is an area), and the address assigned to the symbol. The symbolic listing includes columns 1 to 72 of the input card on the right-hand side of the listing, thus providing a record copy of hand coding used to prepare the symbolic coding.

The left-hand side has the following forms:

DATA WORD LINES

The octal address, followed by up to 16 alphanumeric characters representing the data itself.

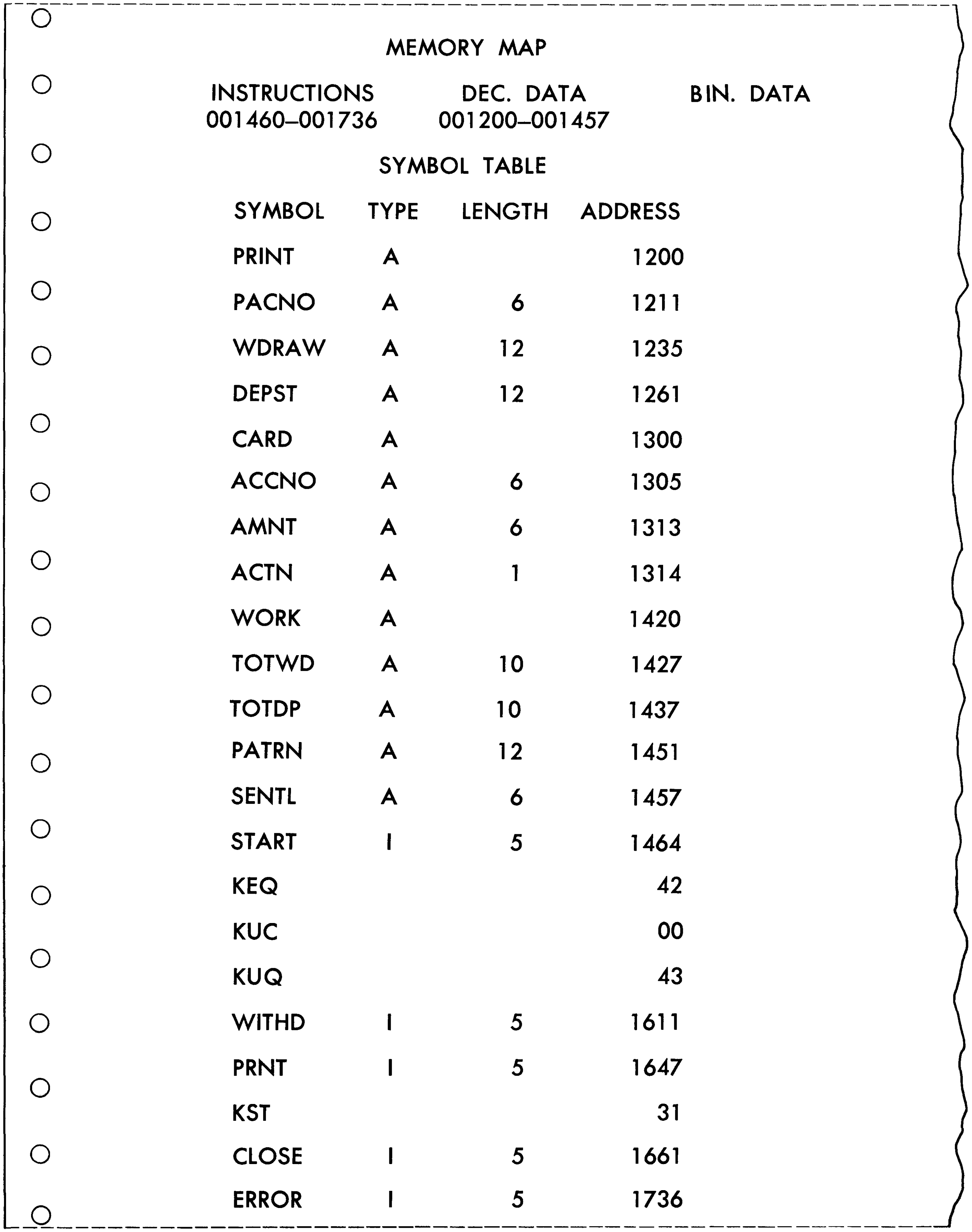
INSTRUCTION WORD LINES

The octal address of the least significant character of the instruction followed by the instruction itself. For instructions other than the XF instruction, this includes the operation code in two octal digits, the Index Register reference in one digit, the octal storage address in six digits, the operation code expansion digit, and two decimal digits for the last field of the instruction (which may represent a Tetrad address, operand length, indicator, etc., depending on the instruction type). For the XF instruction, the fields are octal operation code, channel, unit, function, and detail. The last two fields are presented in octal.

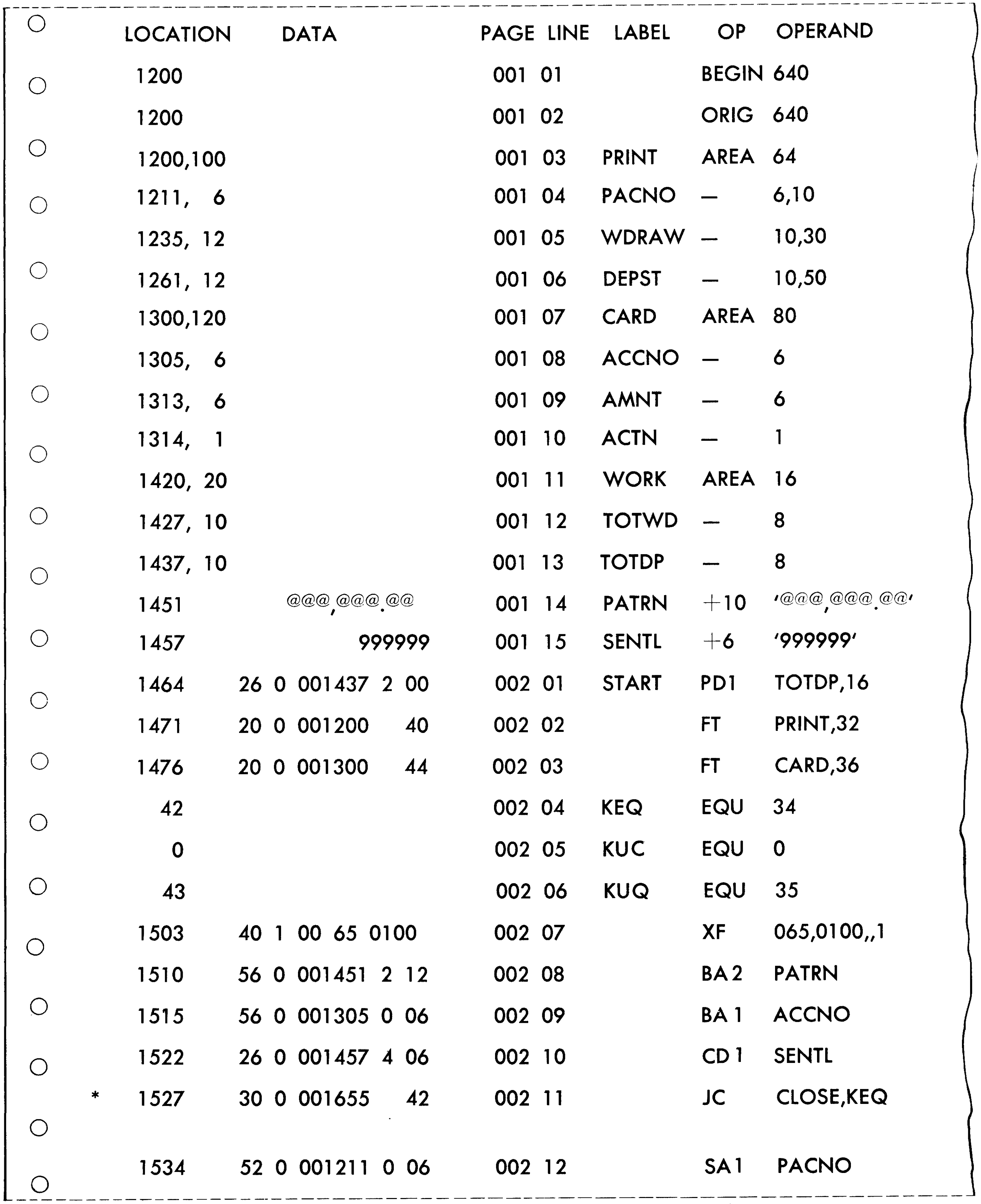
ASSEMBLY DIRECTIVES

The left hand side of the form contains the evaluation of the expression in the operands field of the directive BEGIN. For the directives EQU, ORIG, and AREA, the address field contains the value of the expression in the operands field. For field definitions the address field contains the address assigned to the least significant character of the field and is followed by the field length.

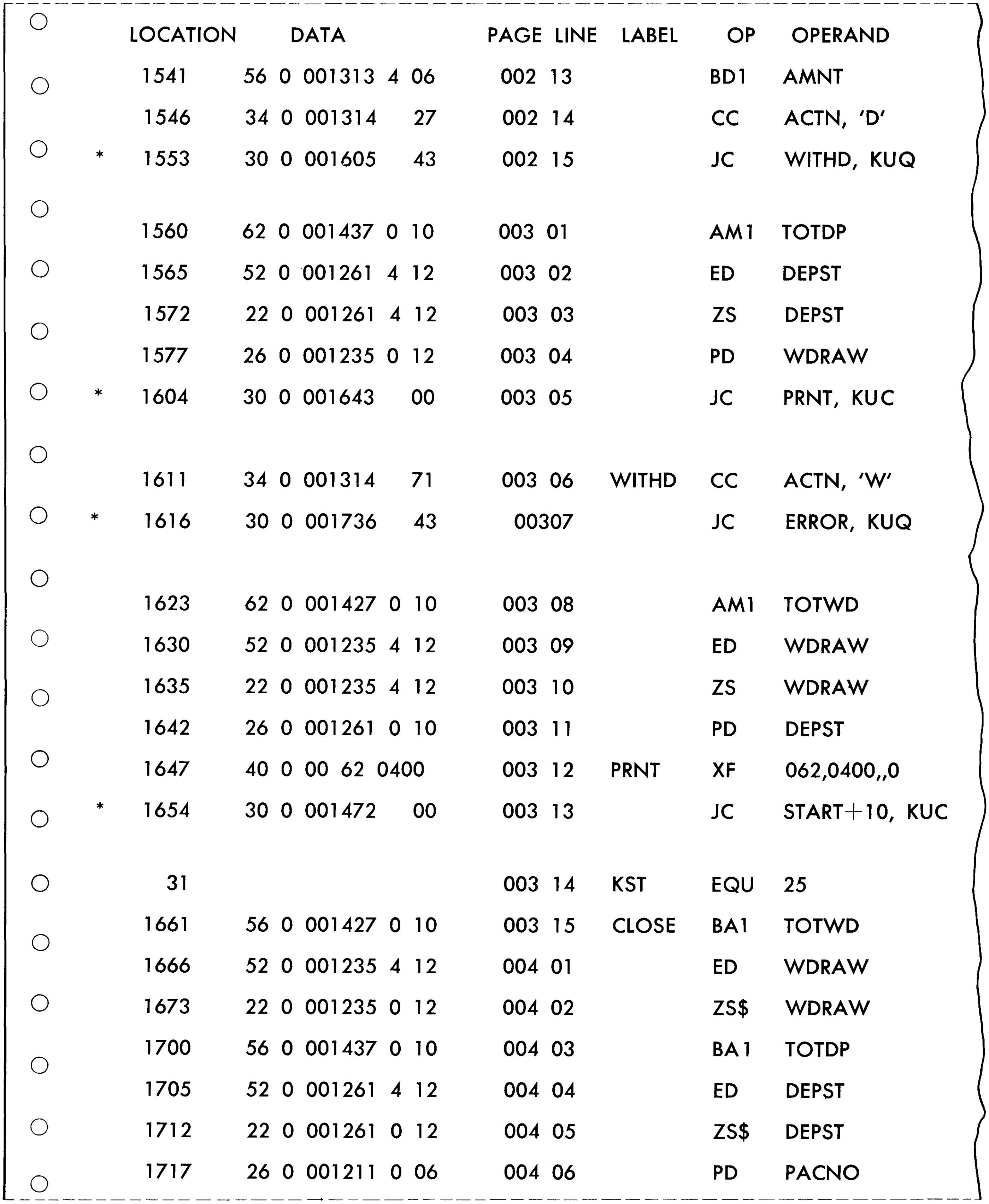
A sample of the various parts of the listing follows. Departures from the normal program flow are marked with an asterisk.



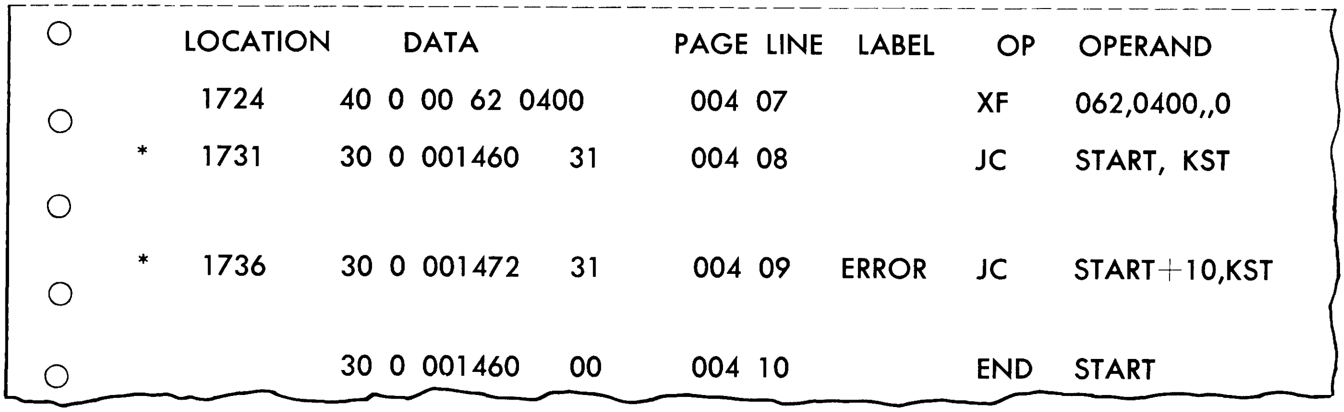
```
                        MEMORY MAP

         INSTRUCTIONS        DEC. DATA          BIN. DATA
         001460-001736       001200-001457

                       SYMBOL TABLE

          SYMBOL    TYPE    LENGTH    ADDRESS

          PRINT      A                  1200
          PACNO      A         6        1211
          WDRAW      A        12        1235
          DEPST      A        12        1261
          CARD       A                  1300
          ACCNO      A         6        1305
          AMNT       A         6        1313
          ACTN       A         1        1314
          WORK       A                  1420
          TOTWD      A        10        1427
          TOTDP      A        10        1437
          PATRN      A        12        1451
          SENTL      A         6        1457
          START      I         5        1464
          KEQ                             42
          KUC                             00
          KUQ                             43
          WITHD      I         5        1611
          PRNT       I         5        1647
          KST                             31
          CLOSE      I         5        1661
          ERROR      I         5        1736
```



```
      LOCATION     DATA                    PAGE LINE  LABEL   OP     OPERAND
       1200                                001  01            BEGIN  640
       1200                                001  02            ORIG   640
       1200,100                            001  03    PRINT   AREA   64
       1211,  6                            001  04    PACNO   —      6,10
       1235, 12                            001  05    WDRAW   —      10,30
       1261, 12                            001  06    DEPST   —      10,50
       1300,120                            001  07    CARD    AREA   80
       1305,  6                            001  08    ACCNO   —      6
       1313,  6                            001  09    AMNT    —      6
       1314,  1                            001  10    ACTN    —      1
       1420, 20                            001  11    WORK    AREA   16
       1427, 10                            001  12    TOTWD   —      8
       1437, 10                            001  13    TOTDP   —      8
       1451             @@@,@@@.@@         001  14    PATRN   +10    '@@@,@@@.@@'
       1457                 999999         001  15    SENTL   +6     '999999'
       1464     26 0 001437 2 00           002  01    START   PD1    TOTDP,16
       1471     20 0 001200   40           002  02            FT     PRINT,32
       1476     20 0 001300   44           002  03            FT     CARD,36
         42                                002  04    KEQ     EQU    34
          0                                002  05    KUC     EQU    0
         43                                002  06    KUQ     EQU    35
       1503     40 1 00 65 0100            002  07            XF     065,0100,,1
       1510     56 0 001451 2 12           002  08            BA2    PATRN
       1515     56 0 001305 0 06           002  09            BA1    ACCNO
       1522     26 0 001457 4 06           002  10            CD1    SENTL
    *  1527     30 0 001655   42           002  11            JC     CLOSE,KEQ

       1534     52 0 001211 0 06           002  12            SA1    PACNO
```



```
   LOCATION    DATA                 PAGE LINE  LABEL    OP    OPERAND
     1541    56 0 001313 4 06       002 13             BD1   AMNT
     1546    34 0 001314   27       002 14             CC    ACTN, 'D'
*    1553    30 0 001605   43       002 15             JC    WITHD, KUQ

     1560    62 0 001437 0 10       003 01             AM1   TOTDP
     1565    52 0 001261 4 12       003 02             ED    DEPST
     1572    22 0 001261 4 12       003 03             ZS    DEPST
     1577    26 0 001235 0 12       003 04             PD    WDRAW
*    1604    30 0 001643   00       003 05             JC    PRNT, KUC

     1611    34 0 001314   71       003 06   WITHD     CC    ACTN, 'W'
*    1616    30 0 001736   43        00307             JC    ERROR, KUQ

     1623    62 0 001427 0 10       003 08             AM1   TOTWD
     1630    52 0 001235 4 12       003 09             ED    WDRAW
     1635    22 0 001235 4 12       003 10             ZS    WDRAW
     1642    26 0 001261 0 10       003 11             PD    DEPST
     1647    40 0 00 62 0400        003 12   PRNT      XF    062,0400,,0
*    1654    30 0 001472   00       003 13             JC    START+10, KUC

       31                           003 14   KST       EQU   25
     1661    56 0 001427 0 10       003 15   CLOSE     BA1   TOTWD
     1666    52 0 001235 4 12       004 01             ED    WDRAW
     1673    22 0 001235 0 12       004 02             ZS$   WDRAW
     1700    56 0 001437 0 10       004 03             BA1   TOTDP
     1705    52 0 001261 4 12       004 04             ED    DEPST
     1712    22 0 001261 0 12       004 05             ZS$   DEPST
     1717    26 0 001211 0 06       004 06             PD    PACNO
```



```
     LOCATION    DATA                  PAGE LINE  LABEL   OP    OPERAND
       1724    40 0 00 62 0400        004 07             XF    062,0400,,0
   *   1731    30 0 001460    31      004 08             JC    START, KST

   *   1736    30 0 001472    31      004 09    ERROR    JC    START+10,KST

               30 0 001460    00      004 10             END   START
```

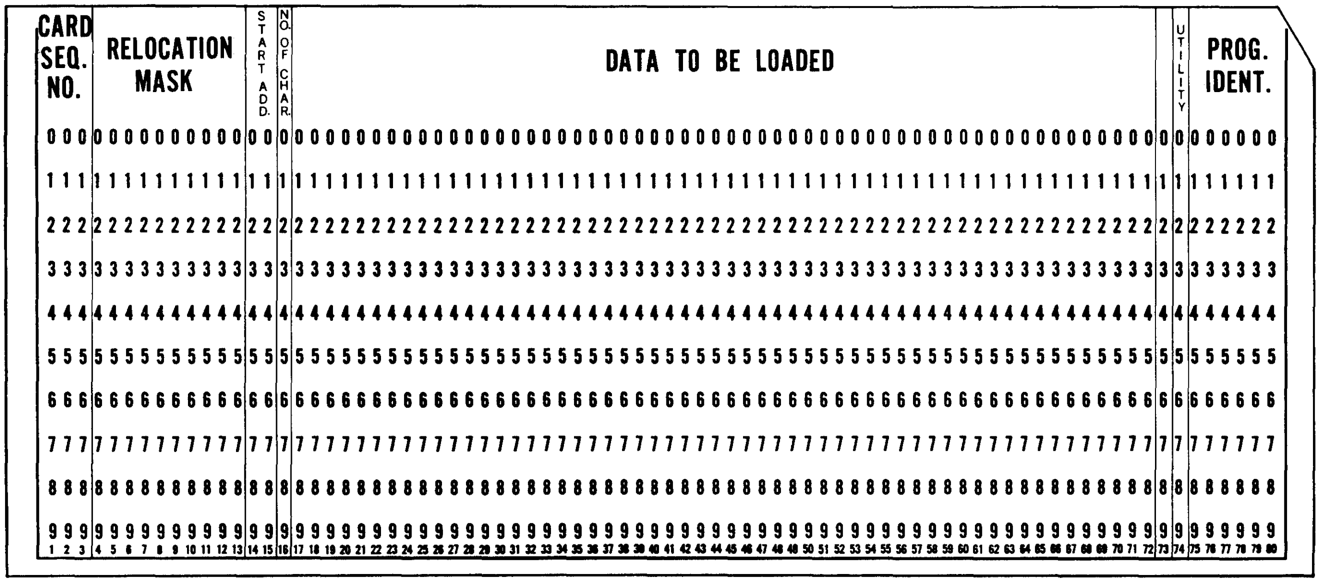
OUTPUT CARD FOR LOADING

80 Column Card

The primary output of the assembly system is the object program card deck. On every card of this deck the card sequence number is punched in columns 1 to 3 and the program identification is punched in columns 75 to 80.

The first card of the program deck is a label card. It contains an 'R' in column 74. Column 73 contains a 1 if the program is relocatable, or a 2 otherwise. If the program is relocatable, columns 4 and 5 contain, in binary, the address relative to which it was assembled. The total number of locations assigned to this program will appear in column 6 and the upper half of column 7. The lower half of column 7 and column 8 will contain a number 1 higher than the address of the highest location (assigned to this program) into which instructions or data will actually be loaded.

The data cards have the following format:



The card is read untranslated providing a card image of 160 characters. Card sequence number and program identification are punched in Hollerith code. All other fields are punched in binary. The starting address appears in columns 14 and 15 in the following form:

Col. 14	Col. 15
YX0123 456789	YX0123 456789

Punches in rows 6 to 9 of column 14 and column 15 comprise the address to be loaded with the leftmost character of the data.

The number of characters to be loaded occupies rows 3 to 9 of column 16.

Up to 112 characters of data to be loaded are contained in columns 17 to 72. They are arranged in the following manner:

Character 1	Character 2	Character 3	Character 111	Character 112
Column 17	Column 17	Column 18	Column 72	Column 72
YX0123	456789	YX0123	YX0123	456789

Columns 4 to 13 contain the relocation mask. Each punching position in this field corresponds to a half-column in columns 17 to 72, and a punch in any position in columns 4 to 13 indicates that the corresponding half-column in the data field contains the least significant character of an address that needs to be modified in the relocation. A punch in row 9 of column 13 indicates that the starting address must be modified.

RELOCATION MASK

COLUMN 4		COLUMN 13
YX0123 456789	. . .	YX0123 456789
000100 001000		0001-- -----1

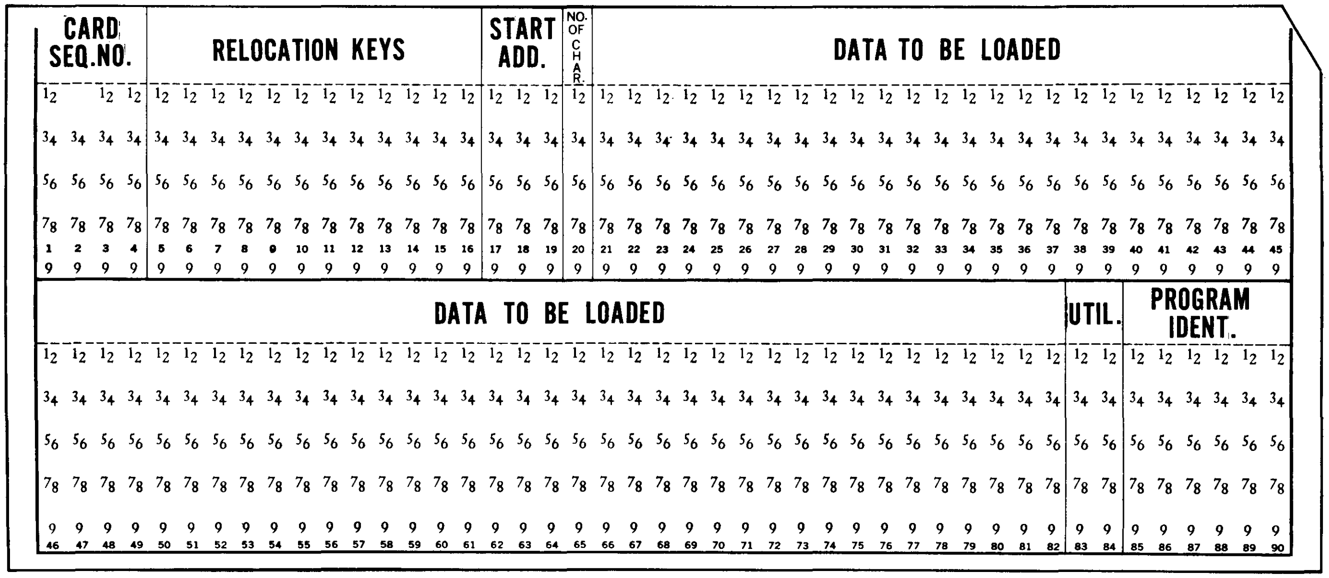
In the example above the 1-punch in Column 4 indicates that the lower half of column 18 contains the least significant character of an address that needs to be modified. The 6-punch in column 4 indicates that the upper half of column 21 contains the least significant character of an address that needs to be modified. The 1-punch in column 13 indicates that the lower half of column 72 contains the least significant character of an address to be modified. The 9-punch in column 13 indicates that the starting address needs to be modified.

The last card of the program deck is a transfer card. Column 74 contains the letter 'T'. The five characters starting with the upper half of column 17 contain a jump instruction to the start of the program just loaded.

90 Column Card

The primary output of the assembly system is the object program card deck. On every card of this deck the card sequence number is punched in columns 1 to 4 and the program identification is punched in columns 85 to 90. These fields are punched in card code.

The first card of the program deck is a label card. It contains an R in column 84. Column 83 will



contain a 1 if the program is relocatable or a 2 otherwise. If the program is relocatable, columns 5 to 7 will contain, in binary, the address relative to which the program was assembled. (Columns 8 to 10 will contain the number of locations assigned to the program. Column 11 to 13 will contain a number one higher than the address of the highest location into which instructions or data will actually be loaded).

The fields contained in columns 5 to 82 are punched in machine language, the others in card code.

Columns 5 to 16 contain relocation keys. Each key expresses in binary the relative position in the Data field of an address which is relative to the start of the program and needs to be modified. A blank column indicates that there are no more modifications required on this card.

COLUMN NO.	COL. 5	COL. 6	COL. 7
PUNCHING POSITION	013579	013579	013579
SAMPLE VALUE	000100	010011	000000

In the example above, column 5 has a 5-punch. This reads into the computer giving a binary value of 4, indicating that columns 22, 23, and 24 contain an address which needs to be modified. The 1, 7, and 9-punches in column 6 give a binary value of 19 indicating that columns 37, 38, and 39 also contain an address requiring modification. The blank in column 7 indicates that these are the only addresses requiring modification.

Columns 17 to 19 contain the address in storage in which the character in column 21 will be loaded. The following characters will be loaded in successively higher memory locations. The address field has the form

COLUMN 17	COLUMN 18	COLUMN 19
013579	013579	013579

The binary value of the address

A zero punch in column 17 indicates that the address in the rest of the field is relative to the start of the program and required modification.

Column 20 contains the binary representation of the number of characters of data contained in this card.

Columns 21 and following contain the actual data.

COLUMN 21	COLUMN 22	. . . .	COLUMN 82
013579	013579		013579

Data in the actual form in which it will be transferred to memory after modification for relocation.

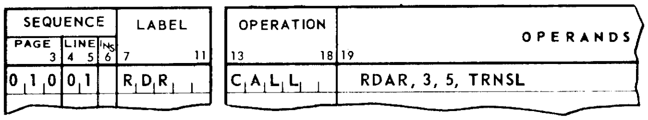
The last card of the program deck is a transfer card. Column 84 contains the letter T. The five characters starting with column 21 contain a jump instruction to the start of the program just loaded.

SPECIALIZATION OF I/O ROUTINES

Specialization of the input/output routines with respect to any particular set of options is accomplished during the assembly by including, in the input to the assembly, a call naming each routine with the parameters describing the specialization desired. The Input-Output routines are stored on a specialized routines tape.

Specification Cards

The call for each routine is on a separate line. The position of the call line card in the source 'language deck determines the position the input-output routine will occupy in the assembled program. The label field of an input-output specification line contains in the first four characters or less an identifier of the routine being specified. The word CALL must appear in the operation field. The operands field contains a series of expressions separated by commas. The number, nature, and interpretation of these expressions is determined by the particular routine being specialized. A sample specification for a card reader routine is shown below:



SEQUENCE			LABEL	OPERATION	OPERANDS
PAGE 3	LINE 4 5	INS 6	7 11	13 18	19
010	01		RDR	CALL	RDAR, 3, 5, TRNSL

This signals the assembly system to include at this point in the source program the symbolic coding for the card reader routine. The coding included will control the reading of translated card images into one of three areas, the first of which is addressed symbolically by the tag RDAR, with the other two following at increments of 128. The relative address of the next image available is supplied upon request and delivered in Index Register 5.

The inclusion of Input-Output routines is accomplished by a portion of the assembler which consists of a control routine that stores in fixed memory locations the specification parameters for the routine to be specialized. Each parameter must be sixteen characters or less. It then passes the assembler tape until it finds the appropriate routine, which it loads into a fixed area in memory which is set aside for these routines. Control is then transferred to the routine just loaded.

The control routine provides certain services to the routine being specialized. Each such service is obtained by loading a parameter for the service function desired into Arithmetic Register 1 and executing a JR instruction to the routine that performs the instruction. The services provided are:

- Bypass the next n lines of the routine.
- Include the next n lines in the assembler output.
- Load some more of the routine.
- Exit to the control routine to process the next routine.

INPUT-OUTPUT LIBRARY

The Input-Output library has been separated into two sections: Handling of the reader, punch and printer using the Input-Output Handling Functions and handling of tape files using the Input-Output Macro Instructions.

Input-Output Handling Functions

The purpose of the Input-Output Handling Functions (IOHF) is to provide a smooth efficient flow of information from and to the UNIVAC 1050 peripheral units. IOHF's are included with the Worker Program thru the use of the CALL operation of the assembly system. In the operand field of the CALL operation, the user provides the number of reserve areas to be used, the index register to be used, and the mode of operation (if applicable).

In general, the IOHF's are made up of three sections (Initialize, Execute and Close) which are available to the Worker Program. These sections are accessed by the Worker Program thru the use of a Jump Return (JR) to a specific tagged location. The Initialize section of each must be entered only once, but this entrance must be prior to any reference to the Execute section. Each time an Input-Output function is required, reference is made to the Execute Section. If the request can be satisfied immediately control is transferred back to the Worker Program; otherwise the IOHF will transfer control to a coordination function.

Each IOHF also contains an Interrupt section and an Error Recovery section; neither of which are accessible to the Worker Program. When interrupt occurs, error indicators are tested. When an error condition exists, programmed recovery will be attempted; otherwise the computer will be brought to a halt with a display indicating the nature of the error. If no error condition exists, the coordination function is referenced to determine whether another order should be issued on the peripheral unit which caused interrupt.

Each IOHF maintains a Unit Status List, which indicates the current condition of each peripheral unit. This list is used primarily by the coordination function.

CARD READER ROUTINE

This routine controls the operation of the High-Speed Reader when reading in the translated or untranslated mode. The reserve areas are aligned consecutively in memory; the Index Register specified in the CALL operation is used to address the current card image area. Programmed error recovery will take place where possible; otherwise coded stops will be used, which together with specific operator instructions will make for efficient error recovery.

Initialization (XINRD) must be accessed before any read images are requested. The address of the first card image is not supplied at this time. Execute (XCTRD) is entered when the Worker Program wants a card image. The relative starting address of the currently available card image will be supplied to the Worker Program in the Index Register specified in the CALL operation. When this section is accessed, the routine assumes that the Worker Program no longer needs the previously supplied card image.

CARD PUNCH ROUTINE

This routine controls the operation of the Punch when punching in the translated or untranslated mode. The reserve areas are aligned consecutively in memory; the Index Register specified will give the Worker Program the relative address of the current punch area. Programmed error recovery will take place when possible, operator corrective actions are supplied for non-program correctable errors.

Initialization (XINPH) must be performed before any attempt to produce card output. This section will clear the punch areas to blanks and provide the relative address of the punch area to the Worker Program.

When a punch area is filled with data and is ready to be released for output, the Worker Program performs the Execute (XCTPH) section. At the conclusion of a run, the Worker Program must enter the closing section (XCLPH) of the routine. This section will punch all remaining images provided and clear the punch unit of data cards.

PRINTER ROUTINE

This routine controls the operation of the High Speed Printer. Reserve areas are aligned in consecutive memory positions; the index register specified is used to tell the Worker Program the relative address of the current print area. Coded error stops, along with operator instructions, are provided for recovery. Initialization (XINPR) must be performed before any attempt to produce printed output. This section clears the print areas to blanks and provides the Worker Program with the relative address of the first print area.

The Execute Print section (XCTPR) is entered when the Worker Program has filled a print area and is ready to release this area for printing. The Worker Program must store character (SC) in XADVC indicating the number of lines of advance (in binary). The number of lines of advance between print lines remains constant unless changed by the Worker Program.

The Execute Advance section (XCTAD) is entered when it is desired to advance paper without printing.

At the end of a run, the closing section (XCLPR) must be entered so that all remaining print images provided will be printed.

The specialized card input-output routines are included with the object programs as a result of including Input-Output Specification Cards (CALL operation) with the input to the assembly.

These cards have the following format:

- **Label field** — Name of the routine being called —

  RDR for the reader routine

  PCH for the punch routine

  PRNT for the printer routine

- **Operation field** — the word CALL
- **Operands field** — a number of parameters (p_1, p_2, . . .)

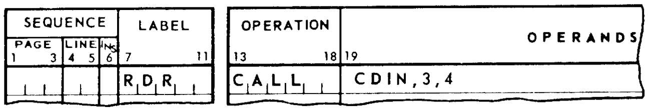
$\mathbf{p_1}$ is the name of the area defined for this routine. The area named must also appear in the label field of an AREA directive. The size of the area must be large enough to hold the entire reserve storage area. The first character of the area must be assigned an address which is a multiple of 64.

$\mathbf{p_2}$ is the number of reserve storage areas to be serviced by the input-output routine. A minimum of 2 for PCH and PRNT and a minimum of 3 for RDR.

$\mathbf{p_3}$ is the Index Register (1-7) associated with the input-output function.
$\mathbf{p_4}$ is the mode of operation of the peripheral unit.

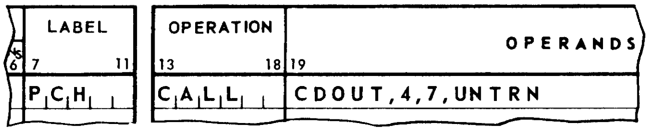
> △ (blank) or TRNSL for the inclusion of a routine operating in the translated mode or when calling for a printer routine.
>
> UNTRN for the inclusion of a reader or punch routine operating in the untranslated mode.

Examples



SEQUENCE PAGE 1 3	LINE 4 5	INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19
			RDR	CALL	CDIN,3,4

The above CALL will include with the object program, a reader routine servicing 3 reserve areas operating in the translated mode. Index register 4 and the area CDIN will be referenced. CDIN must be an area containing at least (3 x 128) 384 characters.



INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19
	PCH	CALL	CDOUT,4,7,UNTRN

The above CALL will include with the object program, a punch routine servicing 4 reserve areas and operating in the untranslated mode. Index register 7 and the area CDOUT will be referenced. CDOUT must be an area containing at least (4 x 192) 768 characters.

Input-Output Macro Instructions

The details of the Input-Output Macro Instructions have been presented earlier in this section. A summary of the use of these macro-instructions is presented.

Each tape file of a program may only use one of the two forms of GET, and one of the three forms of PUT. The form number is summarized in parenthesis below. The choice of the form used depends upon the type of processing to be done. Possible combinations of the forms of GET and PUT are listed below:

In Processing, if Information is		The Operands Field Should be of the Form	
Picked Up From Area	**And Stored in Area**	**GET**	**PUT**
FLA	FLA	(1) FLA	(5) FLA, FLB
FLA	FLB	(1) FLA	(3) FLB
FLA	WKAR	(1) FLA	(4) WKAR, FLB
WKAR	FLB	(2) FLA, WKAR	(3) FLB
WKAR	WKAR	(2) FLA, WKAR	(4) WKAR, FLB

SYSTEMS AND FILE PARAMETERS

The specialization and inclusion of the necessary tape Input-Output routines is accomplished by including Input-Output Specification Cards (CALL operation) with the input to the assembly. These Cards have the following formats:

- 1st Card (One per tape program)
  - Label field — Name of the routine being called, Tape.
  - Operation field — the word CALL.
  - Operands field — a parameter (p_1).

p_1 denotes the computing system that the UNIVAC 1050 System is satellite to (that is, the source or receiving computer).

1 =	UNIVAC III System
2 =	UNIVAC 490 System
3 =	UNIVAC 1107 System
10 =	IBM 1410 System
11 =	IBM 705 System
12 =	IBM 7070 System
13 =	IBM 7080 System
14 =	IBM 7090 System

Samples of the 1st card of a CALL for tape routines are:

LABEL 7 11	OPERATION 13 18	OPERANDS 19
TAPE	CALL	1

This call will include the coding to control a UNIVAC III System data tape.

LABEL 7 11	OPERATION 13 18	OPERANDS 19
TAPE	CALL	10

This call will include the coding to control an IBM 1410 System data tape using the standard header and trailer conventions.

- File Descriptor Cards (one for each tape file) describe the files to be operated on and the conventions to be used.
  - Label field — Blank
  - Operation field — The continuation symbol (-)
  - Operands field — A number of parameters (p_0, p_1, . . .)

p_0 is a 1 to indicate this is a File Descriptor Card.

p_1 is the file identifier consisting of 3 alphabetic characters. Any references to this file in the Input-Output macro-instructions must use this tag. This tag must also appear in the label field of an AREA directive defining an area large enough to contain a block of this file.

p_2 indicates the nature of the file.

I for Input

O for Output

p_3 is the decimal number of characters per record.

p_4 is the decimal number of records per block.

p_5 is the file name to be used in label checking or writing.

p_6 is the number of block areas reserved for this file.

p_7 is the rewind option desired.

1 = rewind with interlock

2 = rewind

3 = leave tape in position

p_8 indicates the form of the macro instruction used.

if $p_2 = I$, p_8 can be 1 or 2.

if $p_2 = 0$, p_8 can be 3, 4 or 5.

p_9 is the logical servo assignment for this file (0-5).

p_{10} is the index register to be referenced (1-7). This entry may be blank if p_8 is 2 or 4, or p_4 is equal to 1. If an index register is to be referenced the same index register must also be referenced in the AREA directive with the file identifier in the label field.

p_{11} denotes the mode of tape reading or writing and the conventions used on the source or receiving computer.

If source or receiving computer is UNIVAC III System p_{11} is

1 for SALT data tape conventions, Block read

2 for SALT data tape conventions, Block write

3 for SALT data tape conventions, Gather write

6 for UTMOST data tape conventions, Block read

7 for UTMOST data tape conventions, Block write

8 for UTMOST data tape conventions, Gather write

If source or receiving computer is UNIVAC 1107 System p_{11} is

21 for SLEUTH data tape conventions

22 for FORMOST data tape conventions

If source or receiving computer is UNIVAC 490 System p_{11} is

31 for SPURT data tape conventions

If source or receiving computer is IBM 1410 System p_{11} is

51 for fixed length unblocked records without record mark

52 for fixed length unblocked records with record mark

53 for fixed length blocked with record marks

54 for variable length unblocked without record marks

55 for variable length unblocked with record marks

56 for variable length blocked with record marks and Block Character Count Fields, and Record Character Count Fields

p_{12} is the tag of the instruction to which control is transferred when the end of file sentinel is sensed. This entry is blank for an output file, but must be provided for an input file.

Examples of File Descriptor Cards follow:

OPERATION 13 18	OPERANDS 19
-	1,FLA,1,36,4,DATA,1,1,1,0,6,8,LAST

The above describes the Input file FLA, which contains 4 records per block and 36 characters per record and 1 block area. Form 1 macro-instruction is used. The logical servo assignment is to be 0, the index register used is to be 6 and the tape is to be rewound with interlock. The name (label block) of the file is DATA. The UNIVAC III System UTMOST data tape conventions were used; the tape was written in the Gather-Write mode. Upon sensing the tape end of file sentinel, control will be transferred to the instruction labeled LAST.

OPERATION 13 18	OPERANDS 19
-	1,FLB,0,80,1,CARD,1,3,4,1,,1

The above describes Output file FLB, which contains 1 record per block and 80 characters per record and 1 block area. Form 4 macro-instruction is used; no index register is required. The logical servo assignment is to be 1 and the tape is to be left in position after the sentinel block(s) are written. The name (label block) is to be CARD. The UNIVAC III System SALT data tape conventions are to be used and the tape will be read (on the UNIVAC III System) in the Block-Read mode.

- Special Option cards allow for the use of nonstandard header and trailer label conventions. These cards follow the File Descriptor Card and have the following format:
  - Lable field — Blank
  - Operation Field — The continuation symbol (-)
  - Operands field — A number of parameters (p_0, p_1, p_2, . . .)

p_0 is a 2 indicating this is a Special Option Card.

p_1 is the file identifier and must be identical to the file identifier on the preceding File Descriptor Card.

p_2 is the tag of a subroutine which is to supplement the standard header label checking or writing routine. If the file is an input file, control will be transferred to this tag after the header label has been read and the standard checking accomplished. The user can insert coding to do additional checking but must return control to XHYYY, where YYY is the file identifier, before further processing will occur. If the file is an output file, control will be transferred to this tag after the header label has been set up in the standard format. The user can insert coding to add additional information to the header block but must return control to XHYYY, where YYY is the file identifier, before the header label is written and processing will continue.

p_3 is the tag of a subroutine which is to supplement the standard trailer label writing routine. Control will be transferred to this tag after the standard trailer label has been set up. The user can insert coding to add additional information to the trailer block, but must return control to XTYYY before the trailer label is written.

An example of a Special Option card follows:

OPERATION 13 18	OPERANDS 19
–	2,FLB,LABEL,TRAIL

This card indicates that for FLB, the standard header and trailer label routines are to be supplemented. Coding to supplement the standard header label routine starts at the instruction tagged LABEL; control must be returned to XHFLB. Coding to supplement the standard trailer label routine starts at an instruction tagged TRAIL; control must be returned to XTFLB.

PATCH ASSEMBLER

Programs for the UNIVAC 1050 System exist on cards in an untranslated form. The facility to make corrections to such programs in the assembler source language is provided in the Patch assembler. This program allows changes, insertions and deletions to be made to such a deck. Its output is a card deck to be added to the back of the program deck being corrected.

The operations used in controlling the correction process are PAREA, CHGE, PTCH. The first of these is written:

OPERATION 13 18	OPERANDS 19
PAREA	n_1, n_2

where n_1 and n_2 are decimal or octal numbers giving the address of the first and the last character positions in a patch area. Octal numbers are written with a leading zero. This area will be made available to the corrector for containing the insertions to be made in the program to be corrected. This area will be used by the corrector for this purpose until it is filled or another patch area is specified.

The directive CHGE may be written as:

OPERATION 13 18	OPERANDS 19
CHGE	n_1, n_2

where n_1 and n_2 are decimal or octal numbers representing the first and last locations of an area containing instructions or constants to be deleted from the program. The deleted data will be replaced by the lines which immediately follow the CHGE line and precede the next PAREA, CHGE, or PTCH line.

These lines must either:

- exactly fill the area formerly occupied by the deleted data; or
- occupy no more locations than 5 less than that area. In this case the first five locations not replaced will be filled with a jump to the first location beyond the deleted area.

This directive may also be written as:

OPERATION 13 18	OPERANDS 19
CHGE	n_1

where n_1 represents the address of the least significant character of a single instruction which it is desired to replace with one or more instructions. The replacements are listed in the immediately succeeding lines. If the replacement is a single line it will be inserted in memory location n. If there is more than one, the replacements will be stored in the patch area starting at the current location and followed by a jump to the location n + 1. Location n will contain a jump to the first replacement instructions.

The directive PTCH is written:

OPERATION 13 18	OPERANDS 19
PTCH	n

where n is a decimal or octal number representing the address of the least significant character of an instruction after which an insertion is to be made. The lines which immediately follow the PTCH line and precede the next PTCH, PAREA, or CHGE line comprise the insertion. The instruction which had occupied locations n−4 to n will at load time be placed instead in the current location in the patch area followed by the insertion and a jump to location n+1. Locations n−4 to n will contain a jump to the new location occupied by that instruction.

The directive may also be written:

OPERATION 13 18	OPERANDS 19
PTCH	*

to achieve the inclusion of the following lines in the patch area without a succeeding jump instruction.

A correction line may be any mnemonic instruction or data generating code which is appropriate. Insertion lines may also include

EQU, AREA, SNAP, PRINT and REPL statements.

The last correction to be made is followed by the command STOP. The input to this routine consists of a DUMP statement, if this routine is desired, followed by the symbol table for the program, followed by the corrections. The first insertion required must be preceded by a PAREA statement. The output consists of an updated symbol table, additions to the object program deck and a listing of the changes showing both the symbolic and absolute coding.

Example

In a program which is to be corrected memory positions 06701 to 07777 (octal) are available for patches. It is desired to delete the instructions in positions 1516 to 2063 (octal) replacing them with a jump to the instruction in positions 2064 to 2070. This instruction is tagged RECAP. After the instruction in positions 2374 to 2400 two lines of coding are to be inserted to store a partial result. The instructions to the Object Code Corrector are:

OPERATION 13 18	OPERANDS 19
PAREA	06701, 07777
CHGE	01516, 02063
JC	RECAP
PTCH	02400
SA1	TS1, 5
SA1	STDHR, 5
STOP	

REGENT REPORT PROGRAM GENERATOR

REGENT is a programming tool which translates organizational reporting requirements into detailed machine programs. Because report specifications can be written in the language of REGENT by personnel who have only a basic knowledge of data processing concepts and terms, the routine saves both time and effort.

The preparation of a report begins with the following specifications: description of the input file; selection of control break fields; description of the processes required to derive values not directly punched in the input record; and description of the output lines which are to be printed; and the description of the output cards which are to be punched. Each of these descriptions is punched into cards referred to as Record Type Descriptor, Control Break, Process (Basic and Extended), Line Position and Field Descriptor Cards. These cards are then operated upon by segments of the REGENT routine to produce various program sections.

The REGENT Record Type Generator processes the Record Type Descriptor cards to produce machine coding which will, in the object program, detect the presence of a record type. Control Break Cards are then processed by the Control Generator to obtain machine coding which will detect the presence of a control break. Next, the REGENT Process Generator acts upon descriptors of simple arithmetic operations to produce coding that will accomplish these operations. An Extended Processor Generator utilizes descriptions of more complex arithmetic functions with the same result. Finally, the output descriptions are employed by the Ouput Generator to create machine coding to line space and assemble a line image.

The object program does not require an assembly phase but may be loaded, ready to run, into storage by the Loader.

The REGENT routine with its powerful generation capability balances the need for coding simplicity with the requirement of flexibility.

INPUT-OUTPUT INSTRUCTIONS XF F, D, U, X

Because the programmer will normally use the macro-instructions of the PAL language, a discussion of the external function instruction has been deferred to this point.

Two sets of memory locations within the first 320 positions of core storage are assigned for use with each input-output channel (see Figure 3-1).

- Four groups of four characters each (Tetrads) are assigned for each input-output channel. These contain memory base addresses, block character counts, and other detail pertaining to an input-output function.
- Eight character positions (Interrupt Entries) are assigned to each input-output channel to work with the automatic input-output interrupt feature. These positions contain the information necessary to transfer to a routine to service an interrupting input-output device and to return control to the interrupted program.

Input-Output Control involves the following functions:

- Placing a memory base address into a Tetrad which provides a specific Input-Output Control Unit the initial memory position to which or from which data transfer will take place.
- Placing into other Tetrads any detail information (character counts, sector counts, and so on) necessary to perform a given input-output instruction.
- Issuing an input-output instruction which specifies the channel to be used, the particular input-output unit involved (if there is more than one unit on that channel) and the function or operation required.
- Testing the input-output indicators to determine if the operation was completed successfully. If it is determined not to have been successful, further testing is required to isolate the cause of failure.

An interrupt routine must be provided for each channel being used. These routines contain the instructions required to provide programmers with decimal overflow optional decision path, to keep the input-output devices running, to detect any errors in the input-output devices which may occur, and to take corrective action where possible.

The memory addresses of these routines must be placed in the Interrupt Entry Areas for each channel in use. This must be done at the beginning of a program.

A jump or transfer instruction without a memory address and containing specific channel inhibit release indication, must also be placed in each Interrupt Entry Area. Upon interruption by an input-output channel, the memory address of the next processor instruction in normal sequence is automatically stored into the address field of this instruction. Then control is automatically transferred to the address of the interrupt routine for that channel. At the end of the interrupt routine, the program jumps to the instruction in the Interrupt Entry Area. This instruction in turn releases the interrupt inhibit condition and jumps to the proper place in the program to allow execution of the instruction following the one which took place just before the interruption was effective.

If the Memory Address Register serving any input-output device attempts to access memory areas with an address beyond the capacity of the specific system, the contents of the actual memory are not affected.

A read from memory access falling outside the actual memory limits will result in a parity error. A write to memory access outside memory limits will be lost.

Format

The format of the external function differs markedly from that of other instructions. Consequently, the significance of bit combinations in the designated positions will be listed in tabular form under the input-output unit to which they pertain.

The machine format for the external function instruction is:

OPCODE	CHANNEL	UNIT	FUNCTION	DETAIL/INDICATORS
29 25	24 22	21 18	17 12	11 0

The OP CODE of 40 determines that an input-output operation is to be performed. The channel designation determines the peripheral device:

Channel	Device
0	High-Speed Printer
1	High-Speed Card Reader
2	Card-Punch Unit
3	Unassigned
4	Tape Read
5	Tape Write
6-7	Unassigned

The UNIT designation specifies which unit on the channel is to be used. When there is only one unit, the entry is always 0. When there is more than one unit, the first is designated as 0, the second as 1, and so on.

In the following discussion, the values of the FUNCTION and DETAIL fields are stated in octal. The PAL Assembler recognizes that an integer is octal rather than decimal if the leading digit is zero.

Functions available with each of the input-output units are:

	Value of the FUNCTION field	Significance
FUNCTION	00	Test Indicators
	040	Reset Indicators

The Test Indicators function reflects the setting of one or more indicators addressed in the DETAIL portion of the XF instruction. If one or more of the indicators addressed is set, testable indicator 43 is set to 1. A single XF instruction followed by a JC instruction is sufficient to determine if the interrupt reflects successful completion of an input or output function. If not, further testing determines the nature of the error and the type of corrective action to be taken.

The Reset Indicators function is used to selectively reset specific fault or error indicators after corrective measures have been taken. In the instances where an error condition arises, it is mandatory that these indicators be reset after the malfunction has been corrected and the Ready Switch has been depressed. Otherwise, Indicators may be set incorrectly at subsequent interrupts.

The indicators available and the remaining functions and their detail expansions are specific to the peripheral unit and are discussed separately below.

High-Speed Printer

The printer advances the paper and prints either a full line of 128 characters or a half line of 64 characters. Paper can be advanced without printing. Print codes are shown in table 5-1 . The manual print functions will print without paper advance and without base address advance to overide the inhibition of printing by a paper-low condition.

TETRADS

Tetrad 33 must contain in its least significant character the count of the number of lines to be advanced (0 through 63) before printing. Tetrad 32 contains in its three least significant characters the base address of the information to be printed. If the advance base address option is selected, this number will be incremented upon com.-pletion of transfer of data to the print buffer. It will be increased by 128 if a full line was printed (DETAIL field of 0200) or by 64 if a half line was printed (DETAIL field of 0600).

FUNCTION and DETAIL Field Values

		SIGNIFICANCE			
	Value	ADVANCE	PRINT	Allow Interrupt at Successful Completion	Continue the Program during the Advance & Print Operation
FUNCTION	02	■	■	■	■
	04	■		■	■
	022	■	■		■
	024	■			■
	042	■	■	■	
	044	■		■	
	062	■	■		
	064	■			
DETAIL	0	Print full line, reset base address			
	0200	Print full line, advance base address			
	0400	Print half line, reset base address			
	0600	Print half line, advance base address			
	02000	Manual print, paper low bypass, print full line			
	02400	Manual print, paper low bypass, print half line			
		The following values of the DETAIL field represent indicators which may be addressed by the Test or Reset Indicators function. These values may be added together in any combination to allow testing or resetting of several indicators with a single instruction.			
INDICATORS	02	Paper low			
	020	Manual print button depressed			
	040	Parity error (code wheel)			
	0100	Parity error (memory)			
	01000	Unit not ready (off normal)			
	02000	Printer busy			
	04000	Memory overload anticipated			

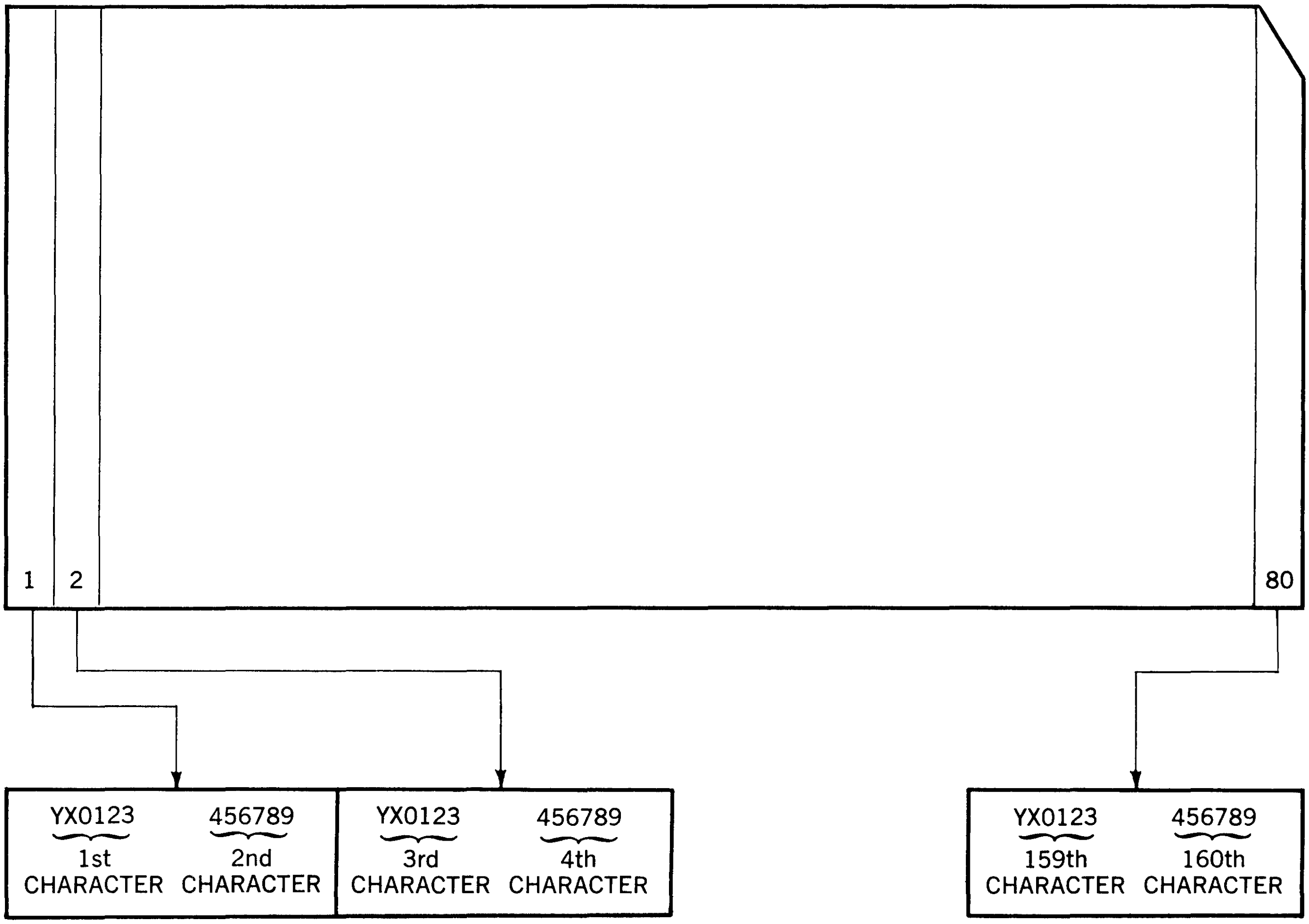
Card Reader

With an 80-column unit, automatic translation to computer code, under program control, is provided. In addition, the 80 column card reader accepts stub cards. The number of characters of storage required for the 80-column card image is 80 if automatic translation takes place, and 160 otherwise. The binary reading of 80-column cards produces 160 characters in memory.

The following diagram shows the image of an 80-column card in memory when read in the untranslated mode.

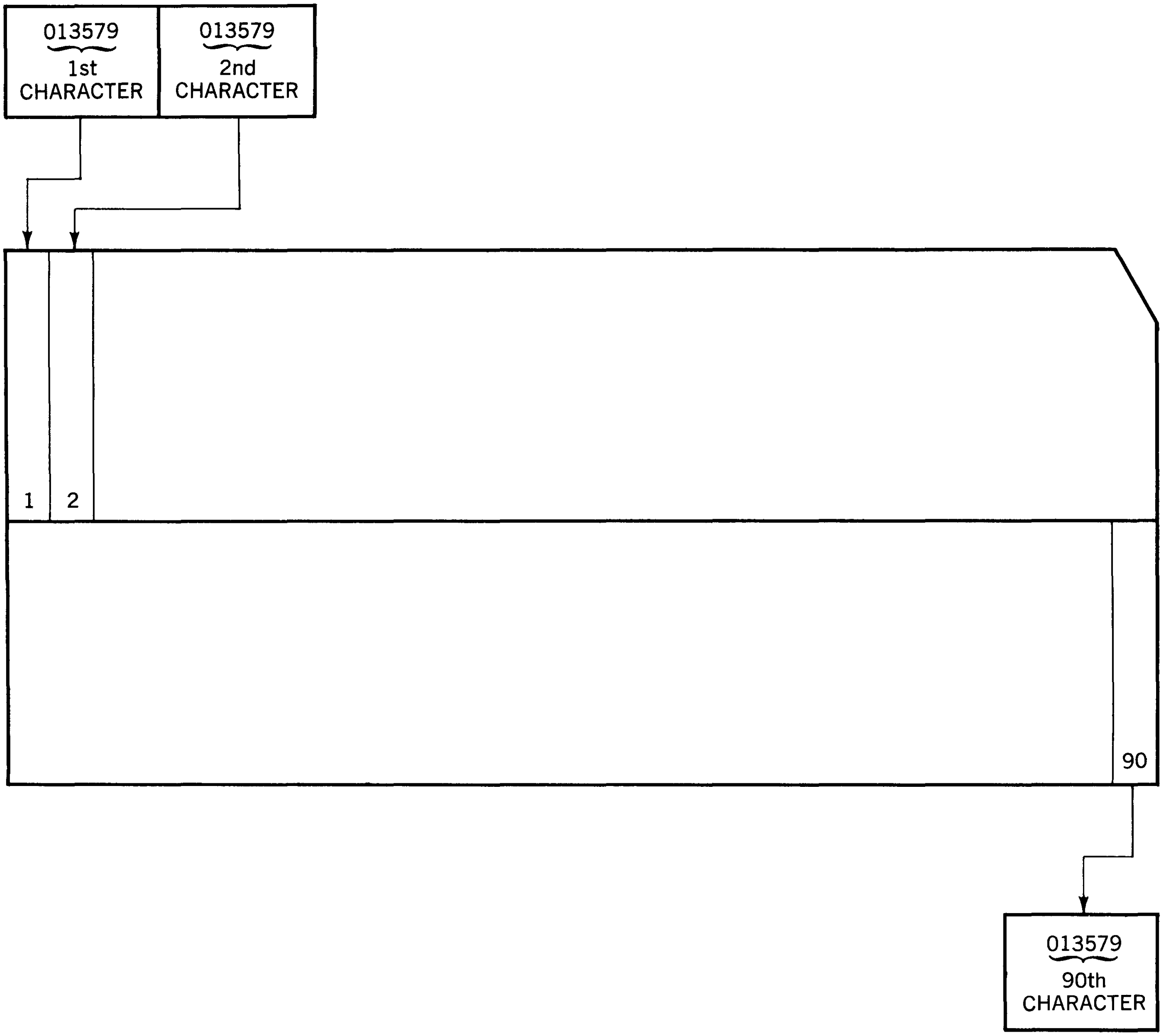
Within each character, bit positions 3 and 9 are the least significant bits. 80-column cards are read face down with the 9-edge leading.

When stub cards are read, 160 characters are transferred to memory. The programmer may not depend on information in the input area other than what is actually on cards.



With a 90-column unit no translation is provided. The control unit can transmit data from a stub card. The number of characters of storage required for the 90-column card is 90.

90-column cards are placed in the hopper face up with the 9-edge leading. The following diagram represents the image of a 90-column card in memory.

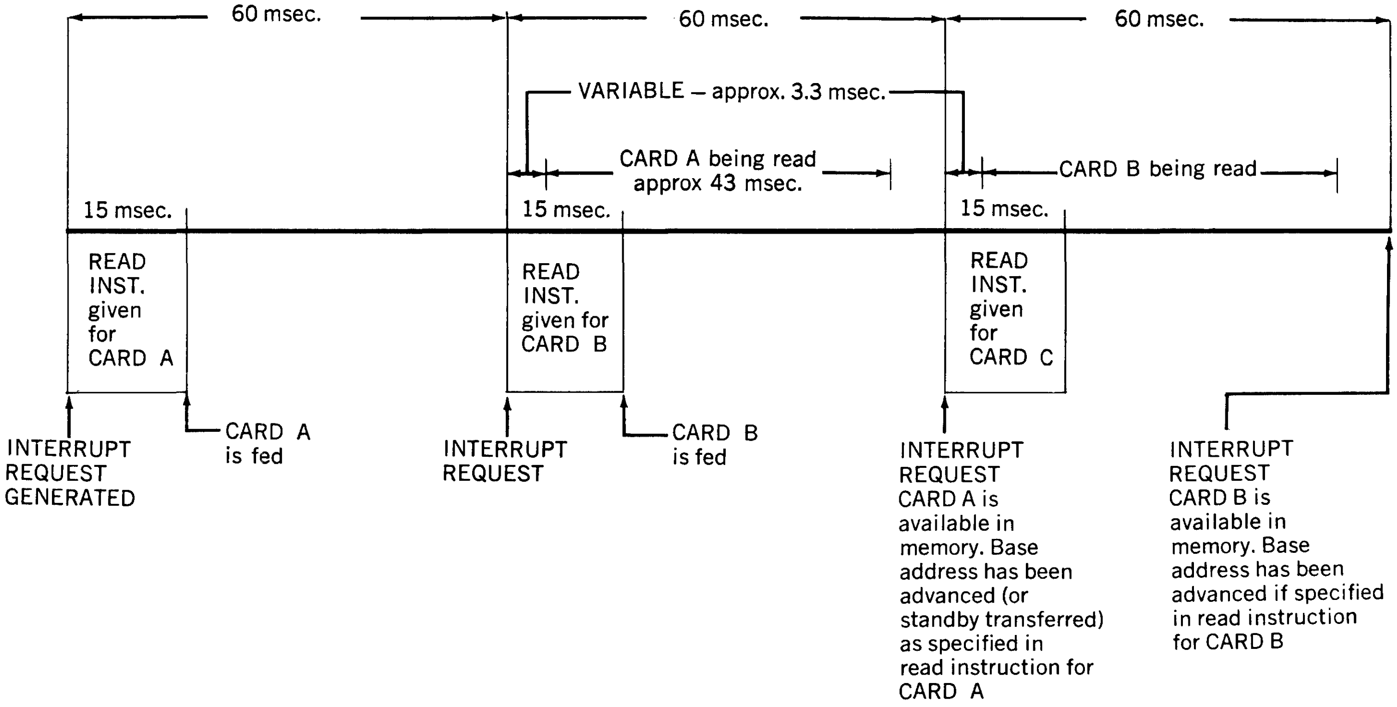


The read card function causes a card to be fed from the input hopper, read into storage, and deposited in a stacker.

The card reader operates at a rate of 1000 cards per minute or 60 milliseconds per cycle. Unless inhibited by the instruction which feeds a card or by a separate instruction to the card reader, an interrupt request will be generated once every card cycle. There is, at the time of the interrupt request, 15 milliseconds in which to issue a card feed instruction.

TETRADS

Tetrad 36 contains the base address. This must be a multiple of 64 and is the address of the position into which the next card will be read. Tetrad 37 contains the standby base address. Upon completion of reading a card the contents of Tetrad 37 are automatically transferred to Tetrad 36. If the instruction which fed the card specified advance base address, this transfer is inhibited. Instead the contents of Tetrad 36 are incremented by 128 for a 90-column card or an 80-column card read in the translated mode or by 192 for an 80-column card read in the untranslated mode.


60 msec.
60 msec.
60 msec.
VARIABLE – approx. 3.3 msec.
CARD A being read approx 43 msec.
CARD B being read
15 msec.
15 msec.
15 msec.
READ INST. given for CARD A
READ INST. given for CARD B
READ INST. given for CARD C
INTERRUPT REQUEST GENERATED
CARD A is fed
INTERRUPT REQUEST
CARD B is fed
INTERRUPT REQUEST CARD A is available in memory. Base address has been advanced (or standby transferred) as specified in read instruction for CARD A
INTERRUPT REQUEST CARD B is available in memory. Base address has been advanced if specified in read instruction for CARD B

FUNCTION and DETAIL Field Values

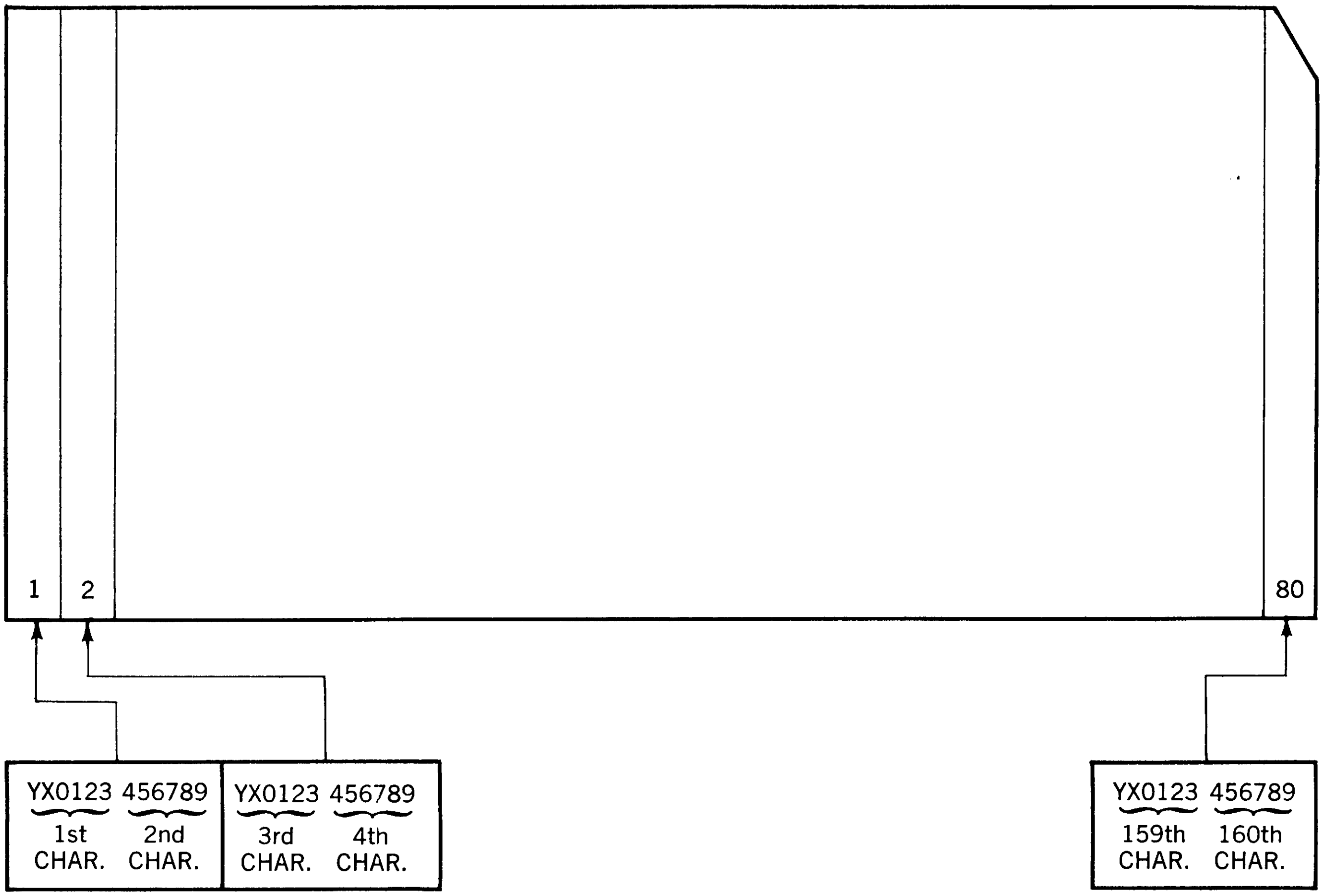
		SIGNIFICANCE		
	Value	Read A Card	Allow Interrupts at successful completion and cyclically thereafter	Continue the Program while the card is being read
FUNCTION	01	■	■	■
	021	■		■
	041	■	■	
	061	■		
	020	Prevent cyclic interrupt beginning with the one that would have occurred at the end of the current card cycle.		
DETAIL		(Detail field is ignored for function codes 00 and 020. The following Detail expansions are used with a read card function).		
	00	Read the card without automatic translation. At the conclusion of reading move the standby address (Tetrad 37) the base address (Tetrad 36).		
	0100	Read the card with translation (for 80-column cards only). At the conclusion of reading move the standby address to the base address.		
	0200	Read the card without translation. At the conclusion of reading increment the base address by 128.		
	0300	Read the card with automatic translation (for 80-column cards only). At the conclusion of reading increment the base address by 192.		
		The following values of the DETAIL field represent indicators which may be addressed by the Test or Reset Indicators function. These values may be added together in any combination to allow testing or resetting several indicators with a single instruction.		
INDICATORS	04	Registration check error.		
	010	Two cards in track at time of error.		
	020	At least one card has been fed which has not yet finished reading successfully.		
	040	Solar cell error.		
	0100	Data Parity error.		
	01000	Unit not ready (off normal).		
	02000	Unit busy.		
	04000	Memory overload anticipated.		

Card Punch Unit

With an 80-column unit, automatic translation to card code, under program control, is provided. The number of characters of storage required for the 80-column card image is 80 if automatic translation takes place, and 160 otherwise.

The following diagram represents the image of an 80-column card in memory when it is to be punched in the untranslated mode.

Within each character, bit positions 3 and 9 are the least significant bits. 80-column cards are punched face down with the 9-edge leading.



With a 90-column unit no translation is provided. The number of characters of storage required for the 90-column card is 90. 90-column cards are placed in the hopper face up and with the 9-edge leading. The following diagram represents the image of a 90-column card in memory.

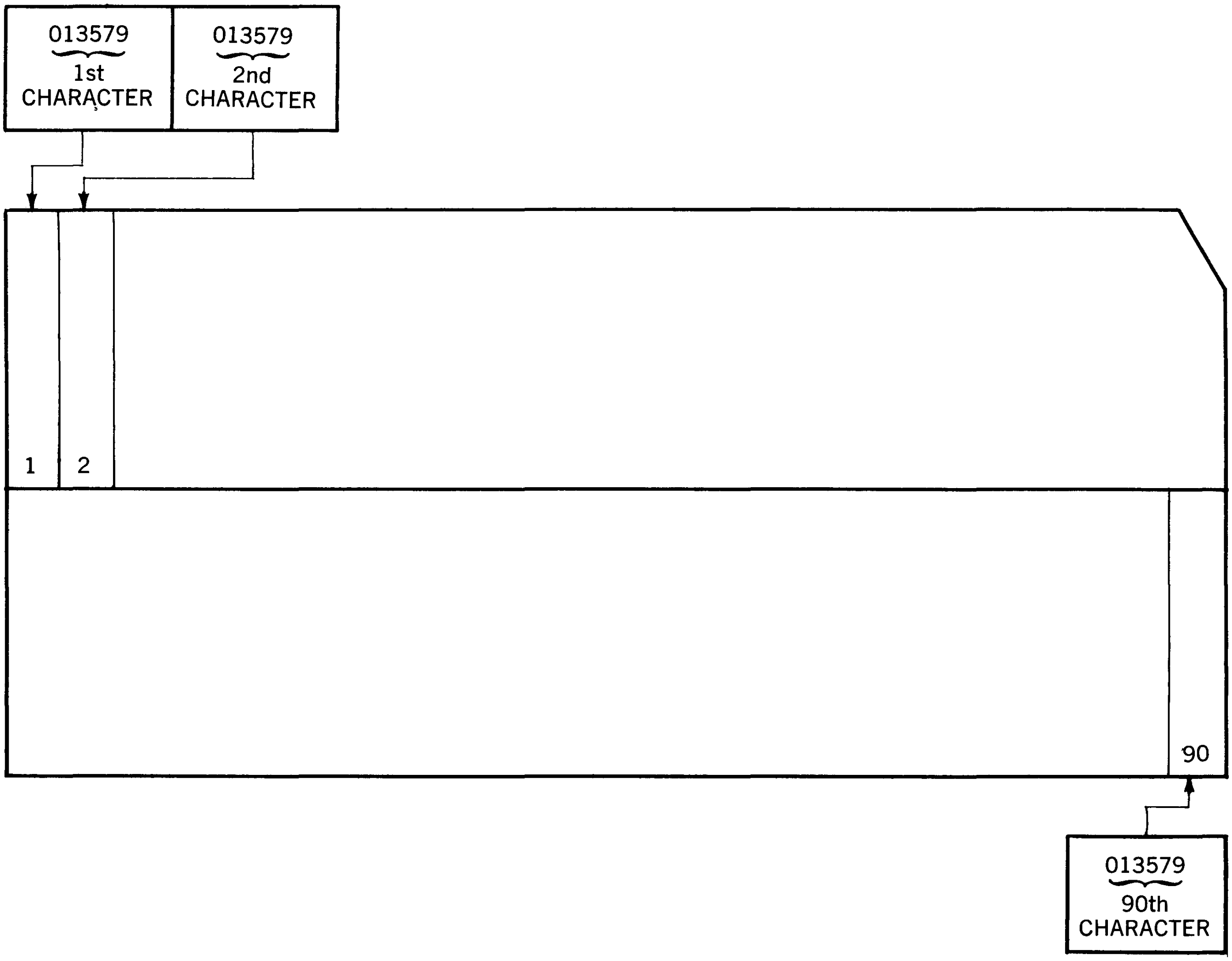
The punch instruction causes all of the cards in the track to be advanced one station.

1. *A card is fed from the input hopper into wait station 1.*
2. *The card that had been in wait station 1 is advanced to wait station 2.*
3. *The card in wait station 2 is advanced to the punch station.*
4. *The card in the punch station is punched row by row as it advances to the post-punch check station.*
5. *A hole count check is made of the card at the post-punch check station and it is passed into an output stacker.*

The Card Punch unit has two output stackers. Cards are normally sent to stacker 2, but in the case of an error, stacker 1 is selected automatically. It is also possible to select stacker 1 with a punch instruction. The selection of stacker 1 will be effective only for the card which is at the post-punch check station when the instruction is given.

TETRADS

The punch base address must be stored in Tetrad 40 before the punch instruction is given. Tetrad 42 is reserved by the punch synchronizer for its own use and should not be used by any program. The punch instruction includes an option to increment the base address at the conclusion of the cycle.



FUNCTION and DETAIL Field Values

		SIGNIFICANCE			
	Value	Punch the Card at the Punch Station. Advance all Cards One Station	Advance All Cards One Station Without Punching	Allow Interrupt at Successful Completion	Continue the Program During the Operation
FUNCTION	04		■	■	■
	06	■		■	■
	24		■		■
	26	■			■
	44		■	■	
	46	■		■	
	64		■		
	66	■			
DETAIL		Except for 00 and 01000, the values described below are used only with the punch functions.			
	00	Card at the check station goes to the normal stacker. Base address is not advanced at end of cycle. No automatic translation.			
	0100	Card at the check station goes to the normal stacker. Base address is not advanced. Card at punch station is punched with automatic translation.			
	0200	Card at the check station goes to the normal stacker. Base address is advanced after card is punched. No automatic translation.			
	0300	Card at the check station goes to the normal stacker. Base address is advanced after the card is punched. Card at punch station is punched with automatic translation.			
	01000	Card at the check station is selected to stacker 1. Base address is not advanced. No automatic translation.			
	01100	Card at the check station is selected to stacker 1. Base address is not advanced. Card at punch station is punched with automatic translation.			
	01200	Card at the check station is selected to stacker 1. Base address is advanced after the card is punched. No automatic translation.			
	01300	Card at the check station is selected to stacker 1. Base address is advanced after the card is punched. The card is punched with automatic translation.			
		The following values of the DETAIL field represent indicators which may be addressed by the Test or Reset Indicators function. These values may be added together in any combination to allow testing or resetting of several indicators with a single instruction.			
INDICATORS	040	Hole count error detected at check station.			
	0100	Parity error.			
	01000	Unit not ready (off normal).			
	02000	Unit busy.			
	04000	Memory overload anticipated.			

UNISERVO III A Magnetic Tape Unit

The UNISERVO III A Magnetic Tape Control Unit controls writing and reading of magnetic tapes generated by or for the UNIVAC III, 1107 or 490 Systems. It allows read-compute and write-compute, but not read-write-compute. The Control Unit permits the writing and reading in two modes, the UNIVAC III mode and the UNIVAC compatible mode.

Information is recorded on tape in nine channels at a density of 1000 frames per inch. The nominal tape speed is 100 inches per second for a reading or recording rate of 100,000 characters per second in the UNIVAC compatible mode and 133,000 characters or 200,000 digits per second in the UNIVAC III mode. In the UNIVAC compatible mode, a frame of tape contains one 6-bit character with an even parity bit. The remaining two channels contain zeroes:

1	2	3	4	5	6	7	8	9
1	3	5	■	P	■	2	4	6

In the UNIVAC III mode, three frames on tape represent 4 characters (24 bits) plus a sign bit as the 25th bit and two mod 3 check bits:

CHANNEL	1	2	3	4	5	6	7	8	9
Frame I	II 13	II 15	II 17	I 21	I 19	I 20	II 14	II 16	II 18
Frame II	III 7	III 9	III 11	P 27	I 22	P 26	III 8	III 10	III 12
Frame III	IV 1	IV 3	IV 5	V sign 25	I 23	I 24	IV 2	IV 4	IV 6

Roman numerals signify character positions of UNIVAC 1050 System's memory as recorded on tape. Arabic numerals signify the corresponding bit positions within a UNIVAC III word. The Control Unit when reading or writing tapes in this mode will convert between three frames on tape and four or five characters in the UNIVAC 1050 System's memory. When reading or writing in the 5-character option, the most significant bit of the 5th or least significant character of the group of five will correspond to the sign bit on tape. The remaining bits of this character will be ignored when writing tape and set to zero when reading from tape. When tapes are written in the 4-character option, bit 25 is always set to zero.

The Read Instruction starts the reading of a block of tape in a forward direction. When the instruction has been accepted, the Base Address Tetrad and the Character Block Count Tetrad are sent by the computer to the Control Unit. The Base Address is stored in the Memory Address Register of the Control Unit. The Block Count is stored in a counter in the Control Unit. As each character is read from tape it is stored into memory at the address furnished by the Control Unit. The Memory Address Register is incremented by 1 after each character enters the computer memory to establish the next memory address for depositing data. The character counter is decremented by 1. Transfer of data to the memory ceases when the counter is reduced to zero or the interblock gap is reached, whichever occurs first. Tape movement ceases upon reaching the interblock gap under either circumstance. When the last character has been read into memory, the Memory Address Register in the Control Unit is stored in the Read Address Record Tetrad. If the Advance Base Address option was specified in the instruction, the Control Unit also stores the Memory Address Register in the Memory Base Address Tetrad. The contents of the Control Unit's Memory Address Register when stored in the Tetrads is one greater than the address of the last data character accessed in memory.

The write instruction causes the writing of data stored in the memory. The starting address of the data to be written is located in Write Memory Address Tetrad and the number of characters to be written is located in the Write Character Block Count Tetrad. The Control Unit handling of memory requests is the same as in the read instructions except that writing terminates only when the character block count has been decremented to zero or a write fault condition is recognized.

The Rewind instruction causes the selected tape unit to rewind tape to the load point ready for further use. The instruction can be given on either tape channel. A further request specifying this tape unit during rewind results in an addressed unit busy error interrupt request.

The Rewind with Interlock instruction causes the Tape Unit to rewind to the unload tape point and makes it unavailable for further program use until the operator changes tapes or depresses the load button. This order may be given on either tape channel. A further request specifying this tape unit during or after rewind results in a non-ready error interrupt request.

The Contingency Backward Read instruction is only used to recover from a bad read condition which resulted while reading. It causes reading once an interblock gap has been recognized and stops reading when the next gap is recognized. Data enters the computer's memory under the same controls as in the Read Backward instruction.

Through the use of the Contingency Write instruction it is possible to write on the same tape after a Forward Read command. The instruction insures that the new block of tape will be written in an erased area. Because of the position of the read, write, and erase heads, the contingency pattern cannot write or erase on the portion of tape that lies between the read and write heads. Because of this a block may erroneously remain as a legitimate block if the block between the read and write heads is less than

450 characters if written in the UNIVAC compatible mode, or

600 characters if written in the UNIVAC III 4-character mode, or

750 characters if written in the UNIVAC III 5-character mode.

If the block between the read and write heads is as long as or longer than these limits no such error can occur.

TETRADS

Each Tape Control Unit requires two channels to accomplish reading and writing of tapes. For each channel there are three Tetrads used by the Control Unit.

The Read Tetrads are:

<table>
<tr><th colspan="3"></th><th>Tetrad Number</th></tr>
<tr><td>6
UNUSED</td><td>000</td><td>15
MEMORY BASE ADDRESS</td><td>48</td></tr>
<tr><td colspan="2">12
UNUSED</td><td>12
CHAR. BLOCK COUNT</td><td>49</td></tr>
<tr><td>6
UNUSED</td><td>000</td><td>15
READ ADDRESS RECORD</td><td>50</td></tr>
</table>

The Write Tetrads are:

<table>
<tr><td>6
UNUSED</td><td>000</td><td>15
MEMORY BASE ADDRESS</td><td>52</td></tr>
<tr><td colspan="2">12
UNUSED</td><td>12
CHAR BLOCK COUNT</td><td>53</td></tr>
<tr><td>6
UNUSED</td><td>000</td><td>15
WRITE ADDRESS RECORD</td><td>54</td></tr>
</table>

The special Read and Write Address Record Tetrads are used for automatic storage of a memory address one greater than the last accessed by the Control Unit during a tape read or write function. This occurs even on reads or writes resulting in an error condition. The base address is advanced only if the operation is successfully completed.

		SIGNIFICANCE					
	Value	Read	Write	Rewind, no Interlock	Rewind with Interlock	Interrupt at successful completion	Continue program during operation
FUNCTION	01	■				■	■
	02		■			■	■
	04			■		■	■
	010				■	■	■
	021	■					■
	022		■				■
	024			■			■
	030				■		■
	041	■				■	
	042		■			■	
	044			■		■	
	050				■	■	
	061	■					
	062		■				
	064			■			
	070				■		

		SIGNIFICANCE							
		Back-ward direc-tion (read only)	Contingency						
	Value		Read	Write	UNIVAC III/1107	UNIVAC III 4-char.	UNIVAC III 5-char.	Adv. Base Add.	Forward For-ward
DETAIL	00					■			■
	0100				■				■
	0200					■		■	■
	0300				■			■	■
	0400			■		■			■
	0500			■	■				■
	0600			■		■		■	■
	0700			■	■			■	■
	01000	■				■			
	01100	■			■				
	01200	■				■		■	
	01300	■			■			■	
	01400	■	■			■			
	01500	■	■		■				
	01600	■	■			■		■	
	01700	■	■		■			■	
	02000						■		■
	02200						■	■	■
	02400			■			■		■
	02600			■			■	■	■
	03000	■					■		
	03200	■					■	■	
	03400	■	■				■		
	03600	■	■				■	■	

	Value of the DETAIL FIELD	SIGNIFICANCE
		The following values of the DETAIL field represent indicators which may be addressed by the Test or Reset Indicators function. These values may be added together in any combination to allow testing or resetting of several indicators with a single instruction.
INDICATORS	02	End of tape detected
	010	Off-Line
	020	Memory Overload Occurred
	040	Parity Error (tape)
	0100	Parity Error (memory)
	01000	Unit not ready (off normal)
	02000	Addressed Unit busy
	04000	Memory Overload anticipated

UNISERVO III C Magnetic Tape Unit

The UNISERVO III C Magnetic Tape Control Unit controls writing and reading of magnetic tapes which are generated for or by the tape servos using the compatible tape reels. It allows read-compute and write-compute, but not read-write-compute. The Control Unit permits the writing and reading in two modes, translated and binary.

This Control Unit handles, in the translated mode, the translation required between the internal code and the even parity code. In the binary mode no translation is employed and data is stored on tape with an odd parity. The Control Unit recognizes only one function code on reading. This code is the tape mark (0001111) when read as a single character record between ¾″ inter-record gaps. All other codes when translated have a normal translation but no function definition. The Control Unit generates the writing of a single character block containing the tape mark by an instruction from the computer. The Control Unit can also cause tape to be advanced and erased for 5 inches in a forward direction *(skip - erase)*.

The Tape Control Unit checks when reading, or generates and checks while writing, the vertical 7th bit even or odd parity combinations. It also checks and generates the horizontal even parity character at the end of record. This parity character is preceded by a minimum of two blank characters which indicate end of record. The horizontal parity character is not read into memory.

The Control Unit transmits data between the 1050 Processor and the Compatible Tape Unit at a character rate of either 22.5 KC or 62.5.

The 22.5 KC character rate represent a recording density on tape of 200 pulses per inch at a speed of 112.5 inches per second: the 62.5 KC character rate represents a recording density on tape of 556 pulses per inch at a speed of 112.5 inches per second.

The Read instruction starts the reading of a block of tape in a forward direction. When the instruction has been accepted, the Base Address Tetrad and the Character Block Count Tetrad are sent by the computer to the Control Unit. The Base Address is stored in the Memory Address Register of the Control Unit. The Block Count is stored in a counter in the Control Unit. As each character is read from tape it is stored into memory at the address furnished by the Control Unit. The Memory Address Register is incremented by 1 after each character enters the computer memory to establish the next memory address for depositing data. The character counter is decremented by 1. Transfer of data to the memory ceases when the counter is reduced to zero or the interblock gap is reached, whichever occurs first. Tape movement ceases upon reaching the interblock gap under either circumstance. When the last character has been read into memory, the Memory Address Register in the Control Unit is stored in the Read Address Record Tetrad. If the Advance Base Address option was specified in the instruction, the Control Unit also stores the Memory Address Register in the Memory Base Address Tetrad. The contents of the Control Unit's Memory Address Register when stored in the Tetrads is one greater than the address of the last data character accessed in memory.

The Write instruction causes the writing of data stored in the memory. The starting address of the data to be written is located in Write Memory Address Tetrad and the number of characters to be written is located in the Write Character Block Count Tetrad. The Control Unit handling of memory requests is the same as in the Read instructions except that writing terminates only when the character block count has been decremented to zero or a write fault condition is recognized.

The Rewind instruction causes the selected Tape Unit to rewind tape to the load point ready for further use. The instruction can be given on either tape channel. A further request specifying this tape unit during rewind results in an addressed unit busy error interrupt request.

The Rewind with Interlock instruction causes the Tape Unit to rewind to the unload tape point and makes it unavailable for further program use until the operator changes tapes or depresses the load button. This order may be given on either tape channel. A further request specifying this tape unit during or after rewind results in a nonready error interrupt request.

TETRADS

Each Tape Control Unit requires two channels to accomplish reading and writing of tapes. For each channel there are three Tetrads used by the Control Unit.

The Read Tetrads are:

			Tetrad Number
6 UNUSED	000	15 MEMORY BASE ADDRESS	48
12 UNUSED		12 CHAR. BLOCK COUNT	49
6 UNUSED	000	15 READ ADDRESS RECORD	50

The Write Tetrads are:

6 UNUSED	000	15 MEMORY BASE ADDRESS	52
12 UNUSED		12 CHAR. BLOCK COUNT	53
6 UNUSED	000	15 WRITE ADDRESS RECORD	54

The special Read and Write Address Record Tetrads are used for automatic storage of a memory address one greater than the last accessed by the Control Unit during a tape read or write function. This occurs even on reads or writes resulting in an error condition. The base address is advanced only if the operation is successfully completed.

FUNCTION and DETAIL Field Values

	SIGNIFICANCE						
	Value	Read	Write	Rewind, no Interlock	Rewind with Interlock	Interrupt at successful completion	Continue program during operation
FUNCTION	01	■				■	■
	02		■			■	■
	04			■		■	■
	010				■	■	■
	021	■					■
	022		■				■
	024			■			■
	030				■		■
	041	■				■	
	042		■			■	
	044			■		■	
	050				■	■	
	061	■					
	062		■				
	064			■			
	070				■		

SIGNIFICANCE

	Value	Low Density	Advance Base Add.	Read or write untrans-lated mode	Backspace	Write Tape Mark	Skip-Erase
DETAIL	00	NORMAL CASE					
	0100			■			
	0200		■				
	0300		■	■			
	0400						■
	01000				■		
	02000	■					
	02100	■		■			
	02200	■	■				
	02300	■	■	■			
	04000					■	

		The following values of the DETAIL field represent indicators which may be addressed by the Test or Reset Indicators function. These values may be added together in any combination to allow testing or resetting of several indicators with a single instruction.
INDICATORS	01	Tape Mark detected (upon reading only)
	02	End of Tape detected
	040	Parity error (tape)
	0100	Parity error (memory)
	01000	Unit not ready (off-normal)
	02000	Addressed unit busy
	04000	Memory overload anticipated

UNIVAC 1050 SYSTEM

6. PROGRAM OPERATION ON UNIVAC 1050 SYSTEM

6. PROGRAM OPERATION ON UNIVAC 1050 SYSTEM

This portion of the UNIVAC 1050 reference manual will illustrate the procedures to be used in writing a program, assembling and debugging this program. The Processor Console is also explained.

The program that has been written demonstrates the ease of using PAL source language and the ease of incorporating diognostic macro-instructions and input-output routines.

CODING EXAMPLE

The operation programmed is a routine to edit card data and prepare a data tape, with UTMOST conventions, for use on the UNIVAC III System. Section totals are punched and the information contained on cards with improper action codes is printed.

Input Card

The input card (Figure 6-1) has the form:

Columns	Field	Tag Assigned
1-2	Section	CSECT
3-6	Account Number	CACNO
11-23	Name	CNAME
31-35	Amount	CAMNT
36	Action Code	CACTN
37-80	Unused	————

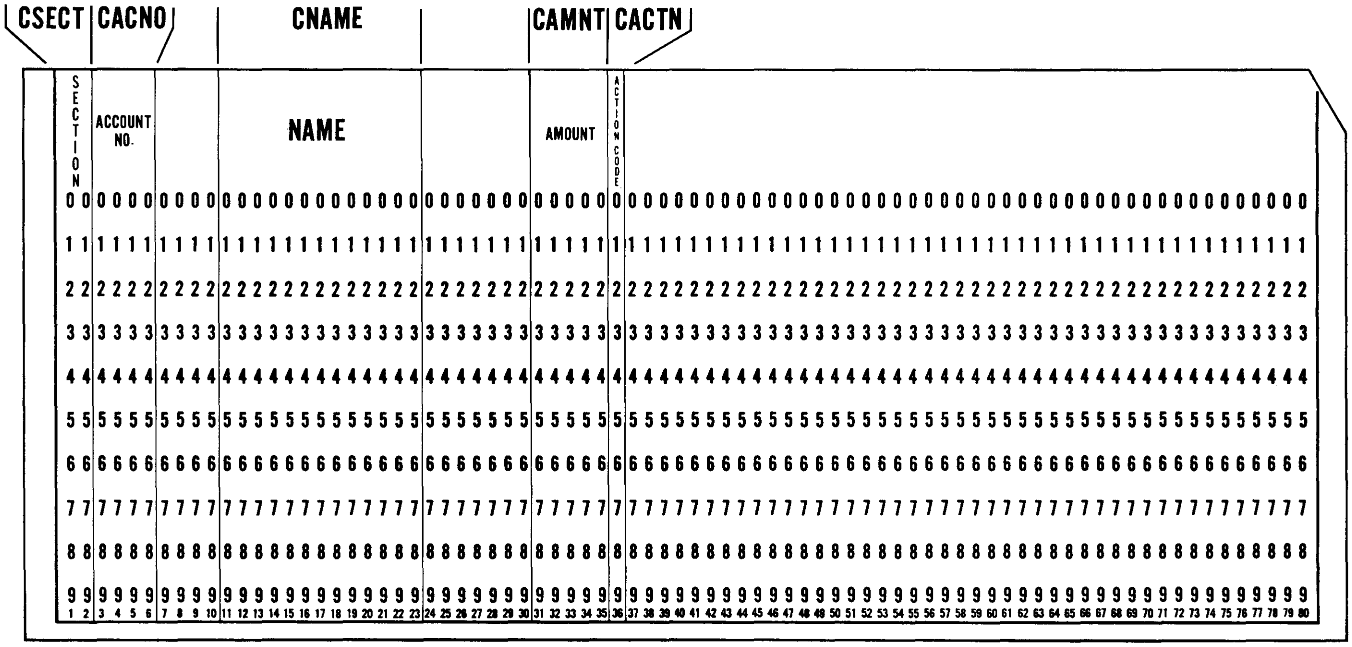


Figure 6-1. Input Card Format For Sample Problem.

The action code may be D for deposit or W for withdrawal; if the code is not D or W, the information on the card is printed (80-80) and is not recorded on tape. Section totals are punched on cards (Figure 6-2) with the format:

Columns	Field	Tag Assigned
1-2	Section	SECTN
4-10	Total	TOTAL

The end-of-run card sentinel contains six 9's in columns 1 through 6. No sections are numbered 99.

Output Tape

The output tape contains five records in each block, with 20 characters per record as follows:

Character	Field	Tag Assigned
1-2	Section	TSECT
3-6	Account Number	TACNO
7	Action Code	TACTN
8-15	Blanks	----
16-20	Amount	TAMNT

A block chart of the operation programmed appears in Figure 6-3. The coding written is shown in Figure 6-4; this is followed with a detailed analysis of the important lines of coding.

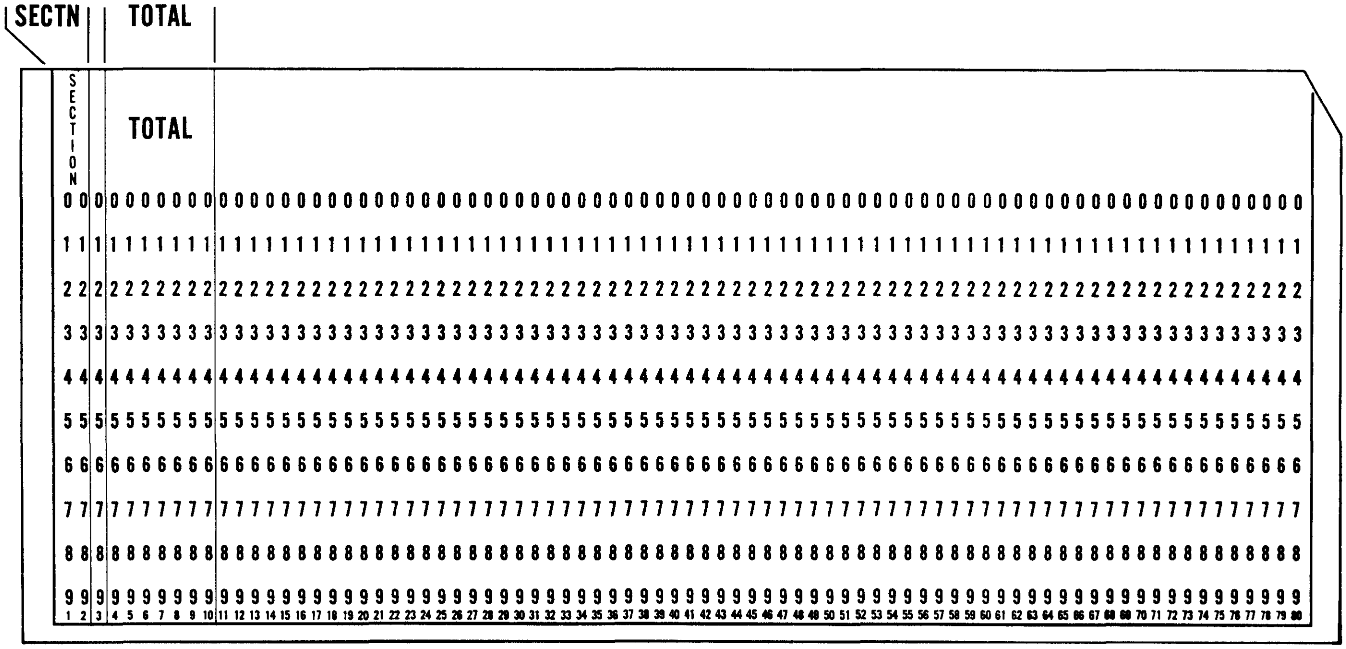


Figure 6-2. Section Total Cards For Sample Problem.

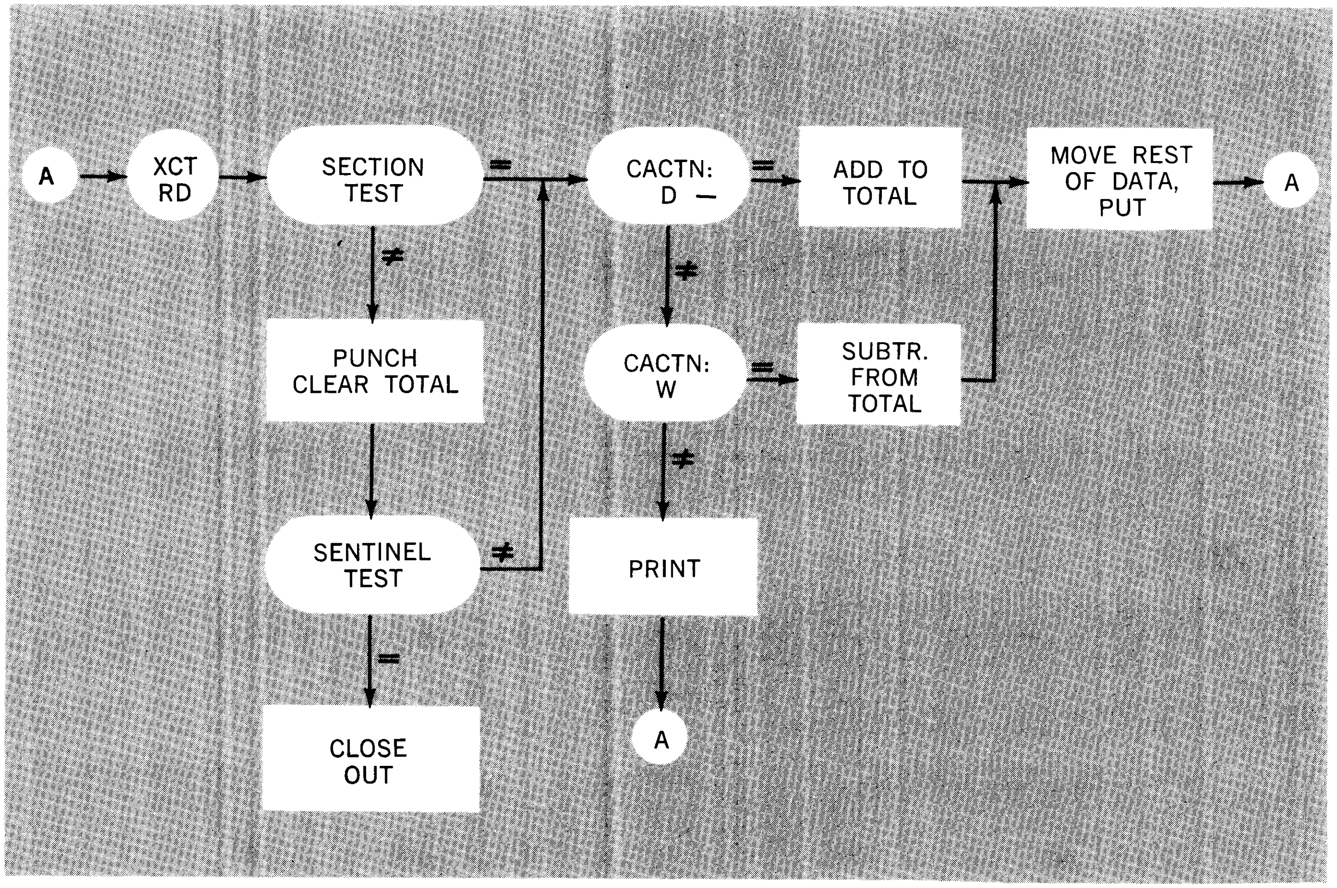
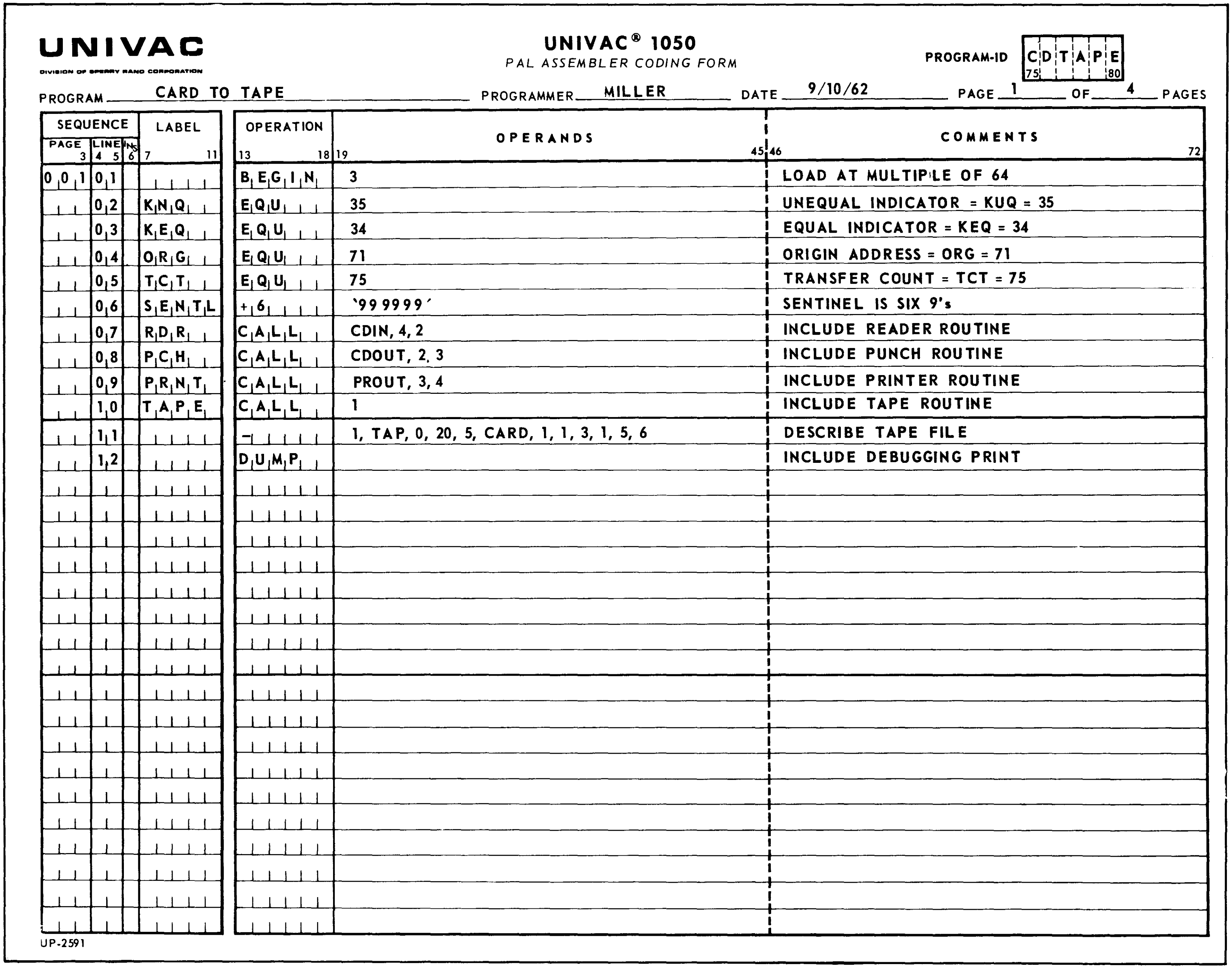


Figure 6-3. Sample Problem Block Chart.



UNIVAC

DIVISION OF SPERRY RAND CORPORATION

UNIVAC® 1050
PAL ASSEMBLER CODING FORM

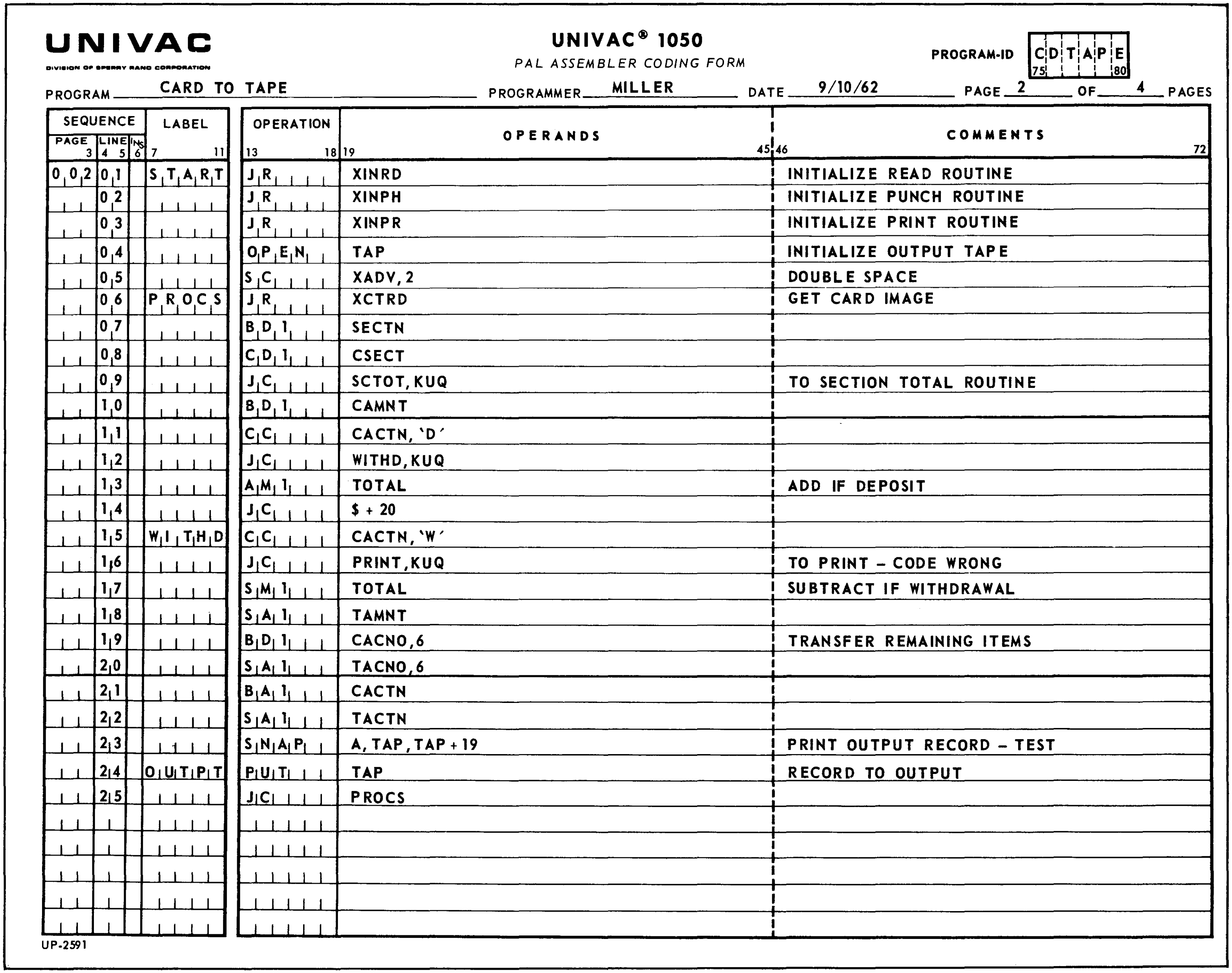
PROGRAM-ID CDTAPE (75–80)

PROGRAM CARD TO TAPE   PROGRAMMER MILLER   DATE 9/10/62   PAGE 1 OF 4 PAGES

SEQUENCE PAGE 3	LINE 4 5	INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 45	COMMENTS 46 72
001	01			BEGIN	3	LOAD AT MULTIPLE OF 64
	02		KNQ	EQU	35	UNEQUAL INDICATOR = KUQ = 35
	03		KEQ	EQU	34	EQUAL INDICATOR = KEQ = 34
	04		ORG	EQU	71	ORIGIN ADDRESS = ORG = 71
	05		TCT	EQU	75	TRANSFER COUNT = TCT = 75
	06		SENTL	+6	'999999'	SENTINEL IS SIX 9's
	07		RDR	CALL	CDIN, 4, 2	INCLUDE READER ROUTINE
	08		PCH	CALL	CDOUT, 2, 3	INCLUDE PUNCH ROUTINE
	09		PRNT	CALL	PROUT, 3, 4	INCLUDE PRINTER ROUTINE
	10		TAPE	CALL	1	INCLUDE TAPE ROUTINE
	11			—	1, TAP, 0, 20, 5, CARD, 1, 1, 3, 1, 5, 6	DESCRIBE TAPE FILE
	12			DUMP		INCLUDE DEBUGGING PRINT

UP-2591

Figure 6-4. Sample Problem Coding.



UNIVAC

DIVISION OF SPERRY RAND CORPORATION

UNIVAC® 1050

PAL ASSEMBLER CODING FORM

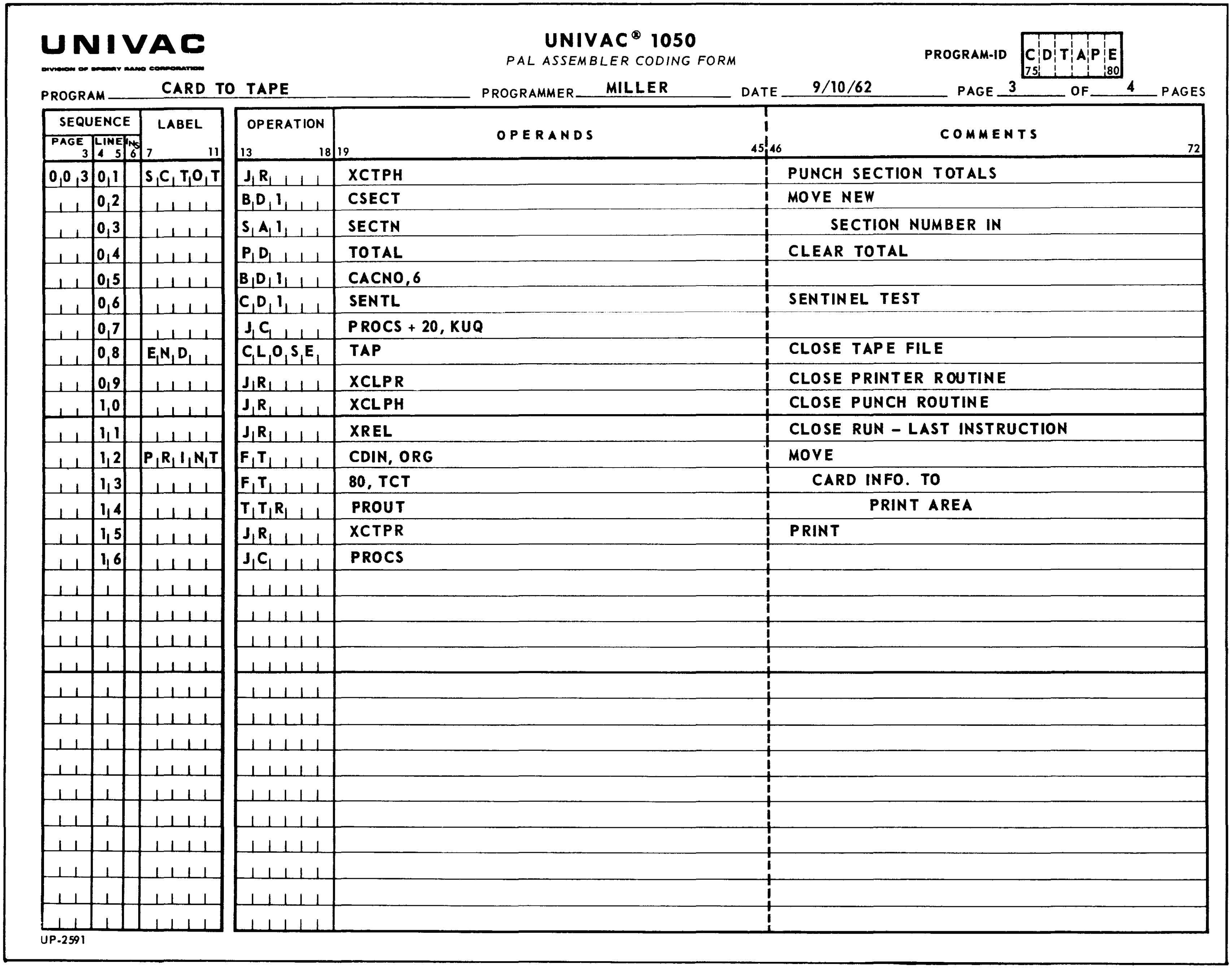
PROGRAM-ID CDTAPE (75–80)

PROGRAM CARD TO TAPE PROGRAMMER MILLER DATE 9/10/62 PAGE 2 OF 4 PAGES

SEQUENCE			LABEL	OPERATION	OPERANDS	COMMENTS
PAGE 3	LINE 4 5	INS 6	7 11	13 18	19 45	46 72
002	01		START	JR	XINRD	INITIALIZE READ ROUTINE
	02			JR	XINPH	INITIALIZE PUNCH ROUTINE
	03			JR	XINPR	INITIALIZE PRINT ROUTINE
	04			OPEN	TAP	INITIALIZE OUTPUT TAPE
	05			SC	XADV, 2	DOUBLE SPACE
	06		PROCS	JR	XCTRD	GET CARD IMAGE
	07			BD1	SECTN	
	08			CD1	CSECT	
	09			JC	SCTOT, KUQ	TO SECTION TOTAL ROUTINE
	10			BD1	CAMNT	
	11			CC	CACTN, 'D'	
	12			JC	WITHD, KUQ	
	13			AM1	TOTAL	ADD IF DEPOSIT
	14			JC	$ + 20	
	15		WITHD	CC	CACTN, 'W'	
	16			JC	PRINT, KUQ	TO PRINT – CODE WRONG
	17			SM1	TOTAL	SUBTRACT IF WITHDRAWAL
	18			SA1	TAMNT	
	19			BD1	CACNO, 6	TRANSFER REMAINING ITEMS
	20			SA1	TACNO, 6	
	21			BA1	CACTN	
	22			SA1	TACTN	
	23			SNAP	A, TAP, TAP + 19	PRINT OUTPUT RECORD – TEST
	24		OUTPT	PUT	TAP	RECORD TO OUTPUT
	25			JC	PROCS	

UP-2591

Figure 6-4. Sample Problem Coding (Continued).



UNIVAC
DIVISION OF SPERRY RAND CORPORATION

UNIVAC® 1050
PAL ASSEMBLER CODING FORM

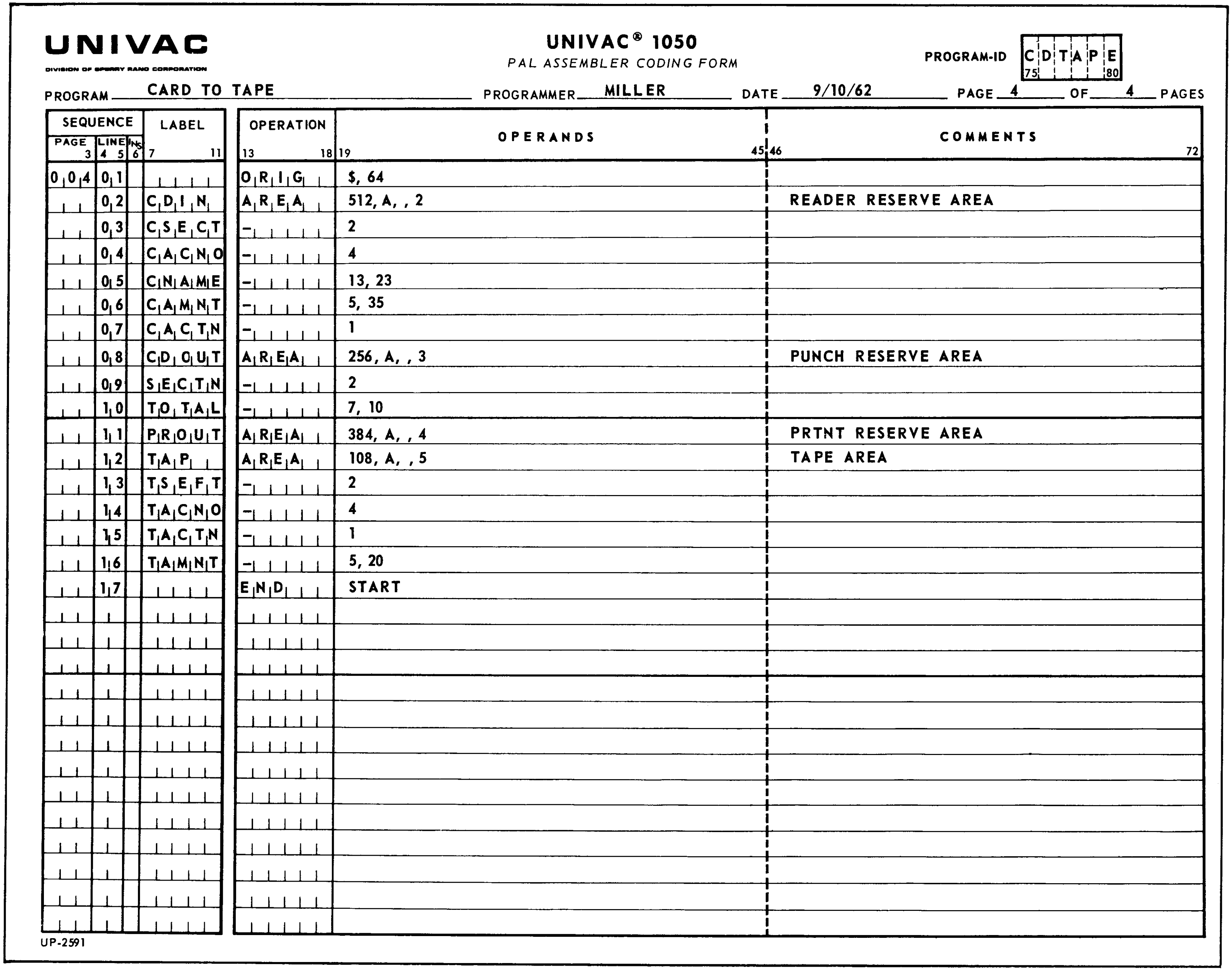
PROGRAM-ID C D T A P E

PROGRAM CARD TO TAPE PROGRAMMER MILLER DATE 9/10/62 PAGE 3 OF 4 PAGES

SEQUENCE PAGE 3	LINE 4 5	INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 45	COMMENTS 46 72
003	01		SCTOT	JR	XCTPH	PUNCH SECTION TOTALS
	02			BD1	CSECT	MOVE NEW
	03			SA1	SECTN	SECTION NUMBER IN
	04			PD	TOTAL	CLEAR TOTAL
	05			BD1	CACNO,6	
	06			CD1	SENTL	SENTINEL TEST
	07			JC	PROCS + 20, KUQ	
	08		END	CLOSE	TAP	CLOSE TAPE FILE
	09			JR	XCLPR	CLOSE PRINTER ROUTINE
	10			JR	XCLPH	CLOSE PUNCH ROUTINE
	11			JR	XREL	CLOSE RUN - LAST INSTRUCTION
	12		PRINT	FT	CDIN, ORG	MOVE
	13			FT	80, TCT	CARD INFO. TO
	14			TTR	PROUT	PRINT AREA
	15			JR	XCTPR	PRINT
	16			JC	PROCS	

UP-2591

Figure 6-4. Sample Problem Coding (Continued).



UNIVAC

DIVISION OF SPERRY RAND CORPORATION

UNIVAC® 1050

PAL ASSEMBLER CODING FORM

PROGRAM-ID CDTAPE (75–80)

PROGRAM CARD TO TAPE PROGRAMMER MILLER DATE 9/10/62 PAGE 4 OF 4 PAGES

SEQUENCE PAGE 3	LINE 4 5	INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 45	COMMENTS 46 72
004	01			ORIG	$, 64	
	02		CDIN	AREA	512, A, , 2	READER RESERVE AREA
	03		CSECT	-	2	
	04		CACNO	-	4	
	05		CNAME	-	13, 23	
	06		CAMNT	-	5, 35	
	07		CACTN	-	1	
	08		CDOUT	AREA	256, A, , 3	PUNCH RESERVE AREA
	09		SECTN	-	2	
	10		TOTAL	-	7, 10	
	11		PROUT	AREA	384, A, , 4	PRTNT RESERVE AREA
	12		TAP	AREA	108, A, , 5	TAPE AREA
	13		TSEFT	-	2	
	14		TACNO	-	4	
	15		TACTN	-	1	
	16		TAMNT	-	5, 20	
	17			END	START	

UP-2591

Figure 6-4. Sample Problem Coding (Continued).

Analysis of Coding Shown in Figure 6-4

Page 1

Line No.	Comment
1	Is a BEGIN directive indicating that the routine must be loaded starting at an address which is a multiple of 64.
2 to 5	Are EQU directives equating the tag addressed hardware features to the numeric equivalent.
6	Generates the data word SENTL which is six 9's.
7	Calls for the inclusion of a reader routine which controls four reserve areas using Index Register 2 and the area labeled CDIN. The four reserve areas are defined on Page 4, Line 2. This AREA is labeled CDIN, consists of 512 (4 x 128) characters and all fields within the area are to have Index Register 2 associated with them.
8 and 9	Call for a punch and printer routine; their respective areas are defined on Page 4, Line 8 and Page 4, Line 11.
10 and 11	Call for a tape routine and describe the tape file in question. The output tape file is named TAP, consists of 20 character records with 5 records per block. The name used in the label block is CARD, form 3 macro-instructions are used, the logical servo assignment is 1 and Index Register 5 is to be used. The data tape conventions of UTMOST Assembly System are to be used. Page 4, Line 12 describes the area for TAP.
12	Calls for the inclusion of a print routine which is used for debugging purposes.

Page 2

Line No.	Comment
1 to 5	Initialize the peripheral control routines.
6 to 25	Accomplish the desired testing and processing.

Page 2

Line No.	Comment
23	Is the diagnostic macro SNAP which will print the contents of the tape area after information has been transferred from the card area.

Page 3

1 to 4	Punch the section totals and reset the TOTAL to zero and reset the new section number.
5 to 7	Perform the sentinel test.
8 to 10	Close out the tape, printer and punch control routines.
11	Releases control to the Preload Section of the Loader.
12 to 14	Transfer the card information to the print area.
15 and 16	Print and return control to processing.

Page 4

Line No.	Comment
1	Is an ORIG directive which assures that the AREA following starts at an address which is a multiple of 64.
2	Defines the area for the card read routine. The length of the area must correspond to the number of reserve storages requested in the CALL for the reader routine. The index register for the fields within the area must also correspond to the one in the CALL for the reader routine.
3 to 7	Describe the format of the card.
8 to 10	Describe the card output area and format.
11	Describe the printer output area.
12 to 16	Describe the tape output area and format.
17	Is an END directive indicating that the first program step to be executed is START.

NORMAL COMPUTER OPERATION

Before the operating procedures involved in assembly, test running and correcting are demonstrated, some operations of the console should be mentioned. The control panel (Figure 6-5) provides the communications link between the Central Processor and the operator. The panel contains indicators to enable the operator to determine normal and abnormal conditions. The panel also allows the operator to access registers and storage locations when necessary. In addition, it contains buttons that enable the operator to correct or override error conditions, to manually insert or inhibit interrupts, and to manually set sense indicators for program use.

Start Up and Shut Down

The first operation necessary is that of turning the UNIVAC 1050 System on and shutting the system off. Two buttons are used for system start-up and close-down, SYSTEM ON and SYSTEM OFF.

- Depressing the SYSTEM ON button turns power on, the SYSTEM ON button will light. When the system is at full operating power, the SYSTEM OFF button will be extinguished.
- Depressing the SYSTEM OFF button removes power from the peripheral units and the Central Processor in an orderly fashion. While this power removal sequence is being completed, both the SYSTEM ON and SYSTEM OFF buttons will be lit. After completion, the SYSTEM ON button will be extinguished.

Program Start and Program Stop

Depression of the PROGRAM START button will illuminate the PROGRAM START button, extinguish the Parity Error indicator, and PROGRAM STOP button; and will permit the processor to proceed under control of the mode buttons.

Depression of the PROGRAM STOP button, or a programmed halt, will illuminate the PROGRAM STOP button and extinguish the PROGRAM START button. The processor will halt after completing the instruction in progress and staticizing the next instruction, or in the case of a programmed halt with the halt instruction staticized. Input-Output orders in progress will be completed; interrupt requests will be stored, unless inhibited.

If neither the PROGRAM START button nor the PROGRAM STOP button is lit, the processor is in a stall conditon.

Operating Mode

The six mutually exclusive mode control switches are used to control the operation of the processor in conjunction with the Start button.

LOAD CARD

This button allows the PROGRAM START button to initiate the binary reading of one card from the High-Speed Reader into memory, starting at the zero position of row 4. At the completion of the read cycle a Class I interrupt request will be stored. Normally the next action is to depress the desired mode button and the PROGRAM START button. The Parity Error indicator will not light.

LOAD TAPE

This button allows the PROGRAM START button to initiate the reading of one block of data from the tape unit designated 0 into memory, starting at the zero position of row 4. At the completion of the read cycle, with advance base address, a Class I interrupt request is stored. Normally the next action is to depress the desired mode button and the PROGRAM START button. The Parity Error indicator will not light.

ONE CYCLE

The One Cycle button allows for progression through a program or instruction one memory cycle at a time. Depression of the One Cycle button halts the processor within one or two cycles. Transference of data between the processor and input-output units continues until complete.

ONE INST

The One Instruction button allows for progression through programs by executing the instruction currently shown in the Instruction Register through the staticization of the next instruction except in the case of a programmed stop. Depression of the One Instruction button halts the processor after completing the staticization of an instruction.

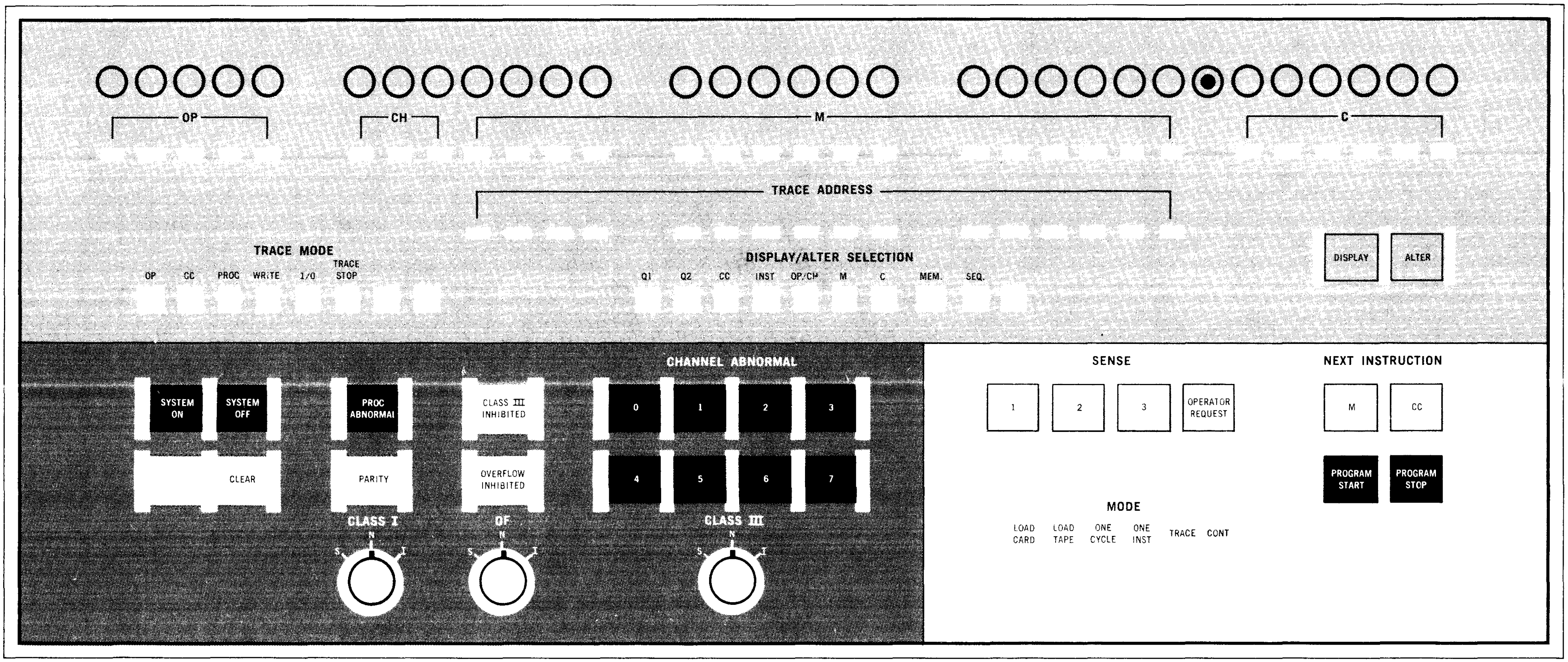


Figure 6-5. Central Processor Control Panel.

TRACE

This button allows selective halting of a running program in conjunction with the Trace Address Switches and the Op Alteration switches. The Trace Stop Indicator is lit when the Trace stop condition has been met.

CONT

The continuous switch allows for the normal running of a program.

CLEAR

Depression of this button provides for clearing stored interrupt requests, resetting all Jump Testable indicators to their normal state and extinguishing the Decimal Overflow Inhibited, Class III Interrupt Inhibited, Operator Interrupt Request and Channel Abnormal indicators. Depression of the Clear button also initiates a Control Panel Lamp Test; that is, all logic-driven indicators are illuminated and will remain so until the Clear button is released. The following indicators are not tested by the Clear button: System On; System Off; Sense buttons; Display, Alter and Trace Mode buttons; Display/Alter Selection buttons; and Mode buttons. The Trace Stop indicator is tested by the Clear button if any of the following mode switches is operated: One Cycle, One Instruction, and Trace.

SENSE

The three 2-position Sense buttons are used in conjunction with 3 jump tests to provide the Control Panel setting of a multiple of external conditions for use by the program via the jump tests.

This completes the description of the central processor buttons normally used in the operation of the UNIVAC 1050 system. The remainder of the console is described later in this section.

PAL ASSEMBLY OPERATING PROCEDURE (TAPE SYSTEM)

The system is first turned on, if necessary. The control panel is checked for abnormal indications, which are shown by the abnormal indicating lights. A check is also made of the position of interrupt switches and for peripherals which may be off line. If no abnormal condition exists the operator may continue.

The **PAL** Assembly System exists as a reel of magnetic tape, which is mounted on UNISERVO tape unit O. A blank tape with a Write Enable ring is also required by the system, and must be mounted on tape unit 1. Blank paper is inserted into the 1050 printer and blank cards are placed in the 1050 punch input hopper. The source code, which has been keypunched from the coding paper, is placed in the 1050 High Speed Reader.

After this setup procedure has been completed, the operator depresses the CLEAR button, the LOAD TAPE button and the PROGRAM START button. This operation will initial read the first block of information from the PAL Assembly System tape. The operator will then depress the CONT button and the PROGRAM START. This action causes the assembly to proceed and run to completion, producing a listing of the symbol table, memory map, source code and object code and a deck of cards representing the symbol table, memory map and object code. (See Programming section for formats.)

Loading and Test Running Operating Procedure

This portion consists of three separate operations:

- Loading Pre-load and Coordinator
- Loading object code
- Test Running

The first is accomplished by placing the Pre-Load and Coordinator deck (supplied by Univac) into the 1050 card reader input hopper, depressing the CLEAR, LOAD CARD and PROGRAM START buttons. Then depressing the CONT and PROGRAM START buttons.

Loading the object is just as simple. The object code deck (output from assembly) is placed in the reader input hopper and the PROGRAM START button is depressed. This accomplishes loading of the program.

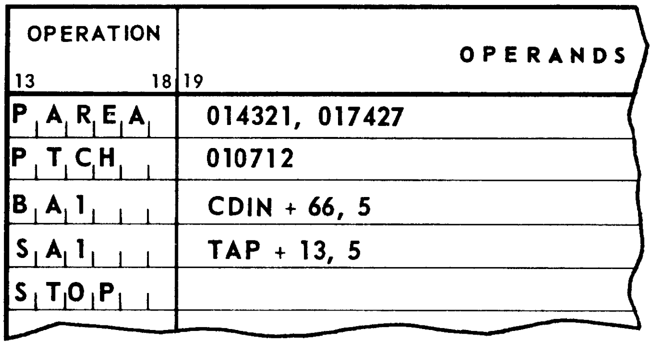
Test running will vary depending upon the program to be tested. For the example, a blank tape is mounted, blank cards are placed in the punch input hopper and blank paper placed in the printer. Test data cards are placed in the reader input hopper and then the PROGRAM START button is depressed.

The diagnostic macro-instruction SNAP (Line 23, page 2 of the coding), requesting that the tape record be printed before it is written, is the means of determining whether the program has been coded correctly. For this program, the total cards can be checked to determine if this portion of the coding has been done correctly. Test running a program with the aid of the powerful diagnostic macro-instructions provided in the PAL Assembly System is a very simple procedure.

Patch Assembler (Operating Instructions)

For purposes of illustrating the operation of the Patch Assembler, the example will be changed to include the transfer of the information from input card columns 63-67 to the output tape area positions 10-14.

This may be accomplished by providing the following input to the Patch Assembler.



OPERATION 13 18	OPERANDS 19
PAREA	014321, 017427
PTCH	010712
BA1	CDIN + 66, 5
SA1	TAP + 13, 5
STOP	

The PAREA line indicates that Positions 014321 to 017427 are available for patches to the object code. The PTCH entry of 010712 (assumed to be the position assigned to the line of coding on line 22, page 2 of the example shown in figure 6-4) indicates that the following lines of coding are to be inserted after the instruction at 010712. The next two lines indicate the coding to be inserted to transfer card columns 63-67 to tape characters 10-14. Those same cards can also be used to update the source code. The STOP line indicates to the Patch Assembler that this is the last of the input.

The loading procedure has been described previously and will not be repeated here. Blank cards are inserted into the punch input hopper and blank paper in the printer. The aforementioned correction cards, memory map and symbol table (output from PAL assembly) are placed in the reader input hopper; the PROGRAM START button is depressed to initiate the patch assembly. The output is a listing of the updated memory map, updated symbol table and corrections and a card deck consisting of an updated memory map and symbol table and object code correction cards. These object code correction cards are combined with the original PAL assembly output to form the updated object code.

SPECIAL COMPUTER OPERATION

In addition to the buttons described earlier (those used in normal operation) there are many special purpose buttons and procedures available to the 1050 operator. The following is a description of the remaining console buttons and how they may be used for special purposes.

Display Indicators

The thirty-one display indicators and the thirty associated alteration switches are used to display the contents of the instruction register, the control counter, selected storage locations, and data registers. In conjunction, the operator uses the Display Alter Selection Switches to specify the choice.

A parity indicator is provided to show the content of the parity bit when the contents of storage positions are displayed.

The M address portion of the display indicators is illuminated to show one of the following groupings:

- the indexed operand address of an instruction,
- the control counter address of the next instruction,
- the unit, function and half of the detail of the input-output instruction,
- the addresses of sequential storage locations to be displayed or altered under the direction of the Sequence Switch.

The display indicators are always illuminated to show a value designated by the Display/Alter Selection Switches.

Alteration Switches

This group of thirty switches is used to alter the contents of storage positions or registers. They are also used in the halting or tracing of running programs, or both, based upon specific operation codes. These switches are used in conjunction with the Display/Alter Selection and Trace Mode Switches, under control of the Mode Switches. No alteration occurs unless the Alter button is depressed.

The eight Op/Ch Code Switches are used to alter the Operation Code and the input-output channel designation in the instruction register. Five of these switches are also used in the Op Trace Mode for presetting the Operation Code reference value. The sixteen M Switches are used to change the Operand Address of a processor order or the Unit, Function and Detail designation of an input-output order in the instruction register.

The C Switches are used to alter the "Constant" or Tetrad Address of a processor instruction. These six switches are also used to introduce or change values in specified storage locations.

DISPLAY

Depression of this button causes information (see Display/Alter Selection) to be displayed, provided the processor has stopped.

ALTER

Depression of the button alters the content of indicators, registers or storage positions—selected by the Display/Alter Selection Switches—by the value in the thirty Alteration Switches used.

Display/Alter Selection

These nine mutually exclusive buttons operate with the Mode buttons and Display or Alter buttons to display or to alter (or both) specific registers and memory positions. When the Alter button is depressed, the value set in the Alteration Switches will be inserted into the selected register or storage position. The value of the contents of the appropriate registers, except for memory, is displayed when selected.

Q1 AND Q2

The Q1 and the Q2 buttons display internal indicators and registers in the thirty Display indicators.

CC

The CC button displays the contents of the control counter via the operand address portion of the Display indicators. The condition of the other indicators is a function of the instruction register.

INST

The Instruction button is used to alter the contents of the entire instruction register. The desired contents of the instruction register are placed in the thirty Alteration Switches.

OP/CH

The Op/Ch button alters the Op codes and the channel designation of the instruction register. The five most significant of the eight switches are used in the trace mode to stall a running program when an instruction containing an Op code equivalent to the setting of the switches is reached. The processor is stalled after the selected instruction is staticized. The transfer of data between the peripheral units and the processor will continue until the current operation has been completed.

M

The M button is used to alter the operand address of the instruction register.

C

The C button alters the least significant character of the instruction register.

MEM

The Memory button displays or alters the contents of the storage position designated by the position of the Operand Alteration Switches. The contents of the specified storage position are shown in the data portion of the display indicators after the Display button is depressed. If the Display button is depressed before the Alter button is depressed, the unaltered contents are displayed; if the Display button is depressed after the Alter button is depressed, the altered contents are displayed. In addition, the address portion of the Display indicators will receive the value of the Alteration switches +1.

SEQ

The Sequence button displays or alters the contents of sequential blocks of storage positions using the memory address shown in the address portion of the 30 display indicators. These are manually set to the desired first location address by use of the Memory and Alter buttons.

If the Sequence button is depressed, the operator may insert data into the storage address displayed by depressing the Alter button. Every operation of the Alter button increases the memory address by 1.

If the Sequence button is depressed, the operator may display the contents of the displayed memory address by depressing the Display button. Every operation of the Display button increases the memory address by 1.

DISPLAY THE CONTENTS OF STORAGE

Any storage position can be displayed when the processor has been brought to an orderly stop. The following procedure can be employed:

Display First Character

1. *Depress the CC—Display/Alter Selection.*
2. *Record the value of CC.*
3. *Depress the MEM—Display/Alter Selection button.*
4. *Set up the desired address in the M portion of the Alteration Switches.*
5. *Depress the Display button.*

The contents of the desired storage position will be displayed in the C portion of the display lights. The Address +1 set up in the M portion of the Alteration Switches will be displayed in the M display lights. This new address will be available for additional sequential displays.

Display Second and Subsequent Sequential Characters

6. *Depress SEQ Display/Alter Selection button.*
7. *Depress Display button.*

Repeat the last step for each new character in sequence to be displayed. The storage address is automatically incremented after each storage character has been displayed.

ALTER THE CONTENTS OF STORAGE

Any storage position can be altered when the processor has been brought to an orderly stop. The following procedure can be employed:

Alter First Character

1. *Depress the CC-Display/Alter Selection button.*
2. *Record the value of CC.*
3. *Depress MEM-Display/Alter Selection button.*
4. *Set up desired Address in the M portion of the Alteration Switch.*
5. *Set up the bit value of the character to be inserted in C portion of the Alteration Switches.*
6. *Depress the Alter button.*

The contents of the desired storage position will be filled with the character represented in the C Alteration Switches. The Address + 1 set up in the M portion of the Alteration Switches will be displayed in the M display lights. This new address will be available for additional sequential alterations.

Alter Second and Subsequent Sequential Characters

7. *Depress the SEQ Display/Alter Selection button.*
8. *Set up the desired character in C portion of the Alteration Switches.*
9. *Depress the Alter button.*

Repeat Steps 8 and 9 for each new character in sequence to be altered. The storage address is automatically incremented after each insertion.

PERFORMING INSTRUCTIONS VIA THE OPERATOR'S PROCESSOR CONTROL PANEL

The Processor Control Panel can be used to perform operator-created instructions for all instructions with the exception of the Jump Loop and the indexing functions.

1. *Depress the One Instruction-Mode button.*
2. *Depress the CC-Display/Alter Selection button.*
3. *Record the value of CC displayed in the M portion of the Display Lights.*
4. *Depress the Inst.-Display/Alter Selection button.*
5. *Record the value of the thirty display lights if required for a later operation.*
6. *Set up the new instruction in thirty Alteration Switches.*
7. *Depress the Alter button.*
8. *Depress the Program Start button.*

The new instruction will be performed instead of the instruction previously staticized in the instruction register. The processor, after completing this new instruction, will bring the next instruction stored at the address specified by the control counter at the end of executing the operator genrated instruction, and stop.

The reason for the special handling of a Jump Loop instruction is that when this instruction is staticized, the control counter has only been incremented four times rather than the usual five times.

Trace Mode

The five mutually exclusive buttons operate in conjunction with the Trace button found under the label *Mode.*

OP

Op trace setting provides for tracing based upon reaching a binary value in the Op Code equivalent to that preset in the five most significant of the eight Op Alteration Switches. When in the Trace Mode, the processor halts after staticizing the instruction associated with the specified operation code.

CC

CC trace setting provides for the tracing of a program based upon reaching a control counter value (during staticization) equivalent to the binary setting of the Trace Address Switches. When in the Trace Mode, the processor halts after staticizing the instruction whose address contains the binary value preset in the Trace Address Switches.

PROC

Processor trace setting provides for the tracing of a program based upon processor referencing any processor instruction or operand address equivalent to the binary setting of the Trace Address Switches. When in the Trace Mode, the processor halts upon reaching the address specified by the Trace Address Switches. (This mode detects processor, as opposed to input-output references to memory.)

WRITE

Write trace setting provides for the tracing of a program based upon any insertion to a memory address equivalent to the setting of the Trace Address switches (by either the processor a Control Unit). When in the Trace Mode, the processor halts upon reaching the address specified by the Trace Address Switches; however, the write operation will already have taken place.

I/O

I/O trace setting provides for the tracing of any Control Unit's reference to a memory address equivalent to the binary setting of the Trace Address Switches. When in the Trace Mode, the processor halts upon reaching the address specified by the Trace Address Switches. The Control Units complete their current functions even though the stopping of the processor instructions was initiated by a Control Unit memory address call.

TRACE STOP

This indicator lights when a trace stop condition has been met. This indicator works in conjunction with one of the following Mode buttons: One Cycle, One Instruction, and Trace.

Trace Address Switches

Each of the fifteen trace address switches has three positions — 0, 1, and neutral. The neutral position may be interpreted as 1 or zero. These switches are used to set the address at which the Processor will stop in the trace mode. Upon reaching the set address, the Processor will stop when the instruction has been staticized.

TRACE MODE OPERATIONS

The Trace Mode has been provided with many possible variations to assist the operator in tracing the use of storage positions by instructions or peripheral units and the use of particular operation codes. When in the trace mode, programs run normally without impedance until the desired condition has been met. The operator can either start off a program in the Trace Mode or stop a running program and restart it in the Trace Mode.

Preparation for Trace Mode

1. *Depress the Program Stop or One Inst. button. In either case, processing will be brought to an orderly halt with the preservation of operating controls.*

2. *If peripheral interrupts are not to operate, turn the Class III rotary switch to Inhibit to delay these interrupts. If normal interrupt*

processing is desirable, leave the switch in the Normal setting.

3. *Depress Trace button under Mode Control.*

4. *Depress the desired Trace Mode button; Op, CC, Proc, Write or I/O. All the options except Op are designed to trace the reference by the processor, or attached control units, to a particular storage address. The Op Trace Mode is used to trace the usage of particular instructions.*

5. *Set the Trace Address switches to the desired memory address to be traced. These switches have three settings 1, 0, or neutral. The neutral position is the equivalent of both 1 and 0. This allows more than one related memory position to be traced at the same time. When using the Op Trace Mode, the five Op Alter Switches are used for control of the Trace.*

6. *Depress the Program Start button.*

The program will now continue normally until the desired address or operation has been used or or performed. When this occurs, the program will be brought to a halt within a few cycles and the Trace indicator will be lit. (In most cases the program has stopped in the middle of an instruction without completing it.) To determine the position of the instruction, perform the following steps:

1. *Depress the CC button of the Display/Alter Selection button.*

2. *Write down the value in the M portion of the display lights. Knowing the value of the control counter allows restarting the program at this point if desired.*

3. *Depress One Inst. button.*

4. *Depress the Program Start button. The instruction which was stopped will be completed and the processor will stop at the end of staticization, before execution of the next instruction.*

5. *Write down the value of CC.*

These procedures insure that program location can be determined without destroying data or the normal results of processing. In the trace mode, the completion of a data transfer by a peripheral unit continues; only processing will have been brought to a stop. In all uses of the trace stop modes, further investigation of storage and other contingent conditions may be required.

NEXT INSTRUCTION

The Next Instruction Buttons are momentary switches labeled *M* and *CC*. Depression of a specific Next Instruction when a JC or JR instruction is in the register results in lighting the depressed indicator. Depression of the M button inserts binary zeros in the C portion of the instruction register.

Depression of the CC button when the processor is stopped forces a JC instruction into the instruction register and the processor to an end of staticize condition. The affected parts of the instruction word are:

The Op Code which is staticized to a JC Op Code

The C portion which is changed to a value to initiate the unconditional skip associated with the JC Op Code.

When the processor is restarted, this new instruction is performed. In this manner, any instruction may be skipped.

If a JC or JR instruction is staticized, and the processor is stopped, a depression of the M button forces the processor to take its next instruction address from the M address. In this manner, any jump instruction may be forced to follow the M path.

When the processor is stopped, and JC or JR instruction is staticized in the instruction register, the alterable testable indicators may be altered via the Control Panel. This is accomplished in the following manner:

1. *Depress the C button of the Display/Alter Selection buttons.*

2. *Arrange the six Alteration Switches to duplicate the bit combination of the desired test.*

3. *Depress the Alter button.*

If the combination inserted into the C portion of the instruction register is not a test function, the indicators are set or reset to the desired condition without the need to perform the instruction. If a test is required to change the indicators, the instruction must be performed by depression of the Start Switch.

Operator Controlled Branching

The operator, during debugging runs and on other occasions, may wish to direct programs to follow the opposite path on a jump instruction than would normally be pursued if present program decisions logic was allowed to hold. This can be accomplished by generating a completely new instruction within the instruction register or as described here by using the next instruction switch.

1. Instruction processing has been stopped by some means on a staticized conditional jump instruction.
2. Check the lights in the next instruction switch to determine where the next instruction is located, the M or CC address.
3. Depress the CC or M switch to establish directional path desired.
4. Depress the Program Start button.

PARITY

This indicator lights when a parity error has occurred; however, this excludes parity errors recognized during the use of memory by peripheral units. The button light may be extinguished by depression of the Program Start Button, or, when processing in the Class I normal mode, by program release of Class I Interrupt Inhibit.

PROC ABNORMAL

If any switch on the internal maintenance panel indicates an off-normal condition, this indicator will be illuminated and will go out without depression of the Program Start Switch when all switches are restored to normal. This indicator also lights when the parity check circuits detect an even parity in a character read from storage and the Parity Error indicator is already illuminated. In this case depression of the Program Start button will not start the processor until the abnormal indicator has been extinguished by depressing the abnormal indicator button.

CLASS III INHIBITED

This indicator lights when a programmed Class III Interrupt Inhibit is set and is extinguished when the inhibit is released by the program.

DECIMAL OVERFLOW INTERRUPT INHIBITED

This indicator lights when a programmed Decimal Overflow Interrupt Inhibit has been set, and is extinguished when the inhibit is released by the program.

Channel Abnormal

The Channel Abnormal Indicating buttons are momentary switches which light upon recognition of fault conditions in associated Control Units and peripherals. The depression of the switches extinguishes the lights and resets testable indicators. Those fault conditions arising at the peripheral units must be corrected at the units as well.

CLASS I

The Class I Interrupt Switch is a three position switch which provides three modes of interrupt operation: Normal, Inhibit, and Stall.

The normal position allows for the acceptance and storage of interrupt requests arising from parity errors and other processor abnormalties. One parity error lights the Parity Error indicator and forces transfer to the Class I interrupt entry channel, thereby enabling parity error processing. A processor abnormality or second parity error occurring while in the Class I Interrupt Mode causes the Processor Abnormal indicator to light and the computer to stall. The input-output units will complete their current functions.

The inhibit position provides the ability for the processor to override the acceptance and storage of interrupt requests resulting from parity error detection. If an interrupt request is present when the switch is placed in the inhibit position, the interrupt request is cleared. If a parity error is detected while processing in the inhibit position, the Parity Error indicator lights to indicate the detection of the parity error and remains lit until the Program Start button is depressed. When the Class I Interrupt Switch is in the inhibit position, a special program testable indicator is set. This indicator is reset when the switch is in Normal or stall positions.

The stall position allows for the halting of the computer immediately upon recognition of a parity error. The input-output units complete their current functions. The Parity Error indicator will be illuminated. There is no storage of the interrupt request. Depression of the Program Start button extinguishes the Parity Error indicator and allows the program to continue. Placing the switch in the stall position clears a stored Class I interrupt request.

DOF

The Decimal Overflow Switch, in conjunction with the Operator Request button provides Class II Interrupt control. It is a three position switch which provides three modes of decimal overflow operation. The positions are Normal, Inhibit, and Stall.

The normal position allows the acceptance of an interrupt request resulting from decimal overflow unless inhibited either by a program setting of Class II Interrupt Inhibit or an automatic interrupt request of equal or higher class which has not been released.

The inhibit position allows decimal overflow to occur without acting upon the resulting interrupt requests. Setting the switch to inhibit position does not clear an existing interrupt request nor does it affect the ability to set the decimal Overflow Indicator which stores such interrupt requests. When the Decimal Overflow (Class II Interrupt) switch is in the inhibit position or stall position, a special program testable indicator is set. This indicator is reset when the switch is placed in the normal position.

The stall position stops the processor when the instruction following the overflow is staticized. The program testable Overflow indicator will be set and the input-output units will complete their current functions with the storage of subsequent interrupt requests.

CLASS III

Class III Interrupt Switch is a three position switch which provides three modes of operation: Normal, Inhibit and Stall.

The normal position permits the storage and acceptance of interrupt requests originating from any input-output or control unit.

The inhibit position prevents acceptance of interrupts from occurring in any of the eight input-output channels. Interrupt requests will be stored until accepted or cleared.

The stall position halts the processor upon the processor acceptance of a Class III Input-Output Interrupt. The processor halts after referencing the associated Input-Output Channel Entry Point and staticizing a generated FT instruction containing the address of entrance to the specific Input-Output Interrupt Subroutine. The transference of data via input-output channels continues until completed. CC contains the address of the next instruction that would have been staticized if interrupt had not occurred.

OPERATOR REQUEST

The Operator Request Button is a momentary switch which permits Class II Interrupt requests to be originated at the console.

When an operator's interrupt request is stored, the indicator is extinguished and further operator requests are inhibited. Program testing of processor indicators determines the cause of the Class II Interrupt. Programmed release of the operator interrupt inhibit permits subsequent operator requests and illuminates the indicator. The programmed inhibit of operator interrupt extinguishes the indicator.

7. AUTOMATIC PROGRAM INTERRUPT

7. AUTOMATIC PROGRAM INTERRUPT

As well as serving as an efficient method for handling error and contingency conditions without stopping the Central Processor, Automatic Program Interrupt makes it possible to optimize the productivity of the input-output units and thus, the entire system. The occurrence of those conditions cause the setting of testable indicators which are acknowledged by the system's control circuitry. When an interrupt is recognized, control is transferred to a routine especially designed to deal with the condition causing the interrupt. At the time of interrupt, but before entering the appropriate interrupt routine, the contents of the control counter are stored in the Interrupt Entry Area—making it convenient to reenter the main stream of processing at the point of interruption. Since program interrupt is automatic, it relieves the programmer of the burden of periodically inserting in the program test instructions advising of a condition requiring prompt action because of timing considerations.

Automatic Program interrupt is separated into three classes (priorities) according to the importance of the function each performs. They are listed below and elaborated upon under the heading *Generation of Interrupt Requests.*

- **Class I** Error fault and Emergency conditions arising in the Central Processor are assigned the highest priority.
- **Class II** Decimal overflow and improper division in the system's arithmetic and control segments and operator interrupt constitute the second most important class of interrupt.
- **Class III** Transmittal of complete or incomplete actions by peripheral devices form the third and lowest priority class of Automatic Program Interrupt.

To effectively utilize Automatic Program Interrupt each channel entry point can store at least one interrupt request. A request remains stored until the processor accepts it or it is cleared manually. The acceptance of an interrupt request transfers control to the address provided in the last three positions of its associated fixed Interrupt Entry Area which comprises eight consecutive memory positions. The Interrupt Entry Area has the format

JC	K_1	K_2	K_3	I	M_1	M_2	M_3

The Interrupt Entry Area performs two basic functions. First it contains in positions M_1*, M_2 and M_3, the location of the first instruction of the proper interrupt routine; that is, the one associated with that particular channel entry point. The positions K_1**, K_2 and K_3 contain the location of the next step to be performed when returning to the main program; that is, the contents of the control counter at the time of interruption.

In addition, characters 1 through 5 normally constitute the Jump Test which releases the interrupt for the priority class associated with its channel Interrupt Entry Area. The index registers bit in character one will be set to binary zero.

Regardless of the class of interrupt, the following sequence of actions is executed whenever an interrupt request is accepted:

1. Except for Class I interrupt the instruction in progress at the time of Automatic Interrupt is completed.
2. The next instruction's address in the program being interrupted is stored in the Interrupt Entry Area associated with the channel which initiated the interrupt request. It is stored in positions K_1, K_2, and K_3 as shown in the diagram above.
3. The Processor enters the interrupt routine by transferring an address from the appropriate Interrupt Entry Area to the control counter. Moreover, a signal is generated which prevents the Processor from accepting additional interrupt requests from channels of the same or lower classes. While steps 2 and 3 are in progress, the processor will not recognize subsequent interrupt requests of any kind. At the completion of step 3, interrupts of higher classes are accepted.

SUMMARY

Upon completion of the current instruction, the Control Unit will automatically store the location of the next instruction of the interrupted program in the Interrupt Entry Area. Next it automatically loads into the control counter the address of the first step of the appropriate interrupt routine. The last three positions of the appropriate Interrupt Entry are filled at "load" time with the address of a routine whose purpose is to issue another action instruction, correct a recoverable error, or bring the system to an orderly halt if the error is not recoverable.

GENERATION OF INTERRUPT REQUESTS

A Class I interrupt request is generated upon recognition of memory parity errors detected when the processor control obtains and executes instructions. Parity errors detected when the memory is being used by peripheral Control Units are excluded from this definition.

When an internal parity error occurs, an ending pulse will be generated to terminate the current machine operation. If the processor is in Class I interrupt and an internal parity error occurs, the processor stalls unless the delete switch is set.

Class II interrupt requests are generated by Decimal Overflow and by Improper Division both of which set the testable Decimal Overflow indicator. The depression of the Operator Interrupt Switch of the console will also cause a Class II interrupt request to be issued. Upon acceptance of this request, the Automatic Class II Interrupt Inhibit and a program testable Operator Interrupt indicator will be set.

Determination of which of the two Class II interrupts caused the interrupt request is under program control.

For Class III interrupts, there is one Interrupt Entry per input-output channel. This enables acceptance of a Class III interrupt request to transfer control directly to the routine which controls the specific kind of unit involved.

Class III interrupt requests are generated by the following conditions in an input or output unit:

*The three most significant bits of M_1 are ignored.

**The three most significant bits of K_1 are set to zero.

Successful Completion Interrupt

- The normal termination of a request operation without detected errors.
- An interrupt request from a demand device without detected errors. A demand device is one which generates an interrupt request at fixed time intervals whether or not an instruction has been issued to the device.

Error Interrupt

Error interrupts occur when:

- A channel is in use and normal termination occurs but an error has been detected.
- A channel is in use and an error has been detected which will prevent normal termination.

Off Normal Interrupt

Off Normal interrupts occur and prevent instruction execution when an instruction is issued under the following conditions:

- The device has not completed a previously requested operation.
- An error or fault was detected while the device was not in use.
- A condition exists whereby the acceptance of the instruction would violate the rules governing the simultaneous use of input-output channels(memory overload).The purpose of these rules is to prevent the occurrence of an input-output data transfer rate which exceeds the memory data transfer rate. When an instruction requests an off-normal device, an interrupt request is generated and the instruction is disregarded.

INTERRUPT INHIBIT

Interrupt inhibits which deny the acceptance and in some cases the storage of interrupt requests can be set automatically upon acceptance of an interrupt request, by program, or manually by a switch on the operator's console.

Automatic Setting

For example, the occurrence of Class I interrupt (parity error) automatically sets an associated indicator which can be reset by a Jump Instruction (JC M, 30).

The occurrence of a Class II interrupt (Overflow or Operator Interrupt) sets an associated indicator which will prevent the acceptance but not the storage of other Class II and Class III interrupt requests. Automatic Interrupt Inhibit of this class can only be reset by the Release Automatic Interrupt Instruction (JC M, 29), and will not affect the setting of programmed or manual interrupt inhibit.

As soon as a Class III interrupt request is accepted the interrupt inhibit indicator is automatically set allowing subsequent Class III interrupt requests to be stored but not accepted until a Class III Automatic Release Interrupt Inhibit order (JC M, 25) has been issued. This instruction does not affect programmed or manual interrupt inhibit settings of the class, nor those of higher classes.

Programmed Interrupt Inhibit

This method of inhibiting interrupt applies only to Class III and II and is set by instruction. Class II operator and Decimal Overflow interrupt may be inhibited by instruction (JC M, 15 and JC M, 28 respectively). This inhibit can be released only by instruction (JC M, 14 or JC M, 31 respectively). The inhibit setting of the programmed Decimal Overflow interrupt is reflected in a program testable indicator.

Setting or resetting the programmed interrupt inhibit does not affect, nor is it affected by, the interrupt activities of any other class.

A Class III interrupt inhibit may be set by instruction (JC M, 26) to prevent the acceptance but not the storage of interrupt requests from any of the input-output Control Units. A program testable indicator will also be set to reflect this condition. This inhibit can only be released by instruction (JC M, 27). A Class III channel interrupt inhibit may be specified with an instruction to inhibit the acceptance of the interrupt request associated only with the successful completion and subsequent interrupts and not the error or off-normal condition resulting from that instruction. This channel inhibit applies only to the specific channel indicated and will not affect the storage or acceptance of interrupt requests originating on other Class III input-output channels. The Card Reader Control Unit will also accept a channel inhibit interrupt as a specific instruction, enabling the immediate inhibition of all further requests. Channel interrupt inhibit is released by the subsequent issuance of a specific channel input-output order that does not contain an associated inhibit indication.

Manual Interrupt Inhibit

A switch is provided which will prevent the acceptance and storage of interrupt requests due to Class I (internal parity) errors. If an interrupt request is present when the switch is placed in the inhibit or stall positions the stored request will be cleared. This switch is used to override internal parity errors and permit continuous processing or to stall the processor immediately upon recognition of such errors.

A switch for Class II interrupt is provided and will prevent acceptance but not storage of the Decimal Overflow Interrupt request. Setting or resetting this switch will not affect programmed or automatic interrupt inhibits of this or any other class. However, a program testable indicator is set to reflect this condition. The testable Overflow indicator is also unaffected by the use of this switch.

For Class III, a switch is provided to prevent the acceptance, but not the storage of interrupt request originating from the input-output Control Units. The setting or resetting of this switch will not affect the programmed Class III interrupt inhibit.

8. PERIPHERAL UNIT OPERATION

8. PERIPHERAL UNIT OPERATION

The input hoppers and the output stackers can easily be loaded or unloaded while the High-Speed Reader and the Card Punch Unit are operating. Paper forms can be quickly loaded and adjusted to a prerecorded position in the High-Speed Printer without the use of tools.

The control panels on the input-output units and on the Central Processor contain all of the controls and indicators required for efficient operation of the system as well as efficient program debugging and system maintenance. The few controls and indicators of concern to the operator are simple to use and monitor. Three sense switches are provided on the Central Processor panel to permit manual direction of the program.

The degree of operator participation in a UNIVAC 1050 System application is less than in other installations of a similar size. As a result the system's productive time is increased and the likelihood of erroneous operator intervention is minimized. Many functions normally performed manually by the operator of a system are automatically performed by the program in the UNIVAC 1050 System. This is achieved largely through the use of the same program interrupt technique that is employed to direct program control of input-output operations. When a need for operator action arises during normal operation of the system, it is detected automatically, and the program is directed to a standard routine designed to take appropriate action. For example, if an operator has neglected to load a High-Speed Reader input magazine before its cards are depleted, program control is directed to a routine that causes the Central Processor to come to an orderly stop. Then, the Central Processor is ready to continue in proper order as soon as the operator loads the empty input magazine and presses a button on the High-Speed Reader. Naturally, the operator has been clearly alerted to the occurrence of the condition by a non-ready light on the High-Speed Reader, and to the nature of the condition by indications on the panels of both the reader and the Central Processor.

The procedure to be executed for a given condition can be planned and programmed, insofar as practical, to be carried out automatically without on-the-spot decisions and improvisations by the operator. Thus, with the automatic program interrupt and contingency checking features of the UNIVAC 1050 System, an efficient balance is achieved between the functions of the operator and of the system itself.

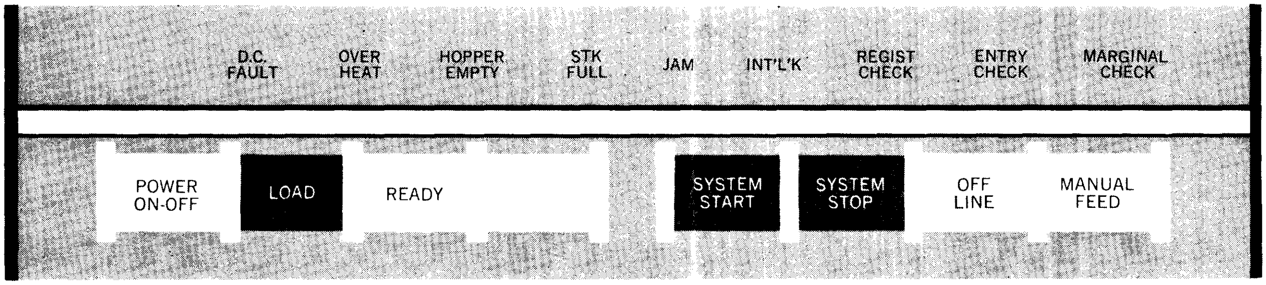


Figure 8-1. High-Speed Reader Control Panel.

High-Speed Reader Control Panel

DC FAULT

The presence of this light, when the power switch is on, indicates d.c. power failure; the drive motor is stopped.

OVERHEAT

This indicator will light when an abnormal temperature condition occurs; it stops the motor and removes all power except the power for the blowers and indicators.

HOPPER EMPTY (INPUT MAGAZINE)

This indicator will light when there are no cards in the feed magazine.

STK FULL

This indicator will light when there is a full capacity in any card stacker, except the normal stacker.

JAM

This indicator will light when a card jam occurs; stops the drive motor.

INTLK

This indicator will light when one of the following occurs:

(1) read assembly is raised or is improperly seated or (2) protective covers are not in place. The indicator will stop the drive motor and remove all power except the power for the blowers and indicators.

REGIST CHECK

When a registration failure occurs, the information is sent to the processor. Indication of this condition or interruption of machine operation is dependent upon processor control.

ENTRY CHECK

This indicator lights when a malfunction in feeding occurs; the drive motor is stopped.

READ CHECK

This indicator lights when a reader test error has occurred indicating that a read channel is in marginal condition. A maintenance check should be effected before further reading is attempted.

POWER ON-OFF

Depressing this button applies and removes all a.c. power.

HOPPER LOAD

Depressing this button initiates a load routine. This switch is interlocked in order that the Abnormal to the processor cannot be cleared until a magazine load is completed.

READY

After abnormal conditions have been corrected, this button should be depressed before operation of the reader can be resumed.

SYSTEM START

Depressing this button restarts the UNIVAC 1050 System from the reader location.

SYSTEM STOP

Depressing this button stops the UNIVAC 1050 System from the reader location.

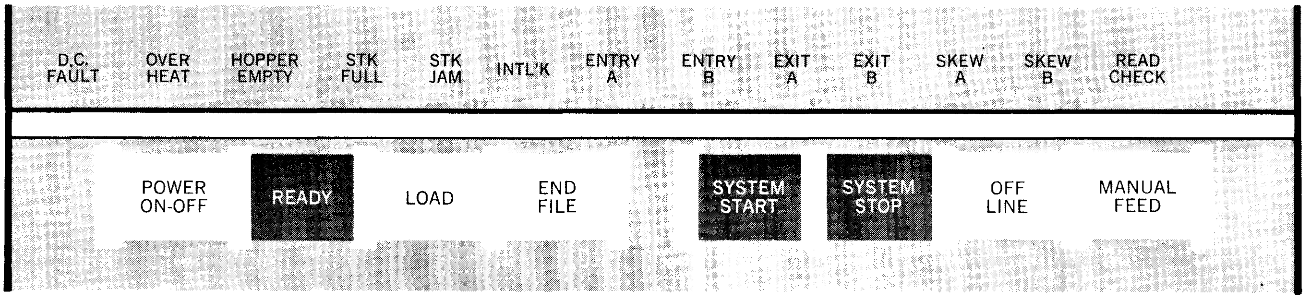


Figure 8-2. Card Punch Unit Control Panel.

OFF LINE

Depressing this button effectively disconnects the reader from the UNIVAC 1050 System.

MANUAL FEED

This button is for maintenance use only, and has no effect except when enabled by internal maintenance control.

STACKER RESET

This button is located next to the output stacker. Depressing it causes card output to be deposited in the normal stacker, provided this stacker is not full.

Card Punch Unit Control Panel

D.C. FAULT

The presence of this light when the power switch is on indicates d.c. power failure; the drive motor is stopped.

OVERHEAT

This indicator lights when an abnormal temperature condition has occurred; the drive motor is stopped.

HOPPER EMPTY (INPUT MAGAZINE)

This indicator lights when there are no cards in the feed magazine; the drive motor is stopped.

STK FULL

This indicator lights when a full capacity of either card stacker has occurred; the drive motor is stopped.

INTLK

This indicator lights when one or more of the following conditions occur:

- Punch assembly and upper card feed raised or improperly seated.
- Post-punch reading brushes not in place.
- Protective covers not in place.

The drive motor is stopped.

ENTRY A

This indicator lights when there is a late or missing card at station 2 (punch-wait station); the drive motor is stopped.

ENTRY B

This indicator lights when there is a late or missing card at post-punch station; the drive motor is stopped.

EXIT A

This indicator lights when there is a late card or failure of card to leave station 2 (punch-wait station); the drive motor is stopped.

EXIT B

This indicator lights when there is a late card or failure of card to leave post-punch station; the drive motor is stopped.

SKEW A

This indicator lights when a transverse card misalignment occurs while exiting first wait station; the drive motor is stopped.

SKEW B

This indicator lights when a transverse card misalignment occurs while exiting from post-punch station; the drive motor is stopped.

READ CHECK

This indicator lights when an incorrect data comparison occurs; the drive motor is not stopped.

POWER ON-OFF

Depression of this button causes the d.c. power supplies to be either activated or turned off.

READY

After abnormal conditions have been corrected, this button must be depressed before operation of the Card–Punch can be resumed.

LOAD

Depression of this button initiates load routines.

END FILE

Depression of this button initiates end file routines.

SYSTEM START

Depression of this button restarts the UNIVAC 1050 System from the punch location.

SYSTEM STOP

Depression of this button stops the UNIVAC 1050 System from the punch location.

OFF LINE

Depression of this button effectively disconnects the punch unit from the UNIVAC 1050 System and enables manual feed.

MANUAL FEED

Depression of this button causes manual feeding of a card one station when the unit is off line. Manual feeding does not prevent operation of the punch abnormal circuitry, and any abnormal condition occurring during manual feed cycles will stop the punch drive motor and light the specific light.

High-Speed Printer Control Panel

CARRIAGE OUT

This indicator lights when the carriage is not in printing position; the printer is placed in a non-ready condition.

D.C. FAULT

This indicator lights when a loss of d.c. power occurs; the printer is placed in a non-ready condition.

FORMS RUNAWAY

This indicator lights when the forms have advanced continously for more than 22 inches; the printer is placed in a non-ready condition.

FORMS OUT

This indicator lights when the printer has a minimum of 2½ inches of paper remaining below the print line; the printer is not placed in a non-ready condition, but printing is inhibited.

INTLK

This indicator lights when an open interlock has disabled the printer; the printer is not placed in a non-ready condition, but printing is inhibited.

RIBBON OUT

This indicator lights when either the ribbon is in a position to be changed or is physically removed from the printer; the printer is placed in a non-ready condition.

OVERHEAT

This indicator lights when an excessive heat condition is encountered; the printer is placed in a non-ready condition. All power is removed except power to the blowers and indicator lights.

POWER ON-OFF

Depression of this button provides the ability to turn power on and off. The indicator will be lit when power is on and not lit when power is off. When power is off, the printer is placed in a non-ready condition and the Ready indicator will be extinguished.

READY

The Ready button light is lit when all conditions necessary to operate the printer have been satisfied. When an abnormal condition is detected, the Ready indicator is extinguished. Depression of this button lights the Ready light and makes the printer available to the system after the abnormal condition is corrected.

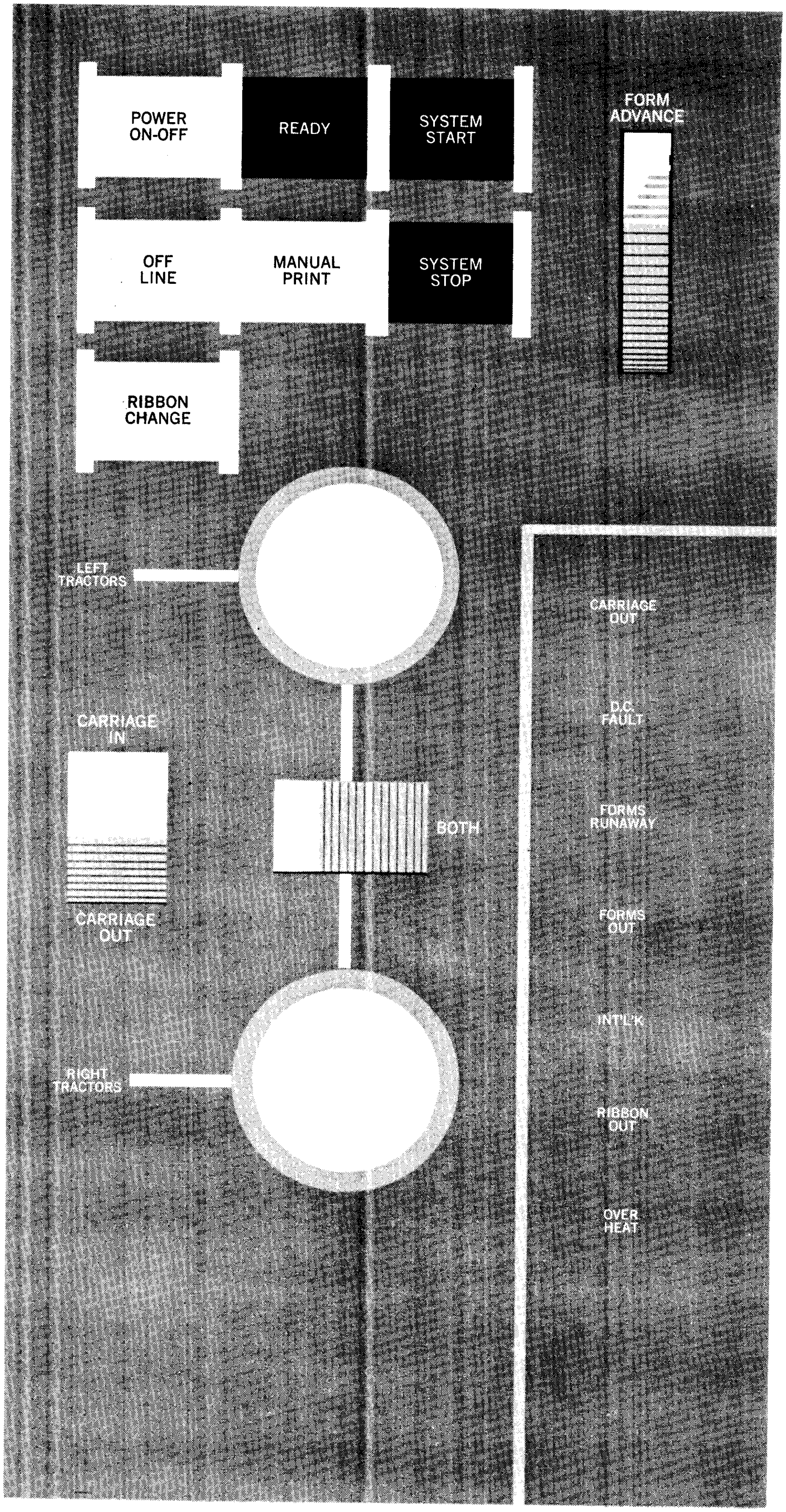


Figure 8-3. High-Speed Printer Control Panel.

SYSTEM START

Depression of this button starts the UNIVAC 1050 System from the printer location.

OFF LINE

Depression of this button effectively disconnects the printer from the UNIVAC 1050 System.

MANUAL PRINT

Depression of this button together with a programmed manual print operation provides the ability to override the paper low condition and print one line of stored information.

SYSTEM STOP

Depression of the button stops the system from the printer location.

RIBBON CHANGE

Depression of this button lights the change indicator light and causes the ribbon to be wound past the automatic ribbon reverse position on the take-up shaft. When the ribbon is wound past the reversing position, the printer is placed in the non-ready condition. Depressing this button once more extinguishes the indicator light and resets the ribbon change function.

LEFT TRACTORS

Depression of this button adjusts left pair of tractors so that the horizontal position of paper can be changed; if Both switch is in Both position, also adjusts right pair of tractors.

RIGHT TRACTORS

Depression of this button adjusts right pair of tractors so that the horizontal position of paper can be changed; if the BOTH switch is in BOTH position, also adjusts left pair of tractors.

CARRIAGE IN

Depression of this button moves the carriage away from the paper, lights the Carriage Out Indicator light, and extinguishes the Ready light. The printer is placed in the non-ready condition. Another depression of this button returns the carriage to the In position and extinguishes the Carriage Out indicator.

CODE WHEEL ERROR

This indicator lights when a character to be printed has an erroneous bit structure; printing will stop.

Uniservo III A and III C Tape Units Control Panel

NUMERAL

This numeral indicates the logical number of the UNISERVO Unit. The numeral indicator lights white when all conditions necessary to operate the tape unit have been satisfied; that is, the unit is ready for use. Any condition which makes the tape unit inoperable causes the indicator to light red.

AIRFLOW

This indicator lights whenever there is insufficient air in the blower system. Power to the tape unit is lost and it is placed in the non-ready condition.

OVERHEAT

This indicator lights when an excessive heat condition is encountered. The tape unit is placed in a non-ready condition. All power is lost except power to the blowers and indicator lights.

VOLTAGE

This indicator lights when a loss of d.c. or motor power occurs; the tape unit is placed in the non-ready condition.

FORWARD

This button light is provided to manually set the tape unit for forward tape movement.

This indication is present whenever the tape unit is set for the condition described above.

BACKWARD

This button light is provided to manually set the tape unit for backward tape movement.

This indication will be present whenever the tape unit is set for backward tape movement.

REWIND

This button light is provided to manually initiate a rewind operation. This indication is present whenever a rewind operation is in progress, whether manual or program initiated.

Upon completion of a manually initiated rewind operation, the tape will be in a condition where removal may take place.

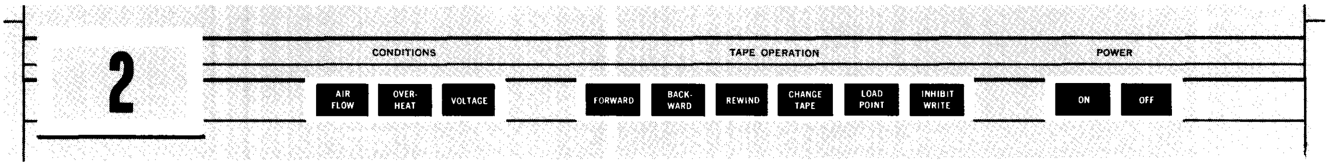


Figure 8-4. Uniservo III A and III C Tape Units Control Panel.

CHANGE TAPE

An indication is given whenever the program initiates a rewind with interlock. This indication wll remain present after the rewind with interlock operation is completed, at which time the tape will be in a condition where removal of the reel may take place.

After a rewind with interlock, when the reel is removed and a new reel loaded on the supply reel hub, closing of the tape unit window will remove the Change Tape indication, release the interlock, and cause tape to be positioned at its load point.

Switching is provided to manually remove the interlock and position the tape at its load point. When this manual intervention takes place, the Change Tape indication will be removed.

LOAD POINT

This indicator lights whenever the tape is positioned at the load point.

INHIBIT WRITE

This indicator will light when a reel mounted on the tape unit does not contain a Write Enable ring. Until this ring is inserted, a writing operation cannot take place. Insertion of this ring releases the write interlock. This indication is also present whenever there is no tape mounted on the tape unit.

POWER ON OFF

These button lights are provided to turn the tape unit power on and off.

UNIVAC
1050
SYSTEM

9. INSTALLATION SPECIFICATIONS

9. INSTALLATION SPECIFICATIONS

Unit	Depth	Width	Height	Weight	Power	BTU
CENTRAL PROCESSOR including Power Supply	26"[5]	115¼"[3]	67⅛"[4]	2,500	5.0 KVA[1]	13,648
HIGH-SPEED READER	25⅞"	38⅜"	41" + 15¾" card tray	650	2.0 KVA[7]	2,474
CARD-PUNCH UNIT	25⅞"	38⅜"	47"	800	2.5 KVA[7]	4,606
HIGH-SPEED PRINTER	31¼"	42⅝"	55"	1,200	2.7 KVA[7]	4,930
UNISERVO III A Tape Unit	29⅞"	30⅞"	63⅜"	810	2.75 KVA[2]	7,480
UNISERVO III C Tape Unit	29⅞"	30⅞"	63⅜"	810	2.75 KVA[2]	7,480
Tape Unit Control	25"	36"	55"	800	1.08 KVA[1]	2,968
Tape Unit Power Supply	26"[5]	46⅝"[6]	55"	1,800	½ of total UNISERVO power[2]	

Recommended operating temperature 70-75°	Max. 82° F	Min. 60° F
Recommended air humidity 50% RH	Max. 70% RH	Min. 40% RH

[1] Receives power through Central Processor power supply.

[2] Receives power through tape unit power supply.

[3] Width of 115¼" includes 17¼" drop leaf shelf.

[4] Height of 67⅛" inc'udes 12⅛" high operator's console on top of 55" processor cabinet.

[5] End panel dimensior front to back casework dimension is 25".

[6] Includes 1¼" space between Tape Unit Control and Tape Unit Power Supply.

[7] Includes 1.0 KVA for convenience outlet.

UNIVAC 1050 SYSTEM

APPENDIX A

APPENDIX A

OCTAL-DECIMAL CONVERSION TABLE

OCTAL 0000 to 0777 DECIMAL 0000 to 0511

	0	1	2	3	4	5	6	7
0000	0000	0001	0002	0003	0004	0005	0006	0007
0010	0008	0009	0010	0011	0012	0013	0014	0015
0020	0016	0017	0018	0019	0020	0021	0022	0023
0030	0024	0025	0026	0027	0028	0029	0030	0031
0040	0032	0033	0034	0035	0036	0037	0038	0039
0050	0040	0041	0042	0043	0044	0045	0046	0047
0060	0048	0049	0050	0051	0052	0053	0054	0055
0070	0056	0057	0058	0059	0060	0061	0062	0063
0100	0064	0065	0066	0067	0068	0069	0070	0071
0110	0072	0073	0074	0075	0076	0077	0078	0079
0120	0080	0081	0082	0083	0084	0085	0086	0087
0130	0088	0089	0090	0091	0092	0093	0094	0095
0140	0096	0097	0098	0099	0100	0101	0102	0103
0150	0104	0105	0106	0107	0108	0109	0110	0111
0160	0112	0113	0114	0115	0116	0117	0118	0119
0170	0120	0121	0122	0123	0124	0125	0126	0127
0200	0128	0129	0130	0131	0132	0133	0134	0135
0210	0136	0137	0138	0139	0140	0141	0142	0143
0220	0144	0145	0146	0147	0148	0149	0150	0151
0230	0152	0153	0154	0155	0156	0157	0158	0159
0240	0160	0161	0162	0163	0164	0165	0166	0167
0250	0168	0169	0170	0171	0172	0173	0174	0175
0260	0176	0177	0178	0179	0180	0181	0182	0183
0270	0184	0185	0186	0187	0188	0189	0190	0191
0300	0192	0193	0194	0195	0196	0197	0198	0199
0310	0200	0201	0202	0203	0204	0205	0206	0207
0320	0208	0209	0210	0211	0212	0213	0214	0215
0330	0216	0217	0218	0219	0220	0221	0222	0223
0340	0224	0225	0226	0227	0228	0229	0230	0231
0350	0232	0233	0234	0235	0236	0237	0238	0239
0360	0240	0241	0242	0243	0244	0245	0246	0247
0370	0248	0249	0250	0251	0252	0253	0254	0255
0400	0256	0257	0258	0259	0260	0261	0262	0263
0410	0264	0265	0266	0267	0268	0269	0270	0271
0420	0272	0273	0274	0275	0276	0277	0278	0279
0430	0280	0281	0282	0283	0284	0285	0286	0287
0440	0288	0289	0290	0291	0292	0293	0294	0295
0450	0296	0297	0298	0299	0300	0301	0302	0303
0460	0304	0305	0306	0307	0308	0309	0310	0311
0470	0312	0313	0314	0315	0316	0317	0318	0319
0500	0320	0321	0322	0323	0324	0325	0326	0327
0510	0328	0329	0330	0331	0332	0333	0334	0335
0520	0336	0337	0338	0339	0340	0341	0342	0343
0530	0344	0345	0346	0347	0348	0349	0350	0351
0540	0352	0353	0354	0355	0356	0357	0358	0359
0550	0360	0361	0362	0363	0364	0365	0366	0367
0560	0368	0369	0370	0371	0372	0373	0374	0375
0570	0376	0377	0378	0379	0380	0381	0382	0383
0600	0384	0385	0386	0387	0388	0389	0390	0391
0610	0392	0393	0394	0395	0396	0397	0398	0399
0620	0400	0401	0402	0403	0404	0405	0406	0407
0630	0408	0409	0410	0411	0412	0413	0414	0415
0640	0416	0417	0418	0419	0420	0421	0422	0423
0650	0424	0425	0426	0427	0428	0429	0430	0431
0660	0432	0433	0434	0435	0436	0437	0438	0439
0670	0440	0441	0442	0443	0444	0445	0446	0447
0700	0448	0449	0450	0451	0452	0453	0454	0455
0710	0456	0457	0458	0459	0460	0461	0462	0463
0720	0464	0465	0466	0467	0468	0469	0470	0471
0730	0472	0473	0474	0475	0476	0477	0478	0479
0740	0480	0481	0482	0483	0484	0485	0486	0487
0750	0488	0489	0490	0491	0492	0493	0494	0495
0760	0496	0497	0498	0499	0500	0501	0502	0503
0770	0504	0505	0506	0507	0508	0509	0510	0511

OCTAL 1000 to 1777 DECIMAL 0512 to 1023

	0	1	2	3	4	5	6	7
1000	0512	0513	0514	0515	0516	0517	0518	0519
1010	0520	0521	0522	0523	0524	0525	0526	0527
1020	0528	0529	0530	0531	0532	0533	0534	0535
1030	0536	0537	0538	0539	0540	0541	0542	0543
1040	0544	0545	0546	0547	0548	0549	0550	0551
1050	0552	0553	0554	0555	0556	0557	0558	0559
1060	0560	0561	0562	0563	0564	0565	0566	0567
1070	0568	0569	0570	0571	0572	0573	0574	0575
1100	0576	0577	0578	0579	0580	0581	0582	0583
1110	0584	0585	0586	0587	0588	0589	0590	0591
1120	0592	0593	0594	0595	0596	0597	0598	0599
1130	0600	0601	0602	0603	0604	0605	0606	0607
1140	0608	0609	0610	0611	0612	0613	0614	0615
1150	0616	0617	0618	0619	0620	0621	0622	0623
1160	0624	0625	0626	0627	0628	0629	0630	0631
1170	0632	0633	0634	0635	0636	0637	0638	0639
1200	0640	0641	0642	0643	0644	0645	0646	0647
1210	0648	0649	0650	0651	0652	0653	0654	0655
1220	0656	0657	0658	0659	0660	0661	0662	0663
1230	0664	0665	0666	0667	0668	0669	0670	0671
1240	0672	0673	0674	0675	0676	0677	0678	0679
1250	0680	0681	0682	0683	0684	0685	0686	0687
1260	0688	0689	0690	0691	0692	0693	0694	0695
1270	0696	0697	0698	0699	0700	0701	0702	0703
1300	0704	0705	0706	0707	0708	0709	0710	0711
1310	0712	0713	0714	0715	0716	0717	0718	0719
1320	0720	0721	0722	0723	0724	0725	0726	0727
1330	0728	0729	0730	0731	0732	0733	0734	0735
1340	0736	0737	0738	0739	0740	0741	0742	0743
1350	0744	0745	0746	0747	0748	0749	0750	0751
1360	0752	0753	0754	0755	0756	0757	0758	0759
1370	0760	0761	0762	0763	0764	0765	0766	0767
1400	0768	0769	0770	0771	0772	0773	0774	0775
1410	0776	0777	0778	0779	0780	0781	0782	0783
1420	0784	0785	0786	0787	0788	0789	0790	0791
1430	0792	0793	0794	0795	0796	0797	0798	0799
1440	0800	0801	0802	0803	0804	0805	0806	0807
1450	0808	0809	0810	0811	0812	0813	0814	0815
1460	0816	0817	0818	0819	0820	0821	0822	0823
1470	0824	0825	0826	0827	0828	0829	0830	0831
1500	0832	0833	0834	0835	0836	0837	0838	0839
1510	0840	0841	0842	0843	0844	0845	0846	0847
1520	0848	0849	0850	0851	0852	0853	0854	0855
1530	0856	0857	0858	0859	0860	0861	0862	0863
1540	0864	0865	0866	0867	0868	0869	0870	0871
1550	0872	0873	0874	0875	0876	0877	0878	0879
1560	0880	0881	0882	0883	0884	0885	0886	0887
1570	0888	0889	0890	0891	0892	0893	0894	0895
1600	0896	0897	0898	0899	0900	0901	0902	0903
1610	0904	0905	0906	0907	0908	0909	0910	0911
1620	0912	0913	0914	0915	0916	0917	0918	0919
1630	0920	0921	0922	0923	0924	0925	0926	0927
1640	0928	0929	0930	0931	0932	0933	0934	0935
1650	0936	0937	0938	0939	0940	0941	0942	0943
1660	0944	0945	0946	0947	0948	0949	0950	0951
1670	0952	0953	0954	0955	0956	0957	0958	0959
1700	0960	0961	0962	0963	0964	0965	0966	0967
1710	0968	0969	0970	0971	0972	0973	0974	0975
1720	0976	0977	0978	0979	0980	0981	0982	0983
1730	0984	0985	0986	0987	0988	0989	0990	0991
1740	0992	0993	0994	0995	0996	0997	0998	0999
1750	1000	1001	1002	1003	1004	1005	1006	1007
1760	1008	1009	1010	1011	1012	1013	1014	1015
1770	1016	1017	1018	1019	1020	1021	1022	1023

OCTAL-DECIMAL CONVERSION TABLE

OCTAL 2000 to 2777 | DECIMAL 1024 to 1535

	0	1	2	3	4	5	6	7
2000	1024	1025	1026	1027	1028	1029	1030	1031
2010	1032	1033	1034	1035	1036	1037	1038	1039
2020	1040	1041	1042	1043	1044	1045	1046	1047
2030	1048	1049	1050	1051	1052	1053	1054	1055
2040	1056	1057	1058	1059	1060	1061	1062	1063
2050	1064	1065	1066	1067	1068	1069	1070	1071
2060	1072	1073	1074	1075	1076	1077	1078	1079
2070	1080	1081	1082	1083	1084	1085	1086	1087
2100	1088	1089	1090	1091	1092	1093	1094	1095
2110	1096	1097	1098	1099	1100	1101	1102	1103
2120	1104	1105	1106	1107	1108	1109	1110	1111
2130	1112	1113	1114	1115	1116	1117	1118	1119
2140	1120	1121	1122	1123	1124	1125	1126	1127
2150	1128	1129	1130	1131	1132	1133	1134	1135
2160	1136	1137	1138	1139	1140	1141	1142	1143
2170	1144	1145	1146	1147	1148	1149	1150	1151
2200	1152	1153	1154	1155	1156	1157	1158	1159
2210	1160	1161	1162	1163	1164	1165	1166	1167
2220	1168	1169	1170	1171	1172	1173	1174	1175
2230	1176	1177	1178	1179	1180	1181	1182	1183
2240	1184	1185	1186	1187	1188	1189	1190	1191
2250	1192	1193	1194	1195	1196	1197	1198	1199
2260	1200	1201	1202	1203	1204	1205	1206	1207
2270	1208	1209	1210	1211	1212	1213	1214	1215
2300	1216	1217	1218	1219	1220	1221	1222	1223
2310	1224	1225	1226	1227	1228	1229	1230	1231
2320	1232	1233	1234	1235	1236	1237	1238	1239
2330	1240	1241	1242	1243	1244	1245	1246	1247
2340	1248	1249	1250	1251	1252	1253	1254	1255
2350	1256	1257	1258	1259	1260	1261	1262	1263
2360	1264	1265	1266	1267	1268	1269	1270	1271
2370	1272	1273	1274	1275	1276	1277	1278	1279
2400	1280	1281	1282	1283	1284	1285	1286	1287
2410	1288	1289	1290	1291	1292	1293	1294	1295
2420	1296	1297	1298	1299	1300	1301	1302	1303
2430	1304	1305	1306	1307	1308	1309	1310	1311
2440	1312	1313	1314	1315	1316	1317	1318	1319
2450	1320	1321	1322	1323	1324	1325	1326	1327
2460	1328	1329	1330	1331	1332	1333	1334	1335
2470	1336	1337	1338	1339	1340	1341	1342	1343
2500	1344	1345	1346	1347	1348	1349	1350	1351
2510	1352	1353	1354	1355	1356	1357	1358	1359
2520	1360	1361	1362	1363	1364	1365	1366	1367
2530	1368	1369	1370	1371	1372	1373	1374	1375
2540	1376	1377	1378	1379	1380	1381	1382	1383
2550	1384	1385	1386	1387	1388	1389	1390	1391
2560	1392	1393	1394	1395	1396	1397	1398	1399
2570	1400	1401	1402	1403	1404	1405	1406	1407
2600	1408	1409	1410	1411	1412	1413	1414	1415
2610	1416	1417	1418	1419	1420	1421	1422	1423
2620	1424	1425	1426	1427	1428	1429	1430	1431
2630	1432	1433	1434	1435	1436	1437	1438	1439
2640	1440	1441	1442	1443	1444	1445	1446	1447
2650	1448	1449	1450	1451	1452	1453	1454	1455
2660	1456	1457	1458	1459	1460	1461	1462	1463
2670	1464	1465	1466	1467	1468	1469	1470	1471
2700	1472	1473	1474	1475	1476	1477	1478	1479
2710	1480	1481	1482	1483	1484	1485	1486	1487
2720	1488	1489	1490	1491	1492	1493	1494	1495
2730	1496	1497	1498	1499	1500	1501	1502	1503
2740	1504	1505	1506	1507	1508	1509	1510	1511
2750	1512	1513	1514	1515	1516	1517	1518	1519
2760	1520	1521	1522	1523	1524	1525	1526	1527
2770	1528	1529	1530	1531	1532	1533	1534	1535

OCTAL 3000 to 3777 | DECIMAL 1536 to 2047

	0	1	2	3	4	5	6	7
3000	1536	1537	1538	1539	1540	1541	1542	1543
3010	1544	1545	1546	1547	1548	1549	1550	1551
3020	1552	1553	1554	1555	1556	1557	1558	1559
3030	1560	1561	1562	1563	1564	1565	1566	1567
3040	1568	1569	1570	1571	1572	1573	1574	1575
3050	1576	1577	1578	1579	1580	1581	1582	1583
3060	1584	1585	1586	1587	1588	1589	1590	1591
3070	1592	1593	1594	1595	1596	1597	1598	1599
3100	1600	1601	1602	1603	1604	1605	1606	1607
3110	1608	1609	1610	1611	1612	1613	1614	1615
3120	1616	1617	1618	1619	1620	1621	1622	1623
3130	1624	1625	1626	1627	1628	1629	1630	1631
3140	1632	1633	1634	1635	1636	1637	1638	1639
3150	1640	1641	1642	1643	1644	1645	1646	1647
3160	1648	1649	1650	1651	1652	1653	1654	1655
3170	1656	1657	1658	1659	1660	1661	1662	1663
3200	1664	1665	1666	1667	1668	1669	1670	1671
3210	1672	1673	1674	1675	1676	1677	1678	1679
3220	1680	1681	1682	1683	1684	1685	1686	1687
3230	1688	1689	1690	1691	1692	1693	1694	1695
3240	1696	1697	1698	1699	1700	1701	1702	1703
3250	1704	1705	1706	1707	1708	1709	1710	1711
3260	1712	1713	1714	1715	1716	1717	1718	1719
3270	1720	1721	1722	1723	1724	1725	1726	1727
3300	1728	1729	1730	1731	1732	1733	1734	1735
3310	1736	1737	1738	1739	1740	1741	1742	1743
3320	1744	1745	1746	1747	1748	1749	1750	1751
3330	1752	1753	1754	1755	1756	1757	1758	1759
3340	1760	1761	1762	1763	1764	1765	1766	1767
3350	1768	1769	1770	1771	1772	1773	1774	1775
3360	1776	1777	1778	1779	1780	1781	1782	1783
3370	1784	1785	1786	1787	1788	1789	1790	1791
3400	1792	1793	1794	1795	1796	1797	1798	1799
3410	1800	1801	1802	1803	1804	1805	1806	1807
3420	1808	1809	1810	1811	1812	1813	1814	1815
3430	1816	1817	1818	1819	1820	1821	1822	1823
3440	1824	1825	1826	1827	1828	1829	1830	1831
3450	1832	1833	1834	1835	1836	1837	1838	1839
3460	1840	1841	1842	1843	1844	1845	1846	1847
3470	1848	1849	1850	1851	1852	1853	1854	1855
3500	1856	1857	1858	1859	1860	1861	1862	1863
3510	1864	1865	1866	1867	1868	1869	1870	1871
3520	1872	1873	1874	1875	1876	1877	1878	1879
3530	1880	1881	1882	1883	1884	1885	1886	1887
3540	1888	1889	1890	1891	1892	1893	1894	1895
3550	1896	1897	1898	1899	1900	1901	1902	1903
3560	1904	1905	1906	1907	1908	1909	1910	1911
3570	1912	1913	1914	1915	1916	1917	1918	1919
3600	1920	1921	1922	1923	1924	1925	1926	1927
3610	1928	1929	1930	1931	1932	1933	1934	1935
3620	1936	1937	1938	1939	1940	1941	1942	1943
3630	1944	1945	1946	1947	1948	1949	1950	1951
3640	1952	1953	1954	1955	1956	1957	1958	1959
3650	1960	1961	1962	1963	1964	1965	1966	1967
3660	1968	1969	1970	1971	1972	1973	1974	1975
3670	1976	1977	1978	1979	1980	1981	1982	1983
3700	1984	1985	1986	1987	1988	1989	1990	1991
3710	1992	1993	1994	1995	1996	1997	1998	1999
3720	2000	2001	2002	2003	2004	2005	2006	2007
3730	2008	2009	2010	2011	2012	2013	2014	2015
3740	2016	2017	2018	2019	2020	2021	2022	2023
3750	2024	2025	2026	2027	2028	2029	2030	2031
3760	2032	2033	2034	2035	2036	2037	2038	2039
3770	2040	2041	2042	2043	2044	2045	2046	2047

OCTAL-DECIMAL CONVERSION TABLE

OCTAL 4000 to 4777 **DECIMAL 2048 to 2559**

	0	1	2	3	4	5	6	7
4000	2048	2049	2050	2051	2052	2053	2054	2055
4010	2056	2057	2058	2059	2060	2061	2062	2063
4020	2064	2065	2066	2067	2068	2069	2070	2071
4030	2072	2073	2074	2075	2076	2077	2078	2079
4040	2080	2081	2082	2083	2084	2085	2086	2087
4050	2088	2089	2090	2091	2092	2093	2094	2095
4060	2096	2097	2098	2099	2100	2101	2102	2103
4070	2104	2105	2106	2107	2108	2109	2110	2111
4100	2112	2113	2114	2115	2116	2117	2118	2119
4110	2120	2121	2122	2123	2124	2125	2126	2127
4120	2128	2129	2130	2131	2132	2133	2134	2135
4130	2136	2137	2138	2139	2140	2141	2142	2143
4140	2144	2145	2146	2147	2148	2149	2150	2151
4150	2152	2153	2154	2155	2156	2157	2158	2159
4160	2160	2161	2162	2163	2164	2165	2166	2167
4170	2168	2169	2170	2171	2172	2173	2174	2175
4200	2176	2177	2178	2179	2180	2181	2182	2183
4210	2184	2185	2186	2187	2188	2189	2190	2191
4220	2192	2193	2194	2195	2196	2197	2198	2199
4230	2200	2201	2202	2203	2204	2205	2206	2207
4240	2208	2209	2210	2211	2212	2213	2214	2215
4250	2216	2217	2218	2219	2220	2221	2222	2223
4260	2224	2225	2226	2227	2228	2229	2230	2231
4270	2232	2233	2234	2235	2236	2237	2238	2239
4300	2240	2241	2242	2243	2244	2245	2246	2247
4310	2248	2249	2250	2251	2252	2253	2254	2255
4320	2256	2257	2258	2259	2260	2261	2262	2263
4330	2264	2265	2266	2267	2268	2269	2270	2271
4340	2272	2273	2274	2275	2276	2277	2278	2279
4350	2280	2281	2282	2283	2284	2285	2286	2287
4360	2288	2289	2290	2291	2292	2293	2294	2295
4370	2296	2297	2298	2299	2300	2301	2302	2303
4400	2304	2305	2306	2307	2308	2309	2310	2311
4410	2312	2313	2314	2315	2316	2317	2318	2319
4420	2320	2321	2322	2323	2324	2325	2326	2327
4430	2328	2329	2330	2331	2332	2333	2334	2335
4440	2336	2337	2338	2339	2340	2341	2342	2343
4450	2344	2345	2346	2347	2348	2349	2350	2351
4460	2352	2353	2354	2355	2356	2357	2358	2359
4470	2360	2361	2362	2363	2364	2365	2366	2367
4500	2368	2369	2370	2371	2372	2373	2374	2375
4510	2376	2377	2378	2379	2380	2381	2382	2383
4520	2384	2385	2386	2387	2388	2389	2390	2391
4530	2392	2393	2394	2395	2396	2397	2398	2399
4540	2400	2401	2402	2403	2404	2405	2406	2407
4550	2408	2409	2410	2411	2412	2413	2414	2415
4560	2416	2417	2418	2419	2420	2421	2422	2423
4570	2424	2425	2426	2427	2428	2429	2430	2431
4600	2432	2433	2434	2435	2436	2437	2438	2439
4610	2440	2441	2442	2443	2444	2445	2446	2447
4620	2448	2449	2450	2451	2452	2453	2454	2455
4630	2456	2457	2458	2459	2460	2461	2462	2463
4640	2464	2465	2466	2467	2468	2469	2470	2471
4650	2472	2473	2474	2475	2476	2477	2478	2479
4660	2480	2481	2482	2483	2484	2485	2486	2487
4670	2488	2489	2490	2491	2492	2493	2494	2495
4700	2496	2497	2498	2499	2500	2501	2502	2503
4710	2504	2505	2506	2507	2508	2509	2510	2511
4720	2512	2513	2514	2515	2516	2517	2518	2519
4730	2520	2521	2522	2523	2524	2525	2526	2527
4740	2528	2529	2530	2531	2532	2533	2534	2535
4750	2536	2537	2538	2539	2540	2541	2542	2543
4760	2544	2545	2546	2547	2548	2549	2550	2551
4770	2552	2553	2554	2555	2556	2557	2558	2559

OCTAL 5000 to 5777 **DECIMAL 2560 to 3071**

	0	1	2	3	4	5	6	7
5000	2560	2561	2562	2563	2564	2565	2566	2567
5010	2568	2569	2570	2571	2572	2573	2574	2575
5020	2576	2577	2578	2579	2580	2581	2582	2583
5030	2584	2585	2586	2587	2588	2589	2590	2591
5040	2592	2593	2594	2595	2596	2597	2598	2599
5050	2600	2601	2602	2603	2604	2605	2606	2607
5060	2608	2609	2610	2611	2612	2613	2614	2615
5070	2616	2617	2618	2619	2620	2621	2622	2623
5100	2624	2625	2626	2627	2628	2629	2630	2631
5110	2632	2633	2634	2635	2636	2637	2638	2639
5120	2640	2641	2642	2643	2644	2645	2646	2647
5130	2648	2649	2650	2651	2652	2653	2654	2655
5140	2656	2657	2658	2659	2660	2661	2662	2663
5150	2664	2665	2666	2667	2668	2669	2670	2671
5160	2672	2673	2674	2675	2676	2677	2678	2679
5170	2680	2681	2682	2683	2684	2685	2686	2687
5200	2688	2689	2690	2691	2692	2693	2694	2695
5210	2696	2697	2698	2699	2700	2701	2702	2703
5220	2704	2705	2706	2707	2708	2709	2710	2711
5230	2712	2713	2714	2715	2716	2717	2718	2719
5240	2720	2721	2722	2723	2724	2725	2726	2727
5250	2728	2729	2730	2731	2732	2733	2734	2735
5260	2736	2737	2738	2739	2740	2741	2742	2743
5270	2744	2745	2746	2747	2748	2749	2750	2751
5300	2752	2753	2754	2755	2756	2757	2758	2759
5310	2760	2761	2762	2763	2764	2765	2766	2767
5320	2768	2769	2770	2771	2772	2773	2774	2775
5330	2776	2777	2778	2779	2780	2781	2782	2783
5340	2784	2785	2786	2787	2788	2789	2790	2791
5350	2792	2793	2794	2795	2796	2797	2798	2799
5360	2800	2801	2802	2803	2804	2805	2806	2807
5370	2808	2809	2810	2811	2812	2813	2814	2815
5400	2816	2817	2818	2819	2820	2821	2822	2823
5410	2824	2825	2826	2827	2828	2829	2830	2831
5420	2832	2833	2834	2835	2836	2837	2838	2839
5430	2840	2841	2842	2843	2844	2845	2846	2847
5440	2848	2849	2850	2851	2852	2853	2854	2855
5450	2856	2857	2858	2859	2860	2861	2862	2863
5460	2864	2865	2866	2867	2868	2869	2870	2871
5470	2872	2873	2874	2875	2876	2877	2878	2879
5500	2880	2881	2882	2883	2884	2885	2886	2887
5510	2888	2889	2890	2891	2892	2893	2894	2895
5520	2896	2897	2898	2899	2900	2901	2902	2903
5530	2904	2905	2906	2907	2908	2909	2910	2911
5540	2912	2913	2914	2915	2916	2917	2918	2919
5550	2920	2921	2922	2923	2924	2925	2926	2927
5560	2928	2929	2930	2931	2932	2933	2934	2935
5570	2936	2937	2938	2939	2940	2941	2942	2943
5600	2944	2945	2946	2947	2948	2949	2950	2951
5610	2952	2953	2954	2955	2956	2957	2958	2959
5620	2960	2961	2962	2963	2964	2965	2966	2967
5630	2968	2969	2970	2971	2972	2973	2974	2975
5640	2976	2977	2978	2979	2980	2981	2982	2983
5650	2984	2985	2986	2987	2988	2989	2990	2991
5660	2992	2993	2994	2995	2996	2997	2998	2999
5670	3000	3001	3002	3003	3004	3005	3006	3007
5700	3008	3009	3010	3011	3012	3013	3014	3015
5710	3016	3017	3018	3019	3020	3021	3022	3023
5720	3024	3025	3026	3027	3028	3029	3030	3031
5730	3032	3033	3034	3035	3036	3037	3038	3039
5740	3040	3041	3042	3043	3044	3045	3046	3047
5750	3048	3049	3050	3051	3052	3053	3054	3055
5760	3056	3057	3058	3059	3060	3061	3062	3063
5770	3064	3065	3066	3067	3068	3069	3070	3071

OCTAL-DECIMAL CONVERSION TABLE

OCTAL 6000 to 6777 **DECIMAL 3072 to 3583**

	0	1	2	3	4	5	6	7
6000	3072	3073	3074	3075	3076	3077	3078	3079
6010	3080	3081	3082	3083	3084	3085	3086	3087
6020	3088	3089	3090	3091	3092	3093	3094	3095
6030	3096	3097	3098	3099	3100	3101	3102	3103
6040	3104	3105	3106	3107	3108	3109	3110	3111
6050	3112	3113	3114	3115	3116	3117	3118	3119
6060	3120	3121	3122	3123	3124	3125	3126	3127
6070	3128	3129	3130	3131	3132	3133	3134	3135
6100	3136	3137	3138	3139	3140	3141	3142	3143
6110	3144	3145	3146	3147	3148	3149	3150	3151
6120	3152	3153	3154	3155	3156	3157	3158	3159
6130	3160	3161	3162	3163	3164	3165	3166	3167
6140	3168	3169	3170	3171	3172	3173	3174	3175
6150	3176	3177	3178	3179	3180	3181	3182	3183
6160	3184	3185	3186	3187	3188	3189	3190	3191
6170	3192	3193	3194	3195	3196	3197	3198	3199
6200	3200	3201	3202	3203	3204	3205	3206	3207
6210	3208	3209	3210	3211	3212	3213	3214	3215
6220	3216	3217	3218	3219	3220	3221	3222	3223
6230	3224	3225	3226	3227	3228	3229	3230	3231
6240	3232	3233	3234	3235	3236	3237	3238	3239
6250	3240	3241	3242	3243	3244	3245	3246	3247
6260	3248	3249	3250	3251	3252	3253	3254	3255
6270	3256	3257	3258	3259	3260	3261	3262	3263
6300	3264	3265	3266	3267	3268	3269	3270	3271
6310	3272	3273	3274	3275	3276	3277	3278	3279
6320	3280	3281	3282	3283	3284	3285	3286	3287
6330	3288	3289	3290	3291	3292	3293	3294	3295
6340	3296	3297	3298	3299	3300	3301	3302	3303
6350	3304	3305	3306	3307	3308	3309	3310	3311
6360	3312	3313	3314	3315	3316	3317	3318	3319
6370	3320	3321	3322	3323	3324	3325	3326	3327
6400	3328	3329	3330	3331	3332	3333	3334	3335
6410	3336	3337	3338	3339	3340	3341	3342	3343
6420	3344	3345	3346	3347	3348	3349	3350	3351
6430	3352	3353	3354	3355	3356	3357	3358	3359
6440	3360	3361	3362	3363	3364	3365	3366	3367
6450	3368	3369	3370	3371	3372	3373	3374	3375
6460	3376	3377	3378	3379	3380	3381	3382	3383
6470	3384	3385	3386	3387	3388	3389	3390	3391
6500	3392	3393	3394	3395	3396	3397	3398	3399
6510	3400	3401	3402	3403	3404	3405	3406	3407
6520	3408	3409	3410	3411	3412	3413	3414	3415
6530	3416	3417	3418	3419	3420	3421	3422	3423
6540	3424	3425	3426	3427	3428	3429	3430	3431
6550	3432	3433	3434	3435	3436	3437	3438	3439
6560	3440	3441	3442	3443	3444	3445	3446	3447
6570	3448	3449	3450	3451	3452	3453	3454	3455
6600	3456	3457	3458	3459	3460	3461	3462	3463
6610	3464	3465	3466	3467	3468	3469	3470	3471
6620	3472	3473	3474	3475	3476	3477	3478	3479
6630	3480	3481	3482	3483	3484	3485	3486	3487
6640	3488	3489	3490	3491	3492	3493	3494	3495
6650	3496	3497	3498	3499	3500	3501	3502	3503
6660	3504	3505	3506	3507	3508	3509	3510	3511
6670	3512	3513	3514	3515	3516	3517	3518	3519
6700	3520	3521	3522	3523	3524	3525	3526	3527
6710	3528	3529	3530	3531	3532	3533	3534	3535
6720	3536	3537	3538	3539	3540	3541	3542	3543
6730	3544	3545	3546	3547	3548	3549	3550	3551
6740	3552	3553	3554	3555	3556	3557	3558	3559
6750	3560	3561	3562	3563	3564	3565	3566	3567
6760	3568	3569	3570	3571	3572	3573	3574	3575
6770	3576	3577	3578	3579	3580	3581	3582	3583

OCTAL 7000 to 7777 **DECIMAL 3584 to 4095**

	0	1	2	3	4	5	6	7
7000	3584	3585	3586	3587	3588	3589	3590	3591
7010	3592	3593	3594	3595	3596	3597	3598	3599
7020	3600	3601	3602	3603	3604	3605	3606	3607
7030	3608	3609	3610	3611	3612	3613	3614	3615
7040	3616	3617	3618	3619	3620	3621	3622	3623
7050	3624	3625	3626	3627	3628	3629	3630	3631
7060	3632	3633	3634	3635	3636	3637	3638	3639
7070	3640	3641	3642	3643	3644	3645	3646	3647
7100	3648	3649	3650	3651	3652	3653	3654	3655
7110	3656	3657	3658	3659	3660	3661	3662	3663
7120	3664	3665	3666	3667	3668	3669	3670	3671
7130	3672	3673	3674	3675	3676	3677	3678	3679
7140	3680	3681	3682	3683	3684	3685	3686	3687
7150	3688	3689	3690	3691	3692	3693	3694	3695
7160	3696	3697	3698	3699	3700	3701	3702	3703
7170	3704	3705	3706	3707	3708	3709	3710	3711
7200	3712	3713	3714	3715	3716	3717	3718	3719
7210	3720	3721	3722	3723	3724	3725	3726	3727
7220	3728	3729	3730	3731	3732	3733	3734	3735
7230	3736	3737	3738	3739	3740	3741	3742	3743
7240	3744	3745	3746	3747	3748	3749	3750	3751
7250	3752	3753	3754	3755	3756	3757	3758	3759
7260	3760	3761	3762	3763	3764	3765	3766	3767
7270	3768	3769	3770	3771	3772	3773	3774	3775
7300	3776	3777	3778	3779	3780	3781	3782	3783
7310	3784	3785	3786	3787	3788	3789	3790	3791
7320	3792	3793	3794	3795	3796	3797	3798	3799
7330	3800	3801	3802	3803	3804	3805	3806	3807
7340	3808	3809	3810	3811	3812	3813	3814	3815
7350	3816	3817	3818	3819	3820	3821	3822	3823
7360	3824	3825	3826	3827	3828	3829	3830	3831
7370	3832	3833	3834	3835	3836	3837	3838	3839
7400	3840	3841	3842	3843	3844	3845	3846	3847
7410	3848	3849	3850	3851	3852	3853	3854	3855
7420	3856	3857	3858	3859	3860	3861	3862	3863
7430	3864	3865	3866	3867	3868	3869	3870	3871
7440	3872	3873	3874	3875	3876	3877	3878	3879
7450	3880	3881	3882	3883	3884	3885	3886	3887
7460	3888	3889	3890	3891	3892	3893	3894	3895
7470	3896	3897	3898	3899	3900	3901	3902	3903
7500	3904	3905	3906	3907	3908	3909	3910	3911
7510	3912	3913	3914	3915	3916	3917	3918	3919
7520	3920	3921	3922	3923	3924	3925	3926	3927
7530	3928	3929	3930	3931	3932	3933	3934	3935
7540	3936	3937	3938	3939	3940	3941	3942	3943
7550	3944	3945	3946	3947	3948	3949	3950	3951
7560	3952	3953	3954	3955	3956	3957	3958	3959
7570	3960	3961	3962	3963	3964	3965	3966	3967
7600	3968	3969	3970	3971	3972	3973	3974	3975
7610	3976	3977	3978	3979	3980	3981	3982	3983
7620	3984	3985	3986	3987	3988	3989	3990	3991
7630	3992	3993	3994	3995	3996	3997	3998	3999
7640	4000	4001	4002	4003	4004	4005	4006	4007
7650	4008	4009	4010	4011	4012	4013	4014	4015
7660	4016	4017	4018	4019	4020	4021	4022	4023
7670	4024	4025	4026	4027	4028	4029	4030	4031
7700	4032	4033	4034	4035	4036	4037	4038	4039
7710	4040	4041	4042	4043	4044	4045	4046	4047
7720	4048	4049	4050	4051	4052	4053	4054	4055
7730	4056	4057	4058	4059	4060	4061	4062	4063
7740	4064	4065	4066	4067	4068	4069	4070	4071
7750	4072	4073	4074	4075	4076	4077	4078	4079
7760	4080	4081	4082	4083	4084	4085	4086	4087
7770	4088	4089	4090	4091	4092	4093	4094	4095

UNIVAC
1050
SYSTEM

APPENDIX B

APPENDIX B

INDICATORS

00-31 Unconditional Jump to M Address E = 0

00 Unconditional Jump
14 Release Operator Interrupt Inhibit and jump
15 Set Operator Interrupt Inhibit and jump
16 Stop, Jump when Console Restart Button is depressed
17 Set Tracing Stall and Jump
18 Set Sense Indicator 1 to 1 and jump
19 Set Sense Indicator 2 to 1 and jump
20 Set Sense Indicator 3 to 1 and jump
21 Set Sense Indicator 1 to 0 and jump
22 Set Sense Indicator 2 to 0 and jump
23 Set Sense Indicator 3 to 0 and jump
24 Unconditional Jump
*25 Release Class 3 Interrupt Inhibit
26 Set I/O Interrupt Inhibit and jump (Class 3)
27 Release I/O Interrupt Inhibit and jump (Class 3) (Resets Programmed Inhibit Only)
28 Set Decimal Overflow Interrupt Inhibit and jump (Class 2)
*29 Release Class 2 Interrupt Inhibit
*30 Release Processor Parity or Abnormal Interrupt Inhibit and jump (Class 1)
31 Release Decimal Overflow Interrupt Inhibit and jump (Class 2), (Resets Programmed Inhibit Only)

32-63 Conditional Jump E = 1

Exceptions to conditional jump are 32, 41, 42, 48, and 56. The status of the indicators is unaltered by the JC and JR instructions except as shown.

32 (KNO) NOOP
33 (KHI) High
34 (KEQ) Equal
35 (KUQ) Unequal
36 (KLO) Low

These four indicators are affected by the comparison instructions: CC, LT, CD, CB, CT.

37 (KZR) Result of last arithmetic operation was zero
38 (KM) Result of last decimal arithmetic operation was negative.
39 (KNO) No overflow in last binary operation
40 (KDF) Decimal Overflow occurred since last test. If the indicator is set to 1, reset it to 0 and jump.
41 Store Indicators 33-40 in M_x memory position and proceed to next instruction
42 Set Indicators 33-40 from M_x memory position and proceed to next instruction
43 Input-Output status test found indicator(s) set to 1
44 Test and reset operator interrupt request
45 Input-Output Interrupt is inhibited (Class 3)
47 Decimal Overflow Interrupt is inhibited (Class 2)
48 Stop/Go to control counter when console start is depressed, ignore M used for display.
49 Processor Parity and Abnormal Interrupt is inhibited (Class 1) (Manual Switch Only)
50 Sense Switch 1 on console is ON
51 Sense Switch 2 on console is ON
52 Sense Switch 3 on console is ON
53 Sense Indicator 1 is set (to 1)
54 Sense Indicator 2 is set (to 1)
55 Sense Indicator 3 is set (to 1)
56 Skip (no operation)
57 If Trace Indicator is set to 1, reset Trace Indicator and Trace Stall to 0 and jump
58 Operator Interrupt is inhibited

*RESETS the inhibit automatically generated when the interrupt occurred.

UNIVAC 1050 PRICES

DESCRIPTION	RENTAL	SELLING PRICE
Central Processor Model 111, 4,096 characters 4.5 microseconds per character with 3 1/0 channels	$1,185	$ 47,500
Central Processor Model 1V, 8,192 characters, 2 microseconds per two characters, no 1/0 channels	2,385	95,500
Modules 4,096 characters for Model 111	325	13,000
Modules 8,192 characters for Model 1V	685	27,400
Freestanding Console	75	3,000
Integrated Console	45	1,800
(either Console is required with a Processor)		
Channels 3 thru 7 - Model 111 Processor	45	1,800
Channels 0 thru 7 - Model 1V Processor	85	4,600
(Two channels required for each magnetic tape synchronizer)		
Advanced Logic Model 111 Processor (multiply and divide)	150	6,000
Advanced Logic Model 1V Processor " " "	275	11,500
Card Reader - 800/900 CPM - 80 Columns	380	15,200
Card Reader - 800/900 CPM - 90 Columns	380	15,200
Card Reader - 600 CPM - 80 Columns	225	9,000
Card Reader - 600 CPM - 90 Columns	225	9,000
Card Punch - 300 CPM - 80 Columns	665	26,600
Card Punch - 300 CPM - 90 Columns	665	26,600
Card Punch - 200 CPM - 80 Columns	400	18,200
Card Punch - 200 CPM - 90 Columns	400	18,200
Printer 700 - 922 Lines per Minute	800	38,400
Printer 600 - 750 Lines per Minute	575	24,300
Print Buffer (Required with 700-922 LPM Printer and with 800/900 CPM Card Reader and on all Model 1V Processor systems as well as on all T, M, R and S systems)	185	7,400
Second Printer Synchronizer and Buffer	550	22,000
U-1VC Servos	700	38,400
U-1VC Servos - 800 PPI	750	40,800
U-1VC Synchronizer	995	39,800
U-1VC Power Supply	215	8,600
U-111A Servos	750	36,500
U-111A Synchronizer	995	39,800
U-111A Power Supply	215	8,600
U-V1C Control and first Servo	500	20,000
U-V1C Next three Servos, each	300	12,000
U-V1C Synchronizer	600	24,000
(Maximum of 4 Servos to each Control, 4 Controls per Synchronizer, two Control units give read/write/compute capability)		

UNIVAC 1050 Prices (continued)

DESCRIPTION	RENTAL	SELLING PRICE
FASTRAND Drum	$3,300	$160,000
FASTRAND Synchronizer	995	39,800
"B" Power Supply (required with two or more Subsystems beyond punched card I/O)	150	6,000
Paper Tape Control Unit	235	9,400
Paper Tape Reader, 1000 CPS	385	15,400
Paper Tape Reader, 400 CPS	200	8,000
Paper Tape Punch, 110 CPS	165	6,600
Reader Spooled Option	85	3,400
Punch Take-up Reel Option	5	200
1004 Adapter	200	8,000
Communication Subsystem		
32 Position Multiplexer	1,000	45,000
CLT - 80L	30	1,350
CLT - 81L	25	1,125
CLT - 80M	35	1,575
CLT - 81M	25	1,125
CLT - Parallel Out	35	1,575
CLT - Parallel In	35	1,575

(Reference 490 Price List)

UNIVAC

DIVISION OF SPERRY RAND CORPORATION