UNIVAC

DATA PROCESSING DIVISION

1050

SYSTEMS

CENTRAL PROCESSOR

REFERENCE MANUAL

UP-3912 Rev. 1

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, are required by the development of its respective Systems.

CONTENTS

TABLES AND FIGURES

TABLES

FIGURES

1. INTRODUCTION

1.1. SCOPE

The primary purpose of this manual is to provide the basic knowledge necessary for programming the UNIVAC 1050 Central Processor, and serve as a reference for the programmer. Background information is provided on the internal operation of the Central Processor and the different types of information representation, as well as information on data and instruction formats, specialized areas of storage (registers, I/O control tetrads, etc.), coding, the instruction repertoire, and automatic program interrupt.

A second purpose of this manual is to describe the Central Processor Console and its operation, and serve as a reference for the operator. A detailed description of all the console controls and indicators is provided along with a description of their use to communicate with the program and control various normal and abnormal conditions.

1.2. GENERAL DESCRIPTION

The Central Processor is the control center of the UNIVAC 1050 System. It contains the circuitry for logic and arithmetic operations, the core storage and the power supply.

The Central Processor performs three main functions:

- Control
- Storage
- Arithmetic Computation

1.2.1. Control Function

The control circuitry of the Central Processor accesses and executes instructions from storage. It also maintains control over the operation of all peripheral devices. External control is facilitated by the lights and buttons on the Central Processor Console.

1.2.2. Arithmetic Function

The arithmetic function instructions employ the arithmetic registers to perform binary and decimal addition and subtraction, as well as decimal multiplication and division. Overflow is indicated and decimal sign control is provided.

1.2.3. Storage Function

The storage function of the UNIVAC 1050 Central Processor is provided by one to eight modules of core storage, each containing 4096 characters.

1.3. INFORMATION REPRESENTATION*

1.3.1. Type of Notation

Digital computers employ a system of notation called the *binary* system. Unlike the decimal system which uses ten symbols (0 through 9) and is based on a radix (root) of 10, the binary system employs only two symbols (0 and 1) and is based on a radix of 2.

---

* *The reader familiar with numbering systems may wish to skip to Section 1.4, DATA AND INSTRUCTION FORMATS.*

The two symbols of the binary system represent the two possible states of an information conveying electronic device. The 1 symbol indicates a registered pulse while the 0 symbol indicates a no pulse registration. Information is represented in the computer by pulse-no-pulse combinations with a specific pattern for each alphabetic, numeric, and special character.

1.3.1.1. Decimal and Binary

Numbering systems are based on positional notation. That is, each digit in a quantity is weighted with a specific value. The value of a digit is determined by its position within the quantity and the radix of the numbering system. For example, using decimal notation, the number *seven thousand four hundred sixty nine* would be represented as 7469 which is equivalent to

$$(7x10^3) + (4x10^2) + (6x10^1) + (9x10^0) = 7,000 + 400 + 60 + 9$$

Note that each digit, from right to left, is considered to be multiplied by a successively higher power of 10.

The binary system is also based on a system of positional notation, but, as was stated previously, it uses a radix of 2 and employs only two symbols to represent quantities. For example, the number nine expressed in *pure binary* would be

1001

which is equivalent to

$$(1x2^3) + (0x2^2) + (0x2^1) + (1x2^0) = 8 + 0 + 0 + 1$$

Note that each binary digit (bit), from right to left, is multiplied by successively higher powers of 2.

1.3.1.2. Fixed Length Notation

Instead of specifying information with a variable series of binary digits (the length of the series dependent upon the quantity to be specified) representing successively higher powers of 2, a system of notation is used that specifies information by smaller, fixed length groupings of binary digits. Each grouping, fixed in format as well as length, is used to represent a digit, an alphabetic character, or a special symbol. Assuming a system of notation that employs a fixed length format, a single digit would be represented by a single group of bits, a two digit polynomial by two bit groupings, a three digit polynomial by three bit-groupings, and so forth. For example, in *pure binary* the number 27 would be

11011

which is equivalent to

$$(1x2^4) + (1x2^3) + (0x2^2) + (1x2^1) + (1x2^0) = 16 + 8 + 0 + 2 + 1$$

However, by employing a fixed format of 4 bit notation, known as binary coded decimal, the number 27 would be represented as

0010	0111
2	7

and similarly, the number 369 as

0011	0110	1001
3	6	9

Note that within each 4 bit grouping, the bit positions are weighted with a value of 8, 4, 2, and 1 or 2^3, 2^2, 2^1, and 2^0. The decimal digits 0 through 9 then are represented in the following manner:

DECIMAL	BINARY 4 BIT NOTATION
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

With 4 bit positional notation, only 16 unique permutations can be created. This is obviously insufficient to specify all numeric, alphabetic, and special characters generally employed in a computing system. By adding two more bit positions and using them as a qualifying factor to a 4 bit combination, a total of 64 unique permutations can be represented. The four bit positions on the right are called the *numeric portion.* Two additional positions on the left, which represent no actual numeric quantity, are called the *zone portion.*

Qualification of a numeric quantity is unnecessary, therefore, the zone portion is always 00. When representing alphabetic characters or special symbols, however, a 1 bit is entered in either or both zone positions. The letters A through I, therefore, may be represented with the numeric portion specifying a value from 1 to 9 (0001 to 1001) and the zone portion containing a 01 qualifier; the letters J through R with the same numeric specifications but with a zone qualification of 10; and finally, the letters S through Z with a numeric specification of from 2 to 9 and a zone qualification of 11. So, for example, the letters A, J, and S would be represented as

ZONE	NUMERIC	CHARACTER
01	0001	A
10	0001	J
11	0010	S

This is not the same as UNIVAC 1050 code however.

The zone and numeric specifications for special symbols such as the comma, apostrophe, asterisk, and so forth are dependent upon computer design. That is, computers are wired to accept a unique bit combination for a particular special symbol. Since there is no natural sequence relationship between special symbols, as with numerics or alphabetics, the bit configuration for special symbols must be arbitrary. The sequence for UNIVAC 1050 special symbols is shown in Table 1-1.

1.3.1.3. Excess Three (XS 3)

Excess three (XS 3) is a method of notation that is used by the UNIVAC 1050 System. It establishes some measure of compatibility with the data formats of the other UNIVAC Computing Systems. The zone position is specified in the standard manner previously described for fixed length binary coded decimal notation. The difference exists in the numeric portion where each binary specification is a value that is three greater than its decimal equivalent. For example, the number 8 is represented in XS 3 as

ZONE	NUMERIC
00	1011

Note that the numeric portion, weighted with positional values of 8, 4, 2, and 1 from left to right, is actually equal to 11. Similarly, the number 6 is represented as

ZONE	NUMERIC
00	1001

Here the numeric portion is specified as 9 or three greater than the decimal digit it represents.

There are several reasons for utilizing this method of notation in certain UNIVAC Systems; some of these reasons are

- *It allows three quantities to test less than 0.*
- *It facilitates complementation.*
- *It permits the carry to occur as in decimal notation.*

An involved discussion of these and other reasons for the utilization of XS 3 notation is beyond the scope of this manual. It is sufficient that the programmer is aware of the basic format and that this provides in the UNIVAC 1050 Computer a factor of data compatibility with other UNIVAC Systems. Table 1-1 gives a listing of the XS 3 code configurations for all the alphabetic, numeric, and special characters utilized in the UNIVAC 1050 System.

1.3.1.4. Parity

A parity check is used by the computer to ensure that accurate transmission of data occurs. The parity position is an extra bit position added to ensure that there will always be an odd number of 1 bits in any character representation. In this way, if a bit is either dropped or added in transmission, the odd parity check will indicate an improper registration. For example, the alphabetic S contains an even number of 1 bits:

ZONE	NUMERIC
11	0101

To pass the odd parity check, a 1 bit is added to the parity position, thereby creating an odd number of 1 bits in the representation:

PARITY	ZONE	NUMERIC
1	11	0101

If the number of 1 bits in the configuration is already odd, the parity position will be 0.

1.3.1.5. Octal Numbers and Complements

Octal notation is used in source language and program testing diagnostic printouts. The octal or base 8, number system expresses values as multiples of powers of 8.

Octal notation is a fixed length system of binary notation. The binary number is interpreted octally by grouping the bits into bytes of three, starting from the right, and interpreting each byte into its octal equivalent. Within each byte the bit positions are weighted with the value of 4, 2, and 1, or 2^2, 2^1, and 2^0. If, after grouping the bits in the fashion described, the most significant byte contains less than three bits, as many binary zeros are implied to the left as are required to bring the number of bits in that group to three. For example, the binary number 10011101101 is interpreted octally as follows:

(0)10	011	101	101
2	3	5	5

An octal number such as the one derived from the binary number described is noted with the subscript 8 following it, e.g., 2355_8, to distinguish it from the decimal number 2355_{10}. In the PAL assembly language employed in programming the UNIVAC 1050 System, however, an octal number is noted by preceding it with a zero; thus, 02355 means 2355_8, while 2355 means 2355_{10}.

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 1-1. UNIVAC 1050 Character Set.

The binary number 10011101101 is the sum of

$$
\begin{aligned}
1 \times 2^{10} &= 1024 \\
0 \times 2^{9} &= 0 \\
0 \times 2^{8} &= 0 \\
1 \times 2^{7} &= 128 \\
1 \times 2^{6} &= 64 \\
1 \times 2^{5} &= 32 \\
0 \times 2^{4} &= 0 \\
1 \times 2^{3} &= 8 \\
1 \times 2^{2} &= 4 \\
0 \times 2^{1} &= 0 \\
1 \times 2^{0} &= 1 \\
&\overline{1261}
\end{aligned}
$$

Therefore, $2355_8 = 1261_{10}$.

Appendix A provides a two-way octal to decimal and decimal to octal conversion table. For the convenience of the programmer who wishes to do his own conversions, the following paragraphs present an octal to decimal and a decimal to octal conversion procedure.

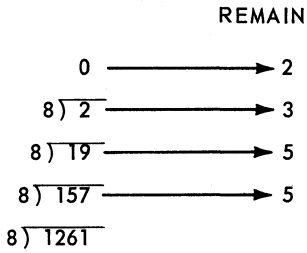
To convert an octal representation to its decimal equivalent, multiply the most significant digit by 8, and add the next most significant digit to the product. Multiply this sum by 8 and add the third most significant digit to the product. Repeat the multiplication and addition process until the least significant digit has been added, whereupon this final sum will be the decimal equivalent of the octal number.

The following example illustrates how this method converts 2355_8 into its decimal equivalent:

$$
\begin{aligned}
2 \times 8 &= 16 \\
+\ 3 & \\
\overline{19} \times 8 &= 152 \\
+\ 5 & \\
\overline{157} \times 8 &= 1256 \\
+\ 5 & \\
\overline{1261} & .
\end{aligned}
$$

To convert a decimal number into its octal equivalent, divide 8 into the number and record the remainder (0 through 7) as the last significant digit of the octal equivalent. Divide 8 into the quotient, and record the remainder as the next least significant digit. Repeat the division of the quotient recording the remainder until a quotient less than eight is realized, whereupon the final quotient is the most significant digit of the octal equivalent and the final remainder is the next most significant digit of the octal equivalent.

The following example illustrates how this method converts 1261_{10} into its octal equivalent:



No signs are involved in binary operations in the UNIVAC 1050 System; however, negative binary values – or, effectively, their equivalent – can be developed and represented within the computer. These negative binary values are represented as the two's complement of the binary representation of the absolute value of the numbers. The two's complement is formed by adding 1 to the one's complement of the value, ignoring any carry beyond the most significant bit position; and the one's complement, in turn, is formed by converting every 1 bit in the binary representation to 0, and converting every 0 bit to 1.

For example, the binary representation of $+1261_{10}$ is

010011101101

the one's complement of this binary number is

101100010010

and the two's complement of the number is

$$\begin{array}{r} 101100010010 \\ + \qquad\qquad 1 \\ \hline 101100010011 \end{array} = 5423_8$$

Whenever the binary integer 101100010011 is employed as an operand in a binary add or subtract operation, the effective value of this operand is -1261_{10}.

1.4. DATA AND INSTRUCTION FORMATS

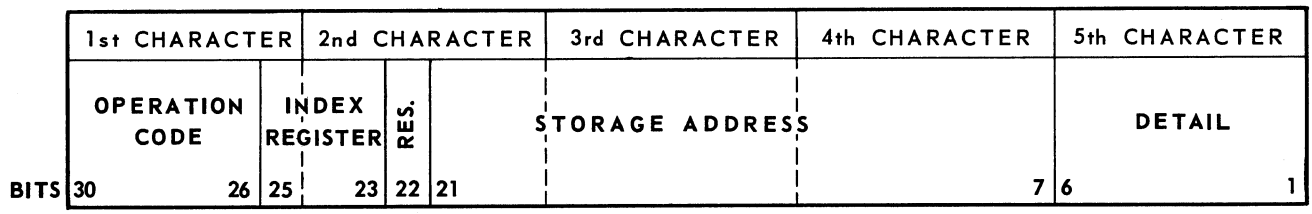
1.4.1. General Description

Instructions are contained in storage. They are always five characters in length whereas data fields may be any number of characters in length. Instructions are executed in sequence except where a programmed instruction initiates a break in the sequence.

The arithmetic unit of the Central Processor performs the calculations and data manipulation called for by the instructions. It contains an adder for decimal and binary arithmetic operations, and additional circuitry which provides a wide range of data handling abilities.

The control unit of the Central Processor selects, interprets, and initiates the execution of instructions in the stored programs which govern the operation of the system.

1.4.2. General Instruction Format



BIT POSITIONS	NAME	
30 – 26	OPERATION CODE	The operation code specifies the function which the Central Processor is to execute.
25 – 23	INDEX REGISTER	The index register modifies the address specified in the instruction.
22	RESERVED	This bit is reserved.
21 – 7	STORAGE ADDRESS	This is the (M) portion of the instruction. It specifies the store address of the operand. If an operand is greater than one character in length, (M) refers to the least significant character of the operand (rightmost). There are two exceptions: Zero Suppress and Block Transfer instructions in which (M) specifies the most significant character of the operand.
6 – 1	DETAIL FIELD	Depending on the instruction, the detail field may specify operand length, tetrad number, a comparison indicator, an arithmetic register, or number of bits.

1.5. STORAGE

1.5.1. General

The basic unit of storage in the UNIVAC 1050 store is the character which consists of six information bits and one parity bit. The parity bit is of no concern to the programmer. It is used only by the circuitry and is not accessible to him.

The UNIVAC 1050 Central Processor may have from 1 to 8 sections of storage, each section comprising 4096 character positions or locations. Each position has its own address and each position is directly addressable.

Each section of main store is divided into rows. There are 64 rows in each section. A row consists of 64 consecutive characters. The address of the most significant character (leftmost) is either zero or some integral multiple of 64.

Program instructions and data are contained in storage. Each instruction occupies five consecutive locations. Data fields are variable in length. The sign, if any, of a data field is in the most significant bit of the least significant character.

The first six rows of storage, portions of which perform unique functions, are illustrated in Figure 1-1.

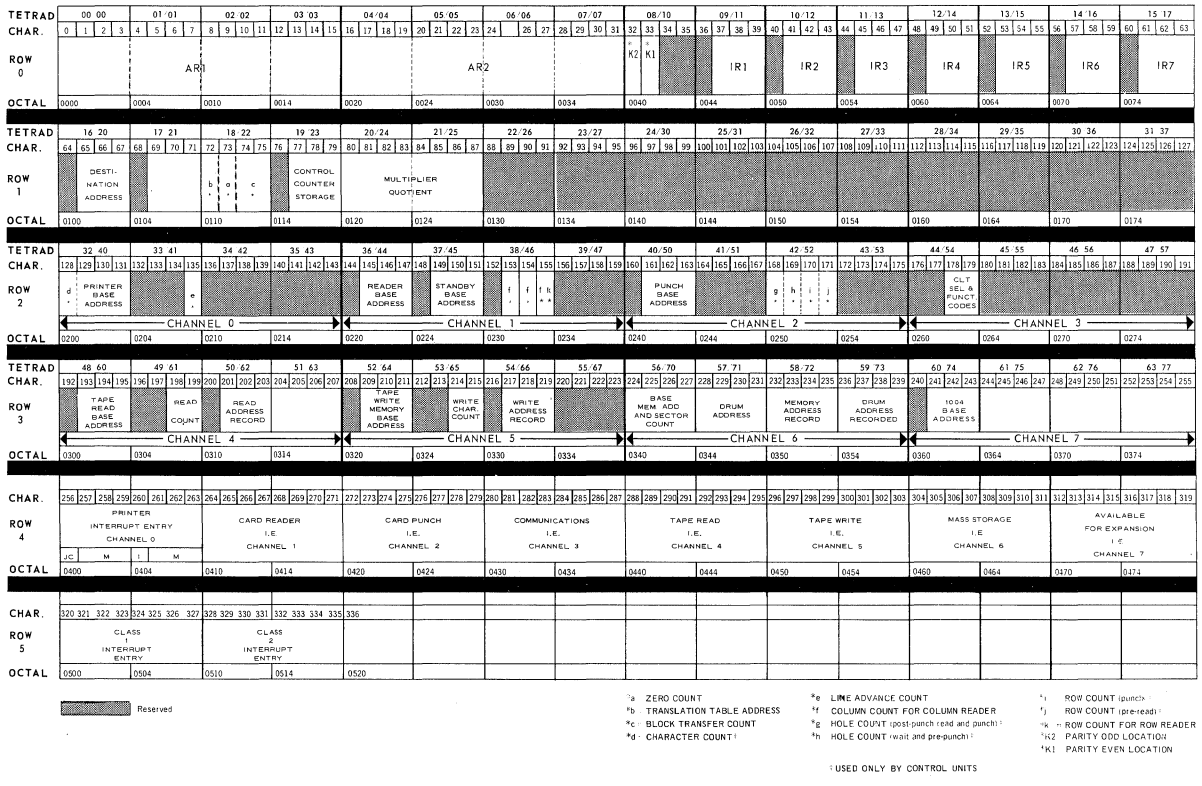


Figure 1-1. Layout of First Six Rows of Store.

1.5.2. Tetrads

The first four rows of storage in the UNIVAC 1050 System are subdivided into 64 fields of four characters each and are called tetrads. Tetrads are addressable either by tetrad number or the actual storage location. The method of addressing tetrads is dependent upon the instruction being used. Certain tetrads are designed for specific functions. A description of what these tetrads do is given in the following table.

TETRADS	LOCATIONS	PURPOSE
0–3	0–15	*Arithmetic Register 1 (AR1)
4–7	16–31	*Arithmetic Register 2 (AR2)
8	32, 33	Character bit sum storage.
9–15	36–63	*Index Registers 1–7
16	64–67	Destination address for Block Transfer
17	68–71	Origin address for Block Transfer
18	72	Address of table for translation
18	73	Count of zeros suppressed after Zero Suppress.
18	74, 75	Controls number of characters in Block Transfer
19	77–79	Control Counter Storage
20–21	80–87	Multiplier – Quotient
22–31	88–127	Unassigned
32–35	128–143	Printer I/O, Channel 0
36–39	144–159	Reader I/O, Channel 1
40–43	160–175	Punch I/O, Channel 2
44–47	176–191	Communications I/O, Channel 3
48–51	192–207	Tape Read, Channel 4
52–55	208–223	Tape Write, Channel 5
56–59	224–239	FASTRAND I/O, Channel 6
60–63	240–255	Channel 7, available for expansion

* *The arithmetic registers and index registers can be addressed in three different ways: as arithmetic or index registers, as tetrads, and as store locations.*

Table 1-2. Tetrad Location Chart

1.5.2.1. Arithmetic Registers

Tetrads 0 through 7 function as arithmetic registers. Arithmetic Register 1 (AR 1) comprises tetrads 0 through 3 (store locations 16 through 31). The arithmetic registers are addressed either by AR 1 or AR 2, tetrad number, or actual storage location.

1.5.2.2. Index Registers

Tetrads 9 through 15 function as index registers 1 through 7. Since only the 15 least significant bits (contained in the three least significant characters) of each index register are used in an indexed operation, the most significant character of each index register tetrad is available to be used for other purposes.

There are no signs in the index registers. The value in the index register is treated as an absolute binary value in an indexed operation. Negative indexing may be accomplished by placing the two's complement of the decrement number in the index register.

The index registers may be addressed by index register number, tetrad number, or actual storage location number.

1.5.2.3. Input/Output Control Tetrads

A fixed storage area consisting of four consecutive tetrads is associated with each input/output channel. Information placed in this area controls the operation of the peripheral device. The input/output control tetrads, that are located in storage rows two and three, are shown in Figure 1-1.

1.5.3. Fixed Interrupt Locations

Store locations 256 through 335 are fixed locations associated with the interrupt circuitry of the system. These eighty locations are divided into ten groups of eight consecutive characters each, which are known as interrupt entries. These interrupt entries are assigned as follows:

OCTAL	DECIMAL	INTERRUPT ENTRY ASSIGNMENTS
0400 – 0407	256 – 263	Channel 0: Printer
0410 – 0417	264 – 271	Channel 1: Reader
0420 – 0427	272 – 279	Channel 2: Card Punch Unit
0430 – 0437	280 – 287	Channel 3: Communications
0440 – 0447	288 – 295	Channel 4: Magnetic Tape Read
0450 – 0457	296 – 303	Channel 5: Magnetic Tape Write
0460 – 0467	304 – 311	Channel 6: Mass Storage
0470 – 0477	312 – 319	Channel 7: Unassigned
0500 – 0507	320 – 327	Class I Interrupt Entry
0510 – 0517	328 – 335	Class II Interrupt Entry

The format of these interrupt entries, and their functions, are discussed fully in the section on Automatic Program Interrupt (Section 4).

1.5.4. Addressing

Instructions and data in storage are accessed by other instructions through the 15 bit memory address designated the M portion of the instruction.

Whenever an instruction references a multicharacter field, the M portion usually designates the address of the rightmost or least significant character. (The exceptions will be explained in the description of the instructions involved.)

2. CODING IN ASSEMBLY LANGUAGE

2.1. CODING FORM

Most programs for a UNIVAC 1050 System with 8192 character storage or larger are written in the language of the PAL Assembly System. Programs for a system with 4096 character storage are written in PAL Jr. See Section 2.4. The PAL assembler is a UNIVAC 1050 program which accepts mnemonic and symbolic input, a form meaningful to the programmer, and generates instructions in absolute binary form, the only form meaningful to the computer. Any action based on attempts to employ instruction forms not described in this reference manual deviates from UNIVAC recommendations and must be the user's responsibility.

Figure 2-1 shows the symbolic coding form for the UNIVAC 1050 Pal Assembly System.

In the description of this form, which follows, certain terms are used with specific definitions:

- *Alphabetic* character means a character of the English alphabet set (A through Z).
- *Numeric* character means a character of the Arabic numeral set (0 through 9).
- *Alphanumeric* character means an alphabetic character, a numeric character, or a special symbol.

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 2-2; the 90 column source card, in Figure 2-3 In 90 column systems, columns 81 through 90 are also available to the Program-ID field, but their contents will not be printed on the output listing; their use, therefore, is not recommended.

2.1.1. Program-ID

The program name is written in this field. It is composed of from one to six alphanumeric characters, and is written starting at column 75. An example of an entry in this field is

PROGRAM-ID
75 80
P A Y 0 1

PAGE 1 3

UNIVAC
DIVISION OF SPERRY RAND CORPORATION

UNIVAC® 1050 PAL ASSEMBLER CODING FORM

PROGRAM-ID 75 80

PROGRAM _____ PROGRAMMER _____ DATE _____ PAGE ____ OF ____ PAGES

For BEGIN only.

SEQUENCE			LABEL	OPERATION	OPERANDS	COMMENTS →	PROGRAM-ID →
PAGE 1 3	LINE 4 5	INS 6	7 11	13 18	19 30 40 45	46 50 60 70	75 80 90

UD1-802 13/12 56 125M 7/65

(80-90 COLUMN FORM)

Figure 2–1. PAL Assembler Coding Form

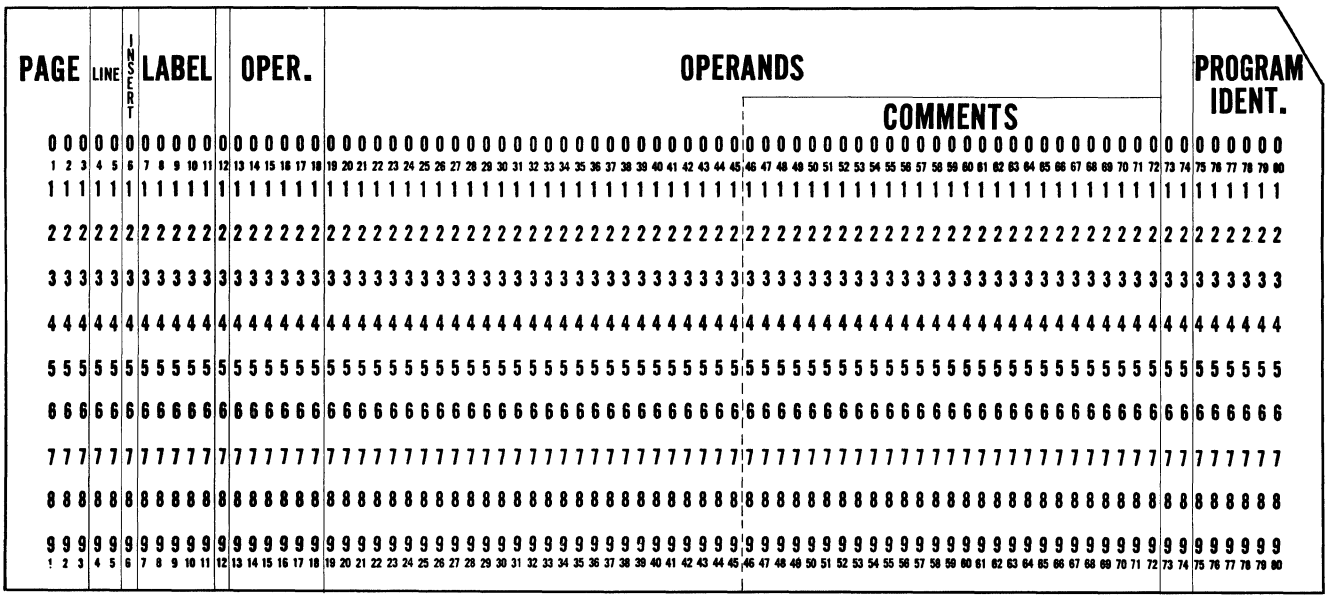


Figure 2-2. PAL 80-Column Source Card.

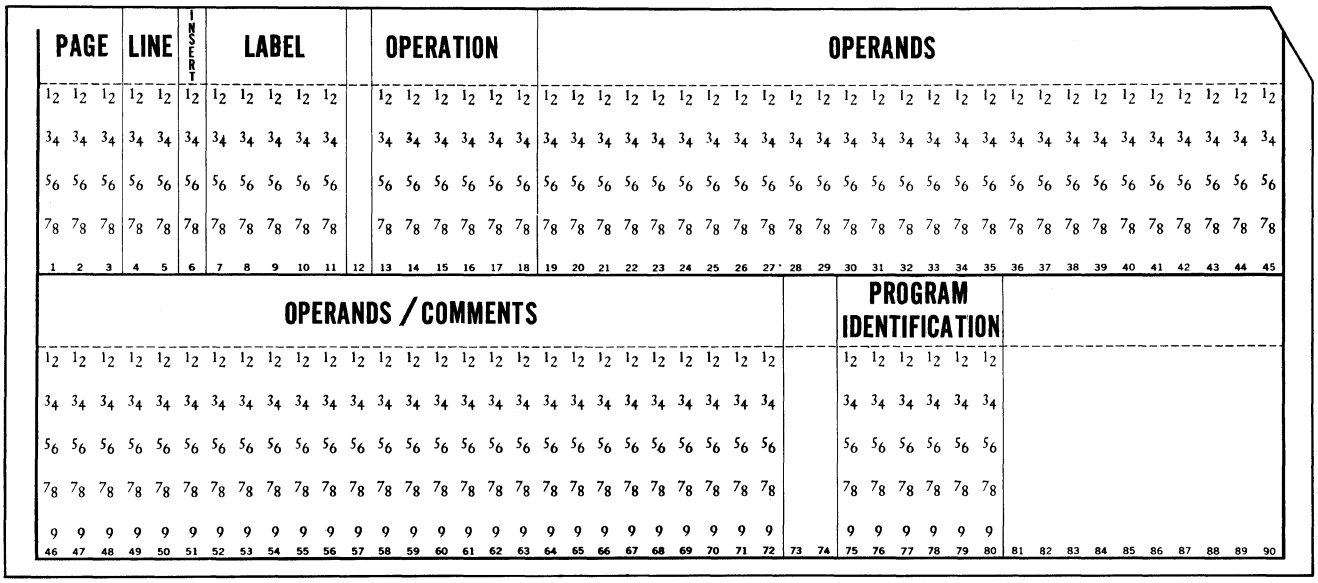


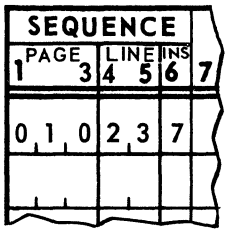
Figure 2-3. PAL 90-Column Source Card.

2.1.2. Sequence

This field is a six character numeric field composed of a three digit page field, a two digit line field, and a one digit insert field. There may be a page field entry and a line field entry on each card.

While processing input lines, the assembler performs a sequence check on this field to make sure that page and line entries are in ascending sequence. An out of sequence line is flagged on the output listing as an "S" error.

The insert field is provided to permit the insertion of additional coding lines when correcting a source program. The insert field entry consists of one numeric digit. 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



There is one restriction on the digit used for INS. If more than one instruction is to follow a particular page and line, each insertion line must have a sequentially higher INS number than any preceding it.

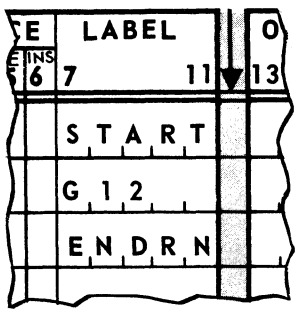
If inserts are made, the cards punched from the insert lines must be physically placed in their proper places in the source deck, prior to assembly.

2.1.3. Label

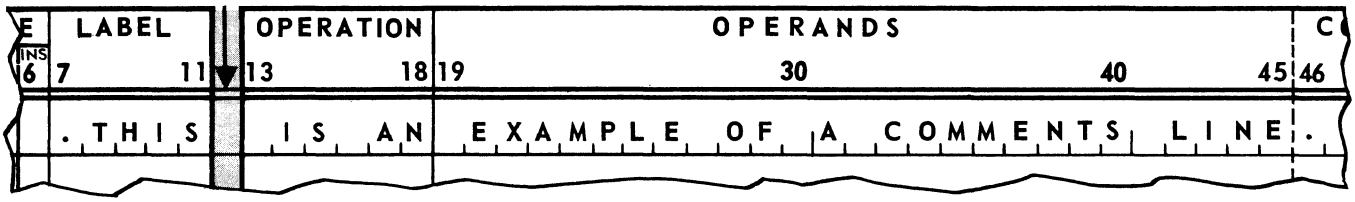
A label is an alphanumeric symbol associated with the line on which it appears. It consists of five characters or less, the first character of which must be an alphabetic character other than the letter X. A label must begin in column 7,and is terminated either by column 12 or by the first blank appearing in the field. The entire field may be blank. (Column 12 can be used only by a six character label, if any, of the assembler directive BEGIN or by a comments line. Otherwise it is always left blank.)

The label of an instruction line names the leftmost character of the instruction, while the label of a data field or a constant names the rightmost character of the field or constant.

Some examples of labels are



If column 7 contains a period, the entire line is a comment. It produces no coding, but the line is printed on the output listing.

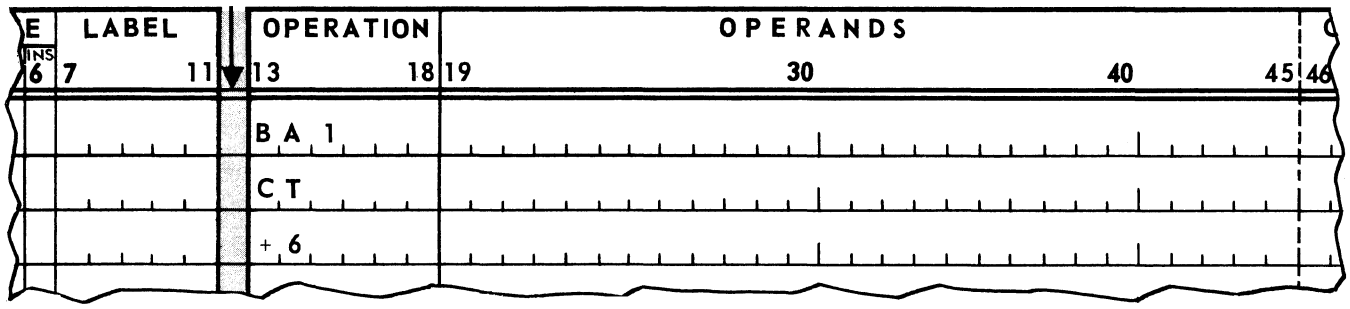


2.1.4. Operation

The operation field is a six character field beginning in column 13. This field may not be blank. The field usually contains a mnemonic operation code, which the assembler converts into a five bit operation code. The operation field may also contain an assembler directive or a data generating code.

An entry in this field must begin in column 13 and is terminated by the first blank appearing in the field or by the end of the field.

The following are examples of operation field entries:



2.1.5. Operands

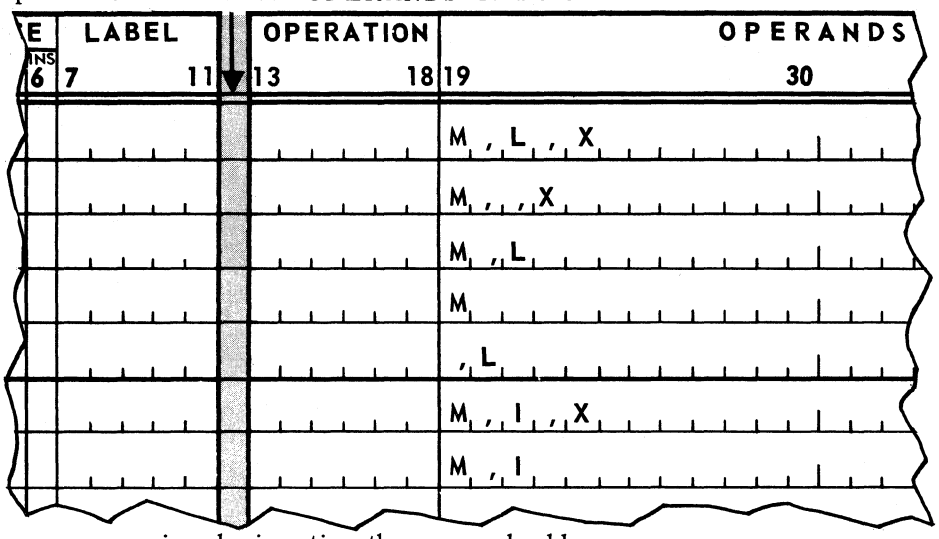
The operands field usually contains symbolic or absolute descriptions of the Index Register, Storage Address, and Detail portions of an instruction. These descriptions are called expressions.

Each expression except the last one on a line must be terminated by a comma immediately following the last character of the expression. The last expression on a line is terminated by a blank. The first blank following a character which is neither a blank nor a comma indicates that no more expressions follow. Column 72 also terminates the operands field.

The assembler processes the operands field from left to right, a character at a time. Whenever a comma is encountered, the assembler recognizes the end of an expression and expects at least one other expression to follow; but whenever a blank appears following a non blank character which is not a comma, the assembler expects no more expressions to follow on the same line. Two successive commas within a string of expressions indicate a blank expression. An expression may have any number of preceding blanks.

The maximum number of expressions that may be written on one line and the interpretation of each expression is determined by the contents of the operation field. Any line may have less than the maximum number of expressions. For example, a symbol written as the M expression on an instruction line might 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



M is an expression designating the operand address.

L is a decimal or octal number or defined label specifying the operand length in terms of characters.

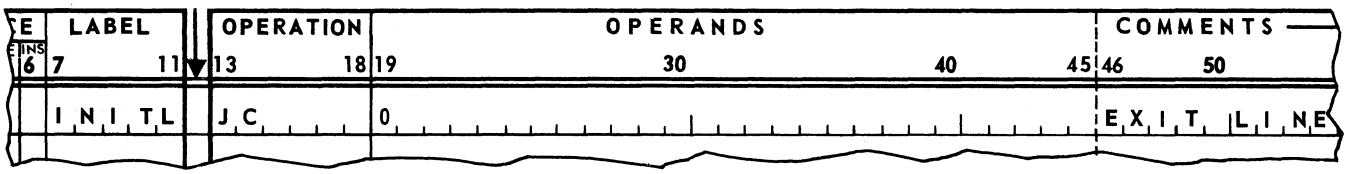
X is an expression naming an index register.

I is an expression identifying an indicator.

Note that if the last expression which might appear on a line is omitted, the comma which would have preceded it is omitted. Also, if the M expression is to be specified as zero, it may not be represented by a comma, but must be written as zero (0).

2.1.6. Comments

Significant comments may be written anywhere on the line beyond the blank which terminates the last expression. It is recommended, however, that comments be indented at column 46, for the sake of obtaining an output listing which is easier to read. For example,

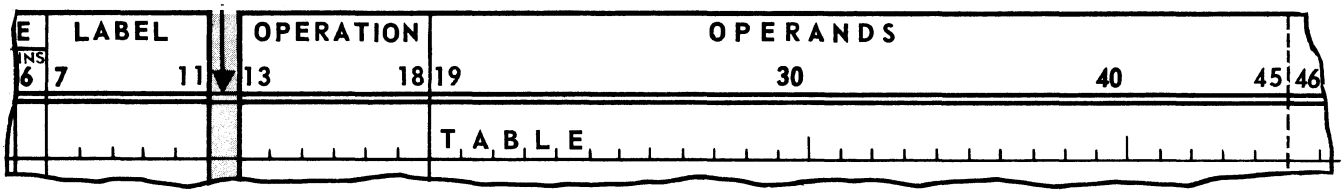


2.2. SYMBOLS AND CONVENTIONS

There are three general types of expressions:

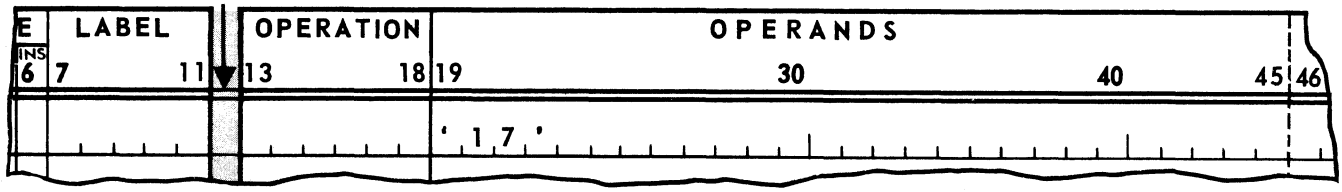
- *Symbolic* value is assigned by the assembler
- *Constant* value is assigned by programmer
- *Combined* value may be wholly or partially assigned by either the assembler or programmer

A symbolic expression is one whose first character is an alphabetic character and is not preceded by an apostrophe. An example of a symbolic expression is



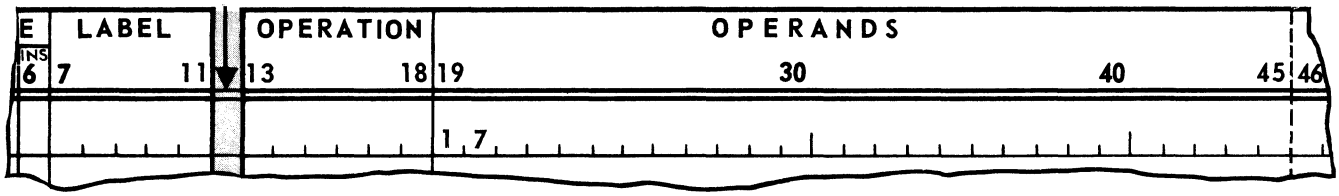
A constant expression is one whose first character is either an apostrophe or a number. A constant expression may be alphanumeric, decimal, or octal.

An alphanumeric constant is represented by enclosing it in apostrophes. From the expression, the assembler generates the UNIVAC 1050 six bit code for every character appearing within the apostrophes. For example, the expression



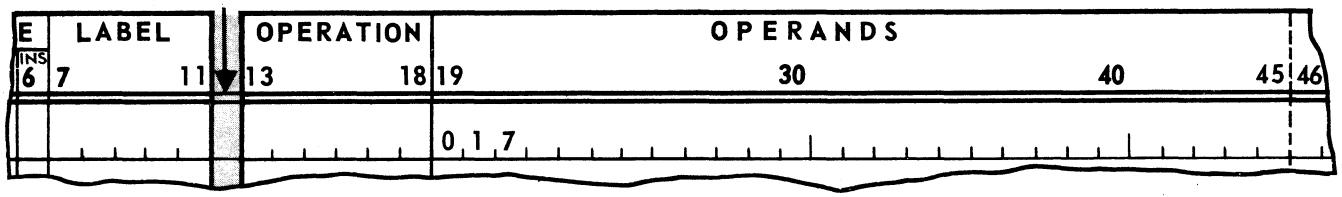
produces the bit configurations 00 0100 and 00 1010, which are the UNIVAC 1050 six bit codes for the characters 1 and 7, respectively.

A constant is decimal if its first character is a number other than zero. The assembler generates the binary equivalent of the decimal number. For example, the expression



produces the bit configuration 010001, which is the number seventeen expressed in binary.

If the first character of a constant expression is zero, the number is taken to be an octal number and is converted from octal to binary. For example, the expression



is converted into 001111.

A special constant expression is the dollar sign ($), which means the current value of the location counter. Its value is one greater than the address of the last location which the assembler has assigned.

The following chart summarizes the interpretation given to each type expression.

TYPE OF EXPRESSION	ABBREVIATION	FORM	VALUE	EXAMPLE
Symbol	S	one to five alphanumeric characters beginning with an alphabetic character other than the letter X.	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 most significant character of the line in which the item $ appears.	$ + 15
Octal	0	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
Alpha-numeric	A	any characters (excluding apostrophes) enclosed in apostrophes (').	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

A combined expression is one that has two or three symbolic or constant expressions connected by a plus (+) or a minus (–) sign.

An expression may have a leading plus or minus sign to denote a positive or a negative quantity. If an expression does not have a sign, it is assumed to be positive.

Since all expressions are converted into binary, a negative expression is converted into the two's complement of the value.

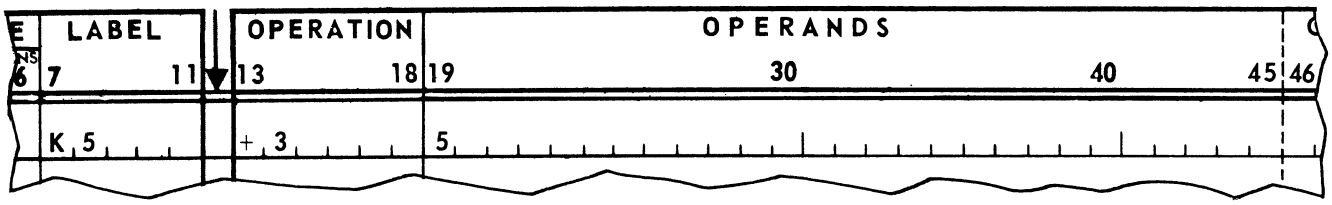
2.3. DATA GENERATION

The PAL assembly system provides means of generating data other than instructions from a coding line.

A constant of up to 16 characters is generated by writing +n or −n in the operation field of a line. The n is a decimal number ranging from 1 through 16 specifying the number of characters in the constant. An alphanumeric constant can range in length from 1 to 16 characters. This constant must be written within apostrophes. A decimal constant can range in length from 1 to 7 characters. An octal constant which can occupy from 1 to 8 characters is written with 1 to 16 digits plus a preceding zero.

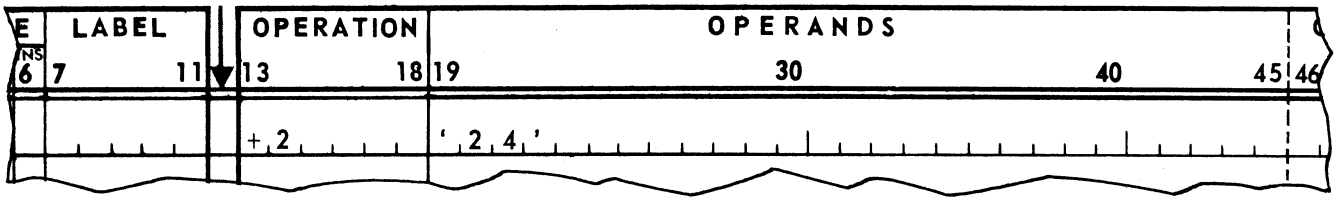
The label of such a line names the least significant character generated from the entry in the operands field of that line.

The operands field must contain a single expression, which may be alphanumeric, decimal, octal, or a label. If the value of the expression is an integer of less than n characters, the assembler generates as many binary zeros to the left of the integer as are needed to fill out the rest of the field. For example, from the line

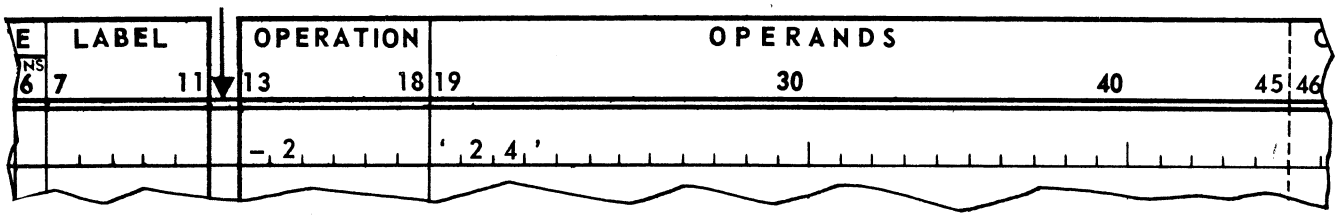


the assembler generates 000000 000000 000101. K5 names the least significant character.

If the operands field expression is alphanumeric and the sign in the operation field is negative, the sign bit of the constant is reversed. For example, from the line

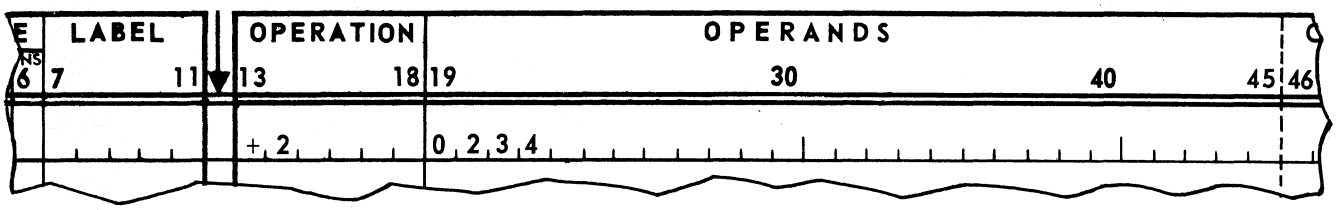


the assembler generates 000101 000111, while from the line

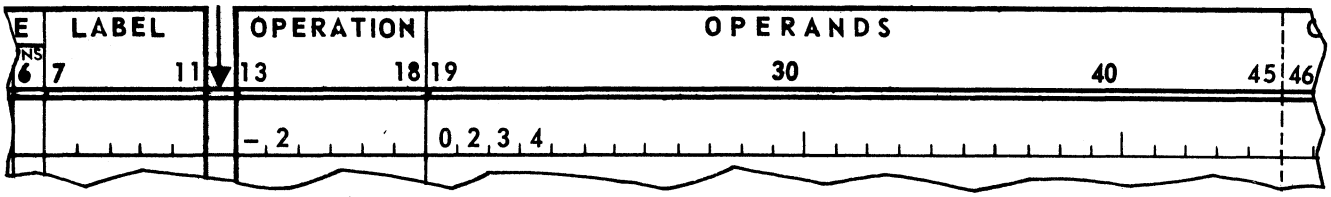


the assembler generates 000101 100111.

When the operands field expression is decimal or octal, and the sign in the operation field is negative, the two's complement of the expression value is generated. For example,



produces 000010 011100, while



produces 111101 100100.

When the expression in the operands field is a label, unmodified, or with a constant modifier, and the operation field contains:

+1 – the length (in number of characters) of the field named by that label is supplied.

+3 – the 15 bit address which the assembler assigns to the label will be supplied, preceded by three binary zeros.

+4 – or higher – the 15 bit address assigned to the label occupies the 15 least significant bit positions of the n character field. The rest of the field contains binary zeros.

2.4. PAL JR ASSEMBLY SYSTEM

The PAL JR card assembler is used with a Central Processor that has a storage capacity of 4096 characters. The features of PAL JR are the same as those of the PAL assembler with certain limitations:

- Label size is limited to three characters.
- There are no implied field lengths. Field lengths and index registers must be specified in the instructions.
- The EQU directive may not be employed to specify the field length or the index register. AREA directives may not be employed to specify index registers or fill characters and cannot define subfields.
- The second expression in the operands field of the Tetrad instructions must be a Tetrad number.
- The I/O areas have fixed labels and index registers and cannot exceed two backup areas for each unit.
- The Comparison Jumps (JG, JE, JU, JS) are eliminated in this system. The Jump Conditional (JC) instruction is employed to perform their function.
- The maximum value of a decimal or octal constant that can be described by the EQU directive is 4095 (07777).

3. INSTRUCTION REPERTOIRE

The instruction repertoire of the UNIVAC 1050 System is arranged in the following pages by functional category. Each category is introduced by a brief description of the general coding rules for the instructions in that category.

Each instruction is described in the following manner:

OPERATION

Format: **PAL Mnemonic** Required Expressions

Function: (Concise description of what the instruction accomplishes)

Notes:

(Programming considerations and further description of the instruction)

Example(s):

(Programming examples and description of the operands in verbal and graphic form, showing the operands before and after the execution of the instruction, if necessary)

In describing the operation of the various instructions, the abbreviation M_x specifies the *effective* character or field position in main store. By effective character is meant M as modified by the contents of index register X (if it is called for).

Any expression of the instruction other than M and X is the detail field. The detail field may have subfields, some of which are extensions of the operation code. This accounts for the fact that the octal operation codes for two or more instructions may be identical.

Preferably, the more commonly used special purpose tetrads should be addressed by means of a label rather than a tetrad number. The ability to do so is provided by the EQU directive, which is fully discussed later in this section. Table 3-1 presents a list of the labels used in the coding examples.

In the Univac 1050 System there are 64 indicators addressed as decimal numbers 0 through 63 (octal 0 through 077). These indicators fall within three functional groups; indicators that are testable, indicators that cause an unconditional jump and specific function to be performed, and indicators that cause a certain function to be performed but do not break the sequence of instructions. The function performed depends upon the indicator involved.

Among the testable indicators are those which test the settings of the three Sense Switches and the three Sense Indicators. Sense *Indicators* are internal devices which are set and reset under program control. Sense *Switches* are on the console and are set and reset manually. Unlike comparison indicators which are set and reset as a result of a comparison, the Sense Indicators may be set and reset arbitrarily to provide programmable switches.

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 STORE				
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	Indicates overflow occured in last binary subtract or didn't occur in last binary add
KDF	EQU	050	40	Decimal overflow indicator

Table 3-1. Suggested Standard Equality Statements.

The use of these indicators is discussed in detail with the instructions involved. Normally the indicators will be addressed using a label which is equated to the indicator number. The EQU operation is defined in the Card Assembly System Manual. Table 3-1 lists the more commonly used indicators and their suggested labels.

Tables 3–2 and 3–3, respectively, summarize the instruction repertoire and the mnemonic operation codes of the UNIVAC 1050 System. The instruction execution times appear in Table 3–4 on page 3–73.

1st CHARACTER	2nd CHARACTER		3rd CHARACTER	4th CHARACTER	5th CHARACTER
OPERATION CODE	INDEX REGISTER	RES.	MAIN STORE	ADDRESS	DETAIL
30 26	25 24 23	22	21 19 18 13	12 7	6 1

INSTRUCTION WORD FORMAT

WRITTEN FORM					IND. NO.	INTERNAL FORM (OCTAL)										
						30-26	25-23	22	21 – 7	6	5	4	3	2	1	
TYPE	MNEM CODE	OPERANDS	INSTRUCTION	DESCRIPTION	Dec.	OP CODE	X		M	DETAIL						
DATA TRANSFER	BDa	M, L, X	BRING DECIMAL	$(M)_L \rightarrow$ AR, DECIMAL + &		56	0 – 7		0 – 77777	1	0/1*	0 — L — 17**				
	BAa	M, L, X	BRING ALPHANUMERIC	$(M)_L \rightarrow$ AR, BINARY		56				0	0/1*	0 — L — 17**				
	BT	M, T, X	BRING TETRAD	$(M) \rightarrow T$		46				0 — T — 77						
	SAa	M, L, X	STORE ARITHMETIC REG.	$(AR)_L \rightarrow M_L$		52				0	0/1*	0 — L — 17**				
	SAR	M , , X	STORE BOTH ARITHMETIC REGISTERS	(AR1, AR2) → M → M – 31		52				1	1	0	0	0	1	
	ST	M, T, X	STORE TETRAD	$(T) \rightarrow M$		42				0 — T — 77						
	SC	M, C, X	STORE CHARACTER	$C_1 \rightarrow M$		44				0 — C — 77						
	FT	M, T, X	FIX TETRAD	$M_1 \rightarrow T$		20				0 — T — 77						
	TFI	M , , X‡	TRANSFER BLOCK FROM STORE, INCREMENT	(M, M + 1, M + ---) DST (TET 16)		24				0	1	0	0	0	0	
	TFR	M , , X‡	TRANSFER BLOCK FROM STORE, RESET	(M, M + 1, M + ---) DST (TET 16)		24				0	0	0	0	0	0	
	TTI	M , , X‡	TRANSFER BLOCK TO STORE, INCREMENT	(ORIGIN) → M, M + 1, M + ---		24				1	1	0	0	0	0	
	TTR	M , , X‡	TRANSFER BLOCK TO STORE, RESET	(ORIGIN) → M, M + 1, M + ---		24				1	0	0	0	0	0	
ARITHMETIC	ADa	M, L, X	ADD DECIMAL	$(AR) + (M)_L \rightarrow$ AR	37, 38	66				0	0/1*	0 — L — 17**				
	ABa	M, L, X	ADD BINARY	$(AR)_L + (M)_L \rightarrow M_L$	37, 39	72				0	0/1*	0 — L — 17**				
	AMa	M, L, X	ADD TO MEMORY	$(AR)_L + (M)_L \rightarrow M_L$, DECIMAL	37, 38	62				0	0/1*	0 — L — 17**				
	AT	M, T, X	ADD TO TETRAD	$(M) + (T) \rightarrow T$	39	76				0 — T — 77						
	AC	M, C, X	ADD CHARACTER	$C_1 + (M) \rightarrow M$, CARRIES	39	60				0 — C — 77						
	SDa	M, L, X	SUBTRACT DECIMAL	$(AR) - (M)_L \rightarrow$ AR	37, 38	66				1	0/1*	0 — L — 17**				
	SBa	M, L, X	SUBTRACT BINARY	$(M)_L - (AR)_L \rightarrow M_L$	37, 39	72				1	0/1*	0 — L — 17**				
	SMa	M, L, X	SUBTRACT FROM MEMORY	$(M)_L - (AR)_L \rightarrow M_L$, DECIMAL	37, 38	62				1	0/1*	0 — L — 17**				
	MPN	, L	MULTIPLY NON-CUMULAT.	(AR2) x (T20, 21) → AR1	37, 38, 40	50	0		0	0	0	0	0 — L — 7§			
	MPC	, L	MULTIPLY CUMULATIVE	(AR2) x (T20, 21) → AR1 CUM.		50				0	1	0	0 — L — 7§			
	DV	, L	DIVIDE	(AR1) ÷ (AR2) → QTN (TET 20, 21); REMAIN. → AR1	40	50				1	0	0	0 — L — 7§			
COMPARISON	CDa	M, L, X	COMPARE DECIMAL	$(AR) : (M)_L$	33–	26	0 – 7		0 – 77777	1	0/1*	0 — L — 17**				
	CBa	M, L, X	COMPARE BINARY	$(AR)_L : (M)_L$	36	70				1	0/1*	0 — L — 17**				
	CC	M, C, X	COMPARE CHARACTER	$C_1 : M$		34				0 — C — 77						
	CT	M, T, X	COMPARE TETRAD	(T):(M)		74				0 — T — 77						
	LC	M, C, X	LOGICAL COMPARE	IF M_X HAS A 1 BIT FOR EVERY 1 BIT IN C THEN =		14				0 — C — 77						
SEQUENCE CONTROL	JE	M, X	JUMP EQUAL	IF I 34 IS SET, M → CC	34	30				1	0	0	0	1	0	
	JG	M, X	JUMP GREATER	IF I 33 IS SET, M → CC	33	30				1	0	0	0	0	1	
	JS	M, X	JUMP SMALLER	IF I 36 IS SET, M → CC	36	30				1	0	0	1	0	0	
	JU	M, X	JUMP UNEQUAL	IF I 35 IS SET, M → CC	35	30				1	0	0	0	1	1	
	J	M, X	JUMP	M → CC		30				0	0	0	0	0	0	
	JC	M, I, X	JUMP CONDITIONAL	IF $I_n = 1$, M → CC (SEE INDICATOR LIST)	ALL	30				0 — I — 77						
	JL	M, N, X	JUMP LOOP	IF (N) – 1 ≠ 0, M → CC		32				0 — N — 77						
	JR	M, I, X	JUMP RETURN	IF $I_n = 1$, CC → M; M + 5 → CC	ALL	10				0 — I — 77						
	JD	M, X	JUMP DISPLAY	STOP, DISPLAY M_1, NEXT INSTR. ON RESTART		30				1	1	0	0	0	0	
	JHJ	M, X	HALT, THEN JUMP	STOP, M → CC ON RESTART		30				0	1	0	0	0	0	
SHIFT	BCn	M, S, X	BIT CIRCULATE	CIRCULATE n CHAR., S BIT POS, LEFT MAX. 7 BIT POS.		16				1	$0^n–3^\dagger$	0 — S — 7				
	BSn	M, S, X	BIT SHIFT	SHIFT n CHAR., S. BIT POS. LEFT MAX. 4 CHAR. AND 7 POS.		16				0	$0^n–3^\dagger$	0 — S — 7				
EDIT	LS	M, C, X	LOGICAL SUM	$(M) \vee C \rightarrow M$		64				0 — C — 77						
	LP	M, C, X	LOGICAL PRODUCT	$(M) \wedge C \rightarrow M$		54				0 — C — 77						
	PD	M, L, X	PAD BLANKS	BINARY ZEROS → M_L		26				0	0	0 — L — 17**				
	PD0	M, L, X	PAD ZEROS	XS-3 ZEROS → M_L		26				0	1	0 — L — 17**				
	ZS	M, L, X‡	ZERO SUPPRESS	FROM MSD CHANGE PRECEDING 0'S, ,'S AND BLANKS TO BLANKS		22				1	0	0 — L — 17**				
	ZS$	M, L, X‡	ZERO SUPPRESS AND FLOATING $ SIGN	TO $'S		22				0	0	0 — L — 17**				
	ZS*	M, L, X‡	ZERO SUPPRESS WITH ASTERISK FILL	TO *'S		22				1	1	0 — L — 17**				
	ED	M, L, X	EDIT	(AR1) → M_L CONTROLLED BY (AR2)		52				1	0	0 — L — 17**				
	TR	M, L, X	TRANSLATE	(M) → TRANSLATE ROW → M (T18 = ROW ADD.)		12				0 — L — 77						

UNIVAC 1050 INSTRUCTION REPERTOIRE

∧ = LOGICAL AND
∨ = LOGICAL OR
& = SENTINEL
‡ = M DESIGNATES THE MOST SIGNIFICANT CHAR. OF THE FIELD
† 00 IS INTERPRETED BY THE CIRCUITRY AS 04.
§ 000 IS INTERPRETED BY THE CIRCUITRY AS 010.
** 0000 IS INTERPRETED BY THE CIRCUITRY AS 020.
* IF a = 1, BIT 5 = 0; IF a = 2, BIT 5 = 1.
NOTE: SUBSCRIPT 1 INDICATES IMMEDIATE DATA, AS OPPOSED TO REFERENCED DATA.

Table 3-2. Instruction Repertoire.

TETRAD	00/00				01/01				02/02				03/03				04/04				05/05				06/06				07/07			
CHAR.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24		26	27	28	29	30	31
ROW 0					AR1												AR2															
OCTAL	0000				0004				0010				0014				0020				0024				0030				0034			

TETRAD	08/10				09/11				10/12				11/13				12/14				13/15				14/16				15/17			
CHAR.	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
ROW 0	* K2	* K1			IR1				IR2				IR3				IR4				IR5				IR6				IR7			
OCTAL	0040				0044				0050				0054				0060				0064				0070				0074			

TETRAD	16/20				17/21				18/22				19/23				20/24				21/25				22/26				23/27			
CHAR.	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
ROW 1		DESTI-NATION ADDRESS							b *	a *	c *			CONTROL COUNTER STORAGE			MULTIPLIER QUOTIENT															
OCTAL	0100				0104				0110				0114				0120				0124				0130				0134			

TETRAD	24/30				25/31				26/32				27/33				28/34				29/35				30/36				31/37			
CHAR.	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
ROW 1																																
OCTAL	0140				0144				0150				0154				0160				0164				0170				0174			

TETRAD	32/40				33/41				34/42				35/43				36/44				37/45				38/46				39/47			
CHAR.	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
ROW 2	d *	PRINTER BASE ADDRESS					e *										READER BASE ADDRESS				STANDBY BASE ADDRESS				f *	f *	f k **					
	CHANNEL 0																CHANNEL 1															
OCTAL	0200				0204				0210				0214				0220				0224				0230				0234			

TETRAD	40/50				41/51				42/52				43/53				44/54				45/55				46/56				47/57			
CHAR.	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
ROW 2	PUNCH BASE ADDRESS								g *	h *	i *	j *							CLT SEL & FUNCT. CODES													
	CHANNEL 2																CHANNEL 3															
OCTAL	0240				0244				0250				0254				0260				0264				0270				0274			

TETRAD	48/60				49/61				50/62				51/63				52/64				53/65				54/66				55/67			
CHAR.	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
ROW 3		TAPE READ BASE ADDRESS				READ CHAR. COUNT				READ ADDRESS RECORD								TAPE WRITE MEMORY BASE ADDRESS				WRITE CHAR. COUNT				WRITE ADDRESS RECORD						
	CHANNEL 4																CHANNEL 5															
OCTAL	0300				0304				0310				0314				0320				0324				0330				0334			

TETRAD	56/70				57/71				58/72				59/73				60/74				61/75				62/76				63/77			
CHAR.	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255
ROW 3	BASE MEM. ADD AND SECTOR COUNT				DRUM ADDRESS				MEMORY ADDRESS RECORD				DRUM ADDRESS RECORDED				1004 BASE ADDRESS															
	CHANNEL 6																CHANNEL 7															
OCTAL	0340				0344				0350				0354				0360				0364				0370				0374			

CHAR.	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287
ROW 4	PRINTER INTERRUPT ENTRY CHANNEL 0								CARD READER I.E. CHANNEL 1								CARD PUNCH I.E. CHANNEL 2								COMMUNICATIONS I.E. CHANNEL 3							
	JC		M		I		M																									
OCTAL	0400				0404				0410				0414				0420				0424				0430				0434			

CHAR.	288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319
ROW 4	TAPE READ I.E. CHANNEL 4								TAPE WRITE I.E. CHANNEL 5								MASS STORAGE I.E. CHANNEL 6								AVAILABLE FOR EXPANSION I.E. CHANNEL 7							
OCTAL	0440				0444				0450				0454				0460				0464				0470				0474			

CHAR.	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336
ROW 5	CLASS 1 INTERRUPT ENTRY								CLASS 2 INTERRUPT ENTRY								
OCTAL	0500				0504				0510				0514				0520

Reserved

*a = ZERO COUNT
*b = TRANSLATION TABLE ADDRESS
*c = BLOCK TRANSFER COUNT
*d = CHARACTER COUNT†
*e = LINE ADVANCE COUNT
*f = COLUMN COUNT FOR COLUMN READER
*g = HOLE COUNT (post-punch read and punch)†
*h = HOLE COUNT (wait and pre-punch)†
*i = ROW COUNT (punch)†
*j = ROW COUNT (pre-read)†
*k = ROW COUNT FOR ROW READER
*K2 = PARITY ODD LOCATION
*K1 = PARITY EVEN LOCATION

†USED ONLY BY CONTROL UNITS

Figure 3–1. Layout of First Six Rows of Store

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	Bit Shift
20	FT	Fix Tetrad
22	ZS*	Zero Suppress 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	J	Jump
30	JC	Jump Conditionally
30	JD	Jump Display
30	JE	Jump if Equal
30	JG	Jump if Greater
30	JHJ	Halt, then Jump
30	JS	Jump if Smaller
30	JU	Jump if Unequal
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	AMa	Add to Memory
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
72	SBa	Subtract Binary
74	CT	Compare Tetrad
76	AT	Add to Tetrad

Table 3-3a. Mnemonic Operations Ordered by Operation Code.

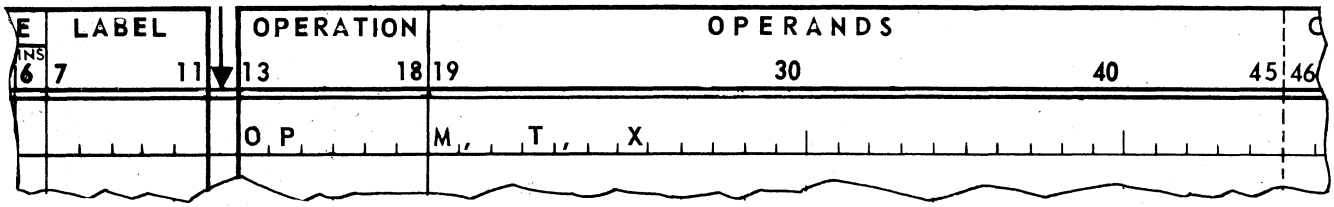
MNEMONIC INSTRUCTION	OCTAL OP CODE	DESCRIPTION	SEE PAGE
ABa	72	Add Binary	3–38
AC	60	Add Character	40
ADa	66	Add Decimal	26
AMa	62	Add to Memory	29
AT	76	Add to Tetrad	7
BAa	56	Bring Alphanumeric	16
BCn	16	Bit Circulate	72
BDa	56	Bring Decimal	14
BSn	16	Binary Shift	71
BT	46	Bring to Tetrad	6
CBa	70	Compare Binary	46
CC	34	Compare Character	47
CDa	26	Compare Decimal	44
CT	74	Compare Tetrad	9
DV	50	DiVide	36
ED	52	EDit	63
FT	20	Fix Tetrad	10
J	30	Jump	51
JC	30	Jump Conditionally	54
JD	30	Jump Display	53
JE	30	Jump if Equal	51
JG	30	Jump if Greater	51
JHJ	30	Halt, then Jump	53
JL	32	Jump Loop	59
JR	10	Jump Return	56
JS	30	Jump if Smaller	51
JU	30	Jump if Unequal	51
LC	14	Logical Comparison	48
LP	54	Logical Product	70
LS	64	Logical Sum	69
MPC	50	MultiPly Cumulative	34
MPN	50	MultiPly Noncumulative	32
PD	26	PaD blanks	68
PD0	26	PaD decimal zeros	68
SAa	52	Store Arithmetic register	17
SAR	52	Store both Arithmetic Registers	17
SBa	72	Subtract Binary	39
SC	44	Store Character	18
SDa	66	Subtract Decimal	28
SMa	62	Subtract from Memory	31
ST	42	Store Tetrad	6
TFI	24	Transfer From memory, Increment destination address	20
TFR	24	Transfer From memory, Reset destination address	20
TR	12	TRanslate	61
TTI	24	Transfer To memory, Increment origin address	22
TTR	24	Transfer To memory, Reset origin address	22
ZS*	22	Zero Suppress with asterisk fill	66
ZS$	22	Zero Suppress with floating dollar sign	66
ZS	22	Zero Suppress with no floating dollar sign	66
XF	40	eXternal Function	*

Table 3-3b. Mnemonic Operations Ordered Alphabetically.

* *The XF instruction is explained in the peripheral hardware manual for the unit to which it pertains.*

3.1 TETRAD INSTRUCTIONS

The format of a tetrad instruction is



where

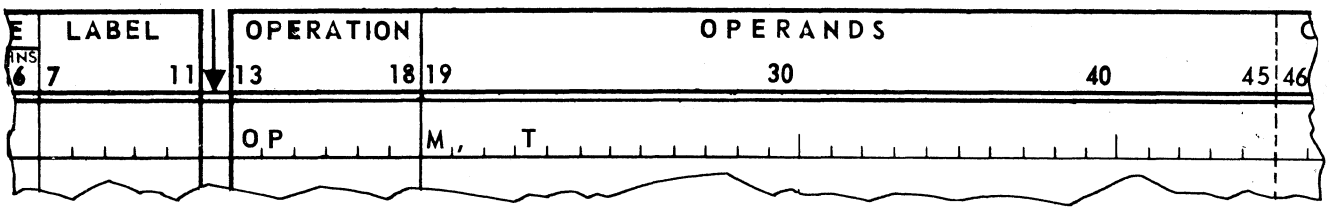
OP is the mnemonic *operation code*,

M is an expression designating the *operand address*,

T is an expression naming a *tetrad*,

X is an expression naming an *index register modifier*.

If index register modification is not desired, X may be omitted, and an instruction may be written as follows:



The assembler will, in this case, supply binary zeros in the index register portion of the instruction.

3.1.1. BRING TO TETRAD

Format: **BT** M, T, X

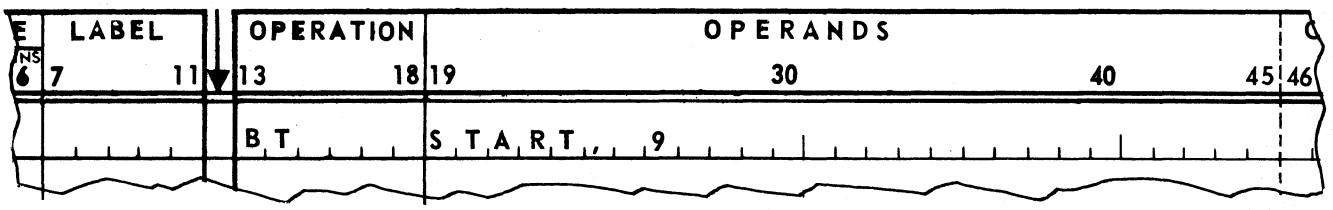
Function: Bring the four characters at M_x-3, M_x-2, M_x-1, and M_x into the specified tetrad T.

Note:

The contents of M_x-3, M_x-2, M_x-1, and M_x are not changed.

Example:

Bring the contents of the four character field labeled START into tetrad 9 (IR1).



3.1.2. STORE TETRAD

Format: **ST** M, T, X

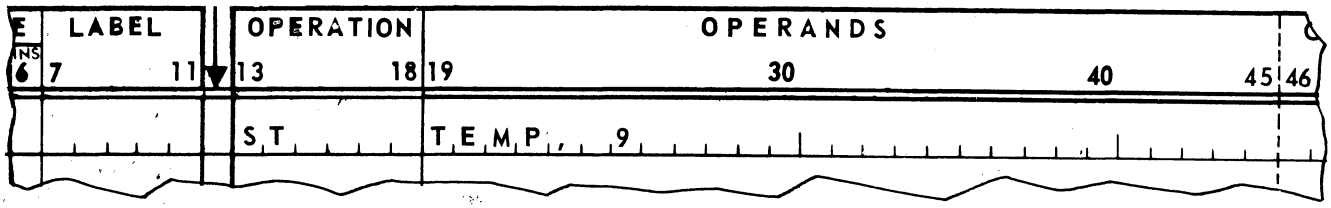
Function: Store the contents of the specified tetrad T into M_X-3, M_X-2, M_X-1, and M_X.

Note:

The contents of the tetrad are not altered.

Example:

Store the contents of tetrad 9 into the four character field labeled TEMP.



3.1.3. ADD TO TETRAD

Format: **AT** M, T, X

Function: Perform a binary addition of the four character field at M_X-3, M_X-2, M_X-1, and M_X to the specified tetrad T.

Notes:

a. The addition is a binary add. No signs are involved.

b. Both operands are always 24 bits in length.

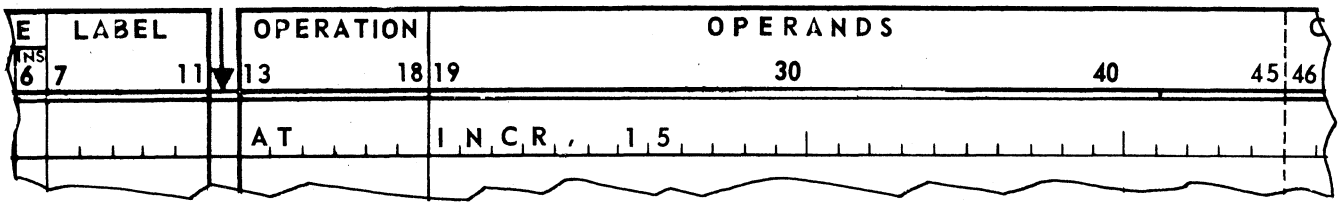
c. If overflow occurs beyond the most significant character position of the tetrad, KNB (the Binary Overflow Indicator) is set to 0. If overflow does not occur, KNB is set to 1.

d. If overflow occurs, the carry beyond the most significant character position of the tetrad is lost.

e. The field at M_X-3, M_X-2, M_X-1, and M_X is not altered.

Examples:

- Add the 24-bit field INCR to tetrad 15.



Tetrad 15 (before) = 000000 010110 101101 110111

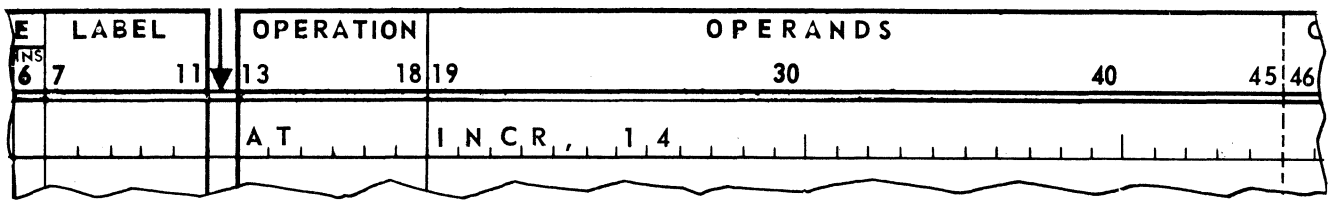
INCR (before) = 000000 000000 000000 000001

Tetrad 15 (after) = 000000 010110 101101 111000

INCR (after) = 000000 000000 000000 000001

Overflow has not occurred; KNB = 1.

- Add the 24 bit field INCR to tetrad 14.



Tetrad 14 (before) = 111111 111111 111111 111111

INCR (before) = 000000 000000 000000 000001

Tetrad 14 (after) = 000000 000000 000000 000000

INCR (after) = 000000 000000 000000 000001

Overflow has occurred; KNB = 0.

3.1.4. COMPARE TETRAD

Format: **CT** M, T, X

Function: Compare the contents of the specified tetrad T against the contents of M_x-3, M_x-2, M_x-1, and M_x.

Notes:

a. The comparison is a 24 bit binary comparison. No signs are involved.

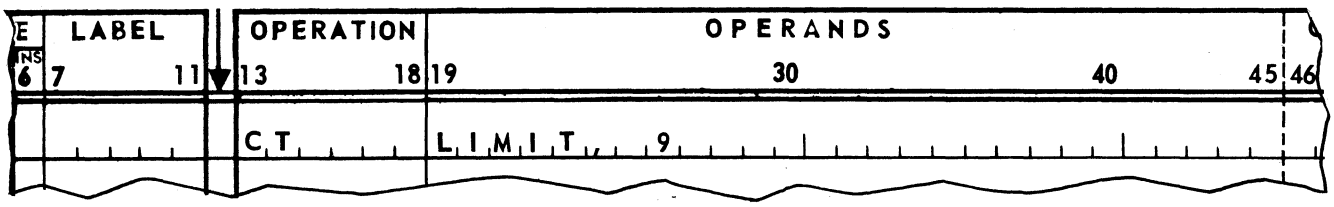
b. The result of the comparison is stored in testable indicators as follows:

Result of Comparison*	Status of Indicators after Comparison			
Indicator Number (octal)	041	042	043	044
Indicator Number (decimal)	33	34	35	36
Suggested Mnemonic	KHI (High)	KEQ (Equal)	KUQ (Unequal)	KLO (Low)
(T) = (M_x)	0	1	0	0
(T) < (M_x)	0	0	1	1
(T) > (M_x)	1	0	1	0

c. Neither operand is altered.

Example:

Compare the contents of tetrad 9 against the four character field labeled LIMIT.



If tetrad 9 contains 001000 101011 100011 010101

and LIMIT contains 000100 101011 100011 010101

the contents of tetrad 9 are greater than the contents of the field LIMIT. After this comparison is made, KHI and KUQ are set to 1, and KLO and KEQ are set to 0.

---

* *(T) means "the contents of tetrad T"; (M_x) means "the contents of M_x".*

3.1.5. FIX TETRAD

Format: **FT** M, T, X

Function: Place the 15 bit M portion of the instruction into the 15 least significant bit positions of the specified tetrad T.

Notes:

a. The value of M is placed in the tetrad specified: not the value *at the address* specified by M; but the 15 bit value of M itself. In this instruction, M is a constant. After the instruction is executed, the 15 least significant bits of the tetrad will equal the M portion of the FT instruction.

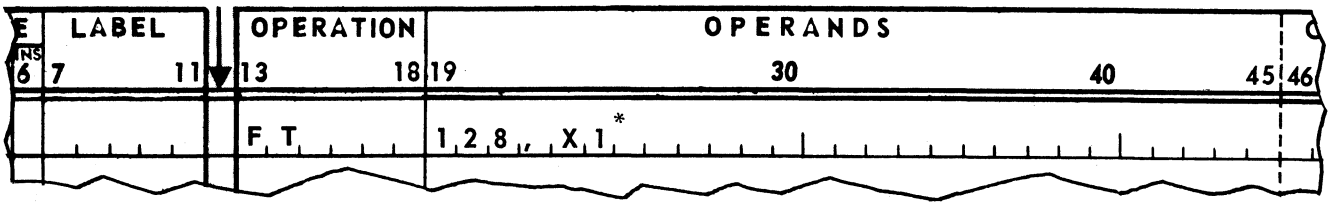
b. Binary zeros are inserted in the most significant bits of the second most significant character of the tetrad.

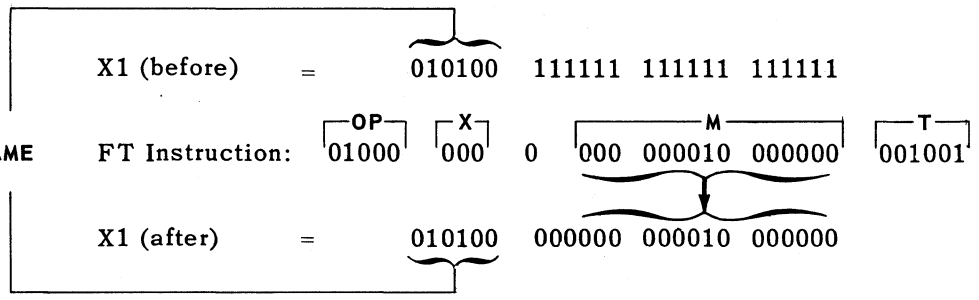
c. The most significant character of the tetrad is not affected by the instruction.

d. The interpretation of indexing is unique for this instruction. If the index register is used, the value which is stored in the tetrad is the binary sum of the M portion and the contents of the index register specified. Carries beyond the fifteenth bit are ignored.

Examples:

- Place the binary equivalent of a decimal 128 in tetrad 9 (index register 1).
- This *replaces* the contents , if any , of IR1.

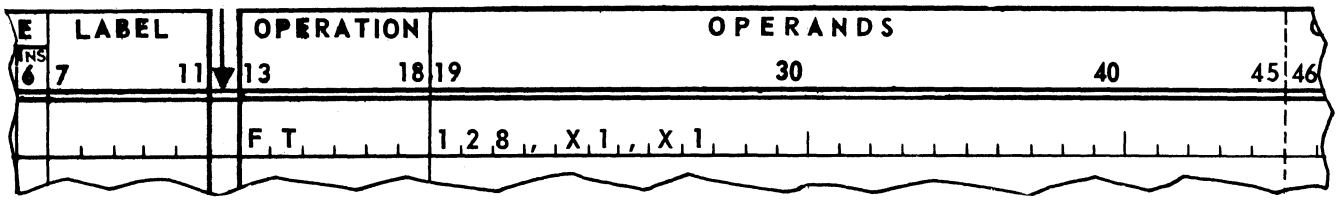


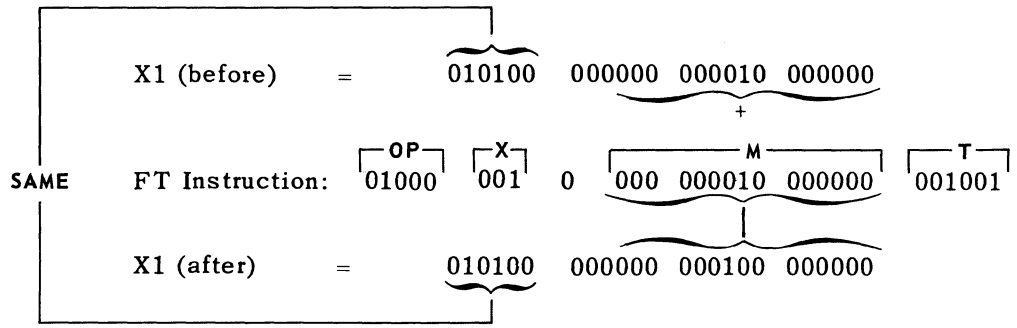


Note that the most significant character is not altered, and that binary zeros are inserted into the three most significant bit positions of the second character.

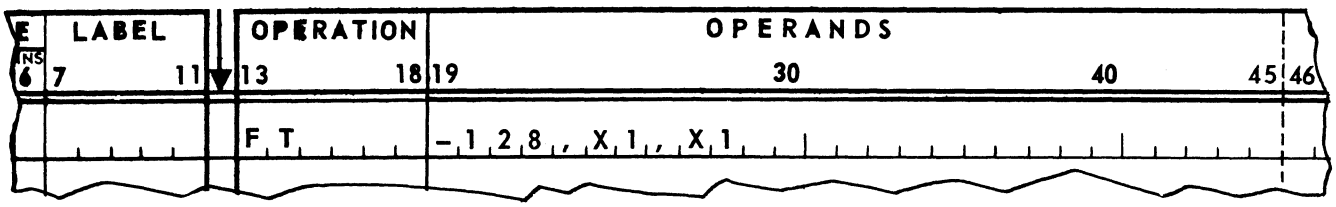
* *This form cannot be used in the processor with 4096 storage locations. PAL Jr. does not have the facility to compute a tetrad number from an index register designation. The tetrad number must be used in the T expression position.*

- Add the binary equivalent of a decimal 128 to the contents of index register 1.

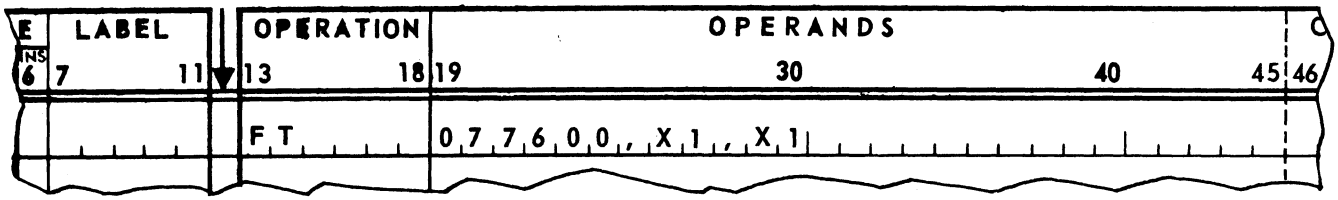




- Subtract the binary equivalent of a decimal 128 from the contents of index register 1.



or



X1 (before) = 010100 000000 000100 000000

X1 (after) = 010100 000000 000010 000000

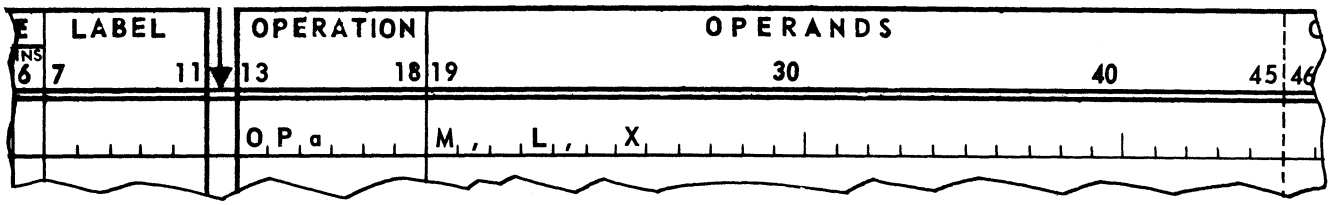
Note that the value to be subtracted is expressed either as a decimal integer with a a leading minus sign, or octally as the fifteen bit two's complement of the value.

N.B. Although the examples show values in the most significant character position of an index register tetrad, it is not advisable to have anything in that character but binary zeros. An index register tetrad should not contain anything other than an index register value.

3.2. DATA TRANSFER INSTRUCTIONS

The UNIVAC 1050 System has two types of data transfer instructions: instructions involving the arithmetic registers, and instructions which do not involve arithmetic registers. Under the first category, data is transferred into and out of arithmetic registers. In the second category, data are transferred either from one area of store to another, or from the instruction itself into store.

a. The format of data transfer instructions using the arithmetic registers is



where

OP is the mnemonic *operation code*,

a is 1 or 2, indicating *arithmetic register* 1 or *arithmetic register 2*,

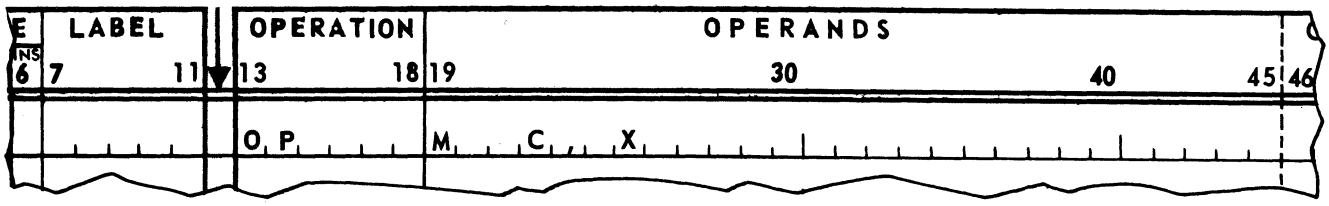
M is an expression naming the *operand address*,

L is a decimal or octal number or a defined label specifying the *operand length* in terms of characters,

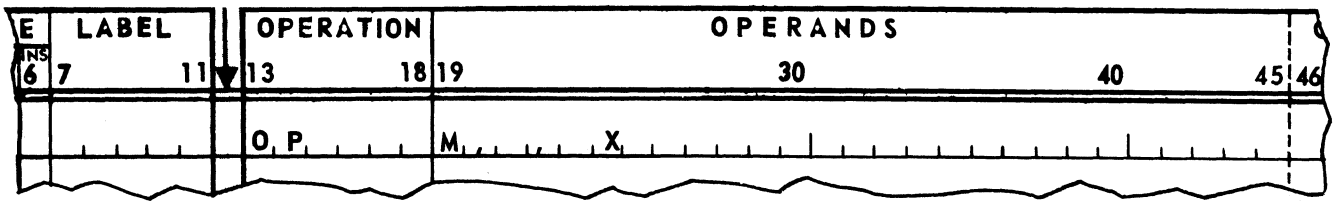
X is an expression naming an *index register*.

If index register modification is not desired, the X expression is omitted. The assembler will insert binary zeros in the index register portion of the instruction.

The format of data transfer instructions using the arithmetic registers is



(See Store Character) or



(See Transfer Block)

where

- **OP** is the mnemonic *operation code*,
- **M** is an expression naming an *operand address*,
- **C** is the actual *character* that is to be transferred,
- **X** is an *index register* expression.

Note: In all data transfer instructions, the sending field is never altered except when sending and receiving fields overlap.

3.2.1. BRING DECIMAL

Format: **BDa** M, L, X

Function: Bring the L consecutive characters whose least significant character is at M_x into the least significant characters of AR1 or 2. All zone bits except the sign bit are changed to binary zeros.

Notes:

a. L is a decimal number ranging from 1 to 16, or an equivalent expression.

b. If less than sixteen characters are transferred, a *sentinel* is inserted in that character position of the arithmetic register which is immediately to the left of the Lth character copied. This sentinel is the character &, which, in the UNIVAC 1050 character set, is 110011. Insertion of the sentinel is an automatic hardware function. Characters to the left of the sentinel are not affected.

c. The zone bits of each character with the exception of sign the bit (most significant bit of the LSD) are changed to binary zeros.

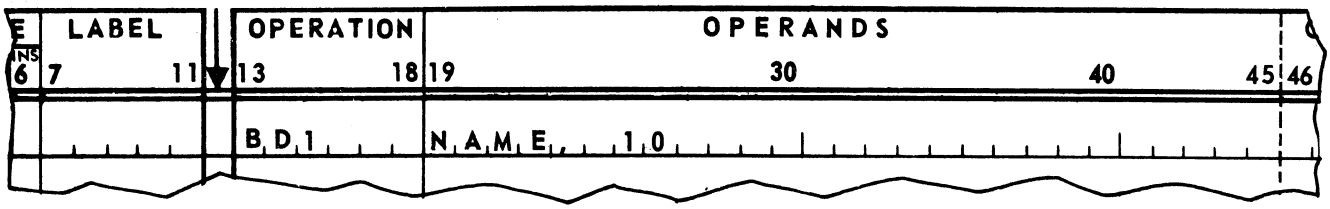
Examples:

- Bring the four character constant K1 into the four least significant character positions of AR1.

E INS 6	LABEL 7 11	▼	OPERATION 13 18	OPERANDS 19 30 40 45	46
			BD1	K1,4	

AR1 (before)	=	7	9	4	2	4	5	5	2	7	3	8	6	0	8	2	+4
K1	=													7	9	1	+6
AR1 (after)	=	7	9	4	2	4	5	5	2	7	3	8	&	7	9	1	+6

- If a field containing information other than numeric information is brought to an arithmetic register by a BDa instruction, all zone bits are deleted in the transfer, with the exception of the sign bit. For example,



results in

AR1 (before)	=	7	W	I	2	4	5	5	2	7	3	8	6	0	8	2	+ 4
NAME	=							J		W	I	L	L	I	A	M	S
AR1 (after)	=	7	W	I	2	4	&	1		6	9	3	3	9	1	4	K

The least significant character of NAME is an S(110101). When it is transferred to AR1, only the sign bit appears in AR1; the least significant zone bit is deleted, changing the S to K(100101).

3.2.2. BRING ALPHANUMERIC

Format: **BAa** M, L, X

Function: Bring the L consecutive characters whose least significant character is at M_x into the L least significant character positions of AR1 or 2.

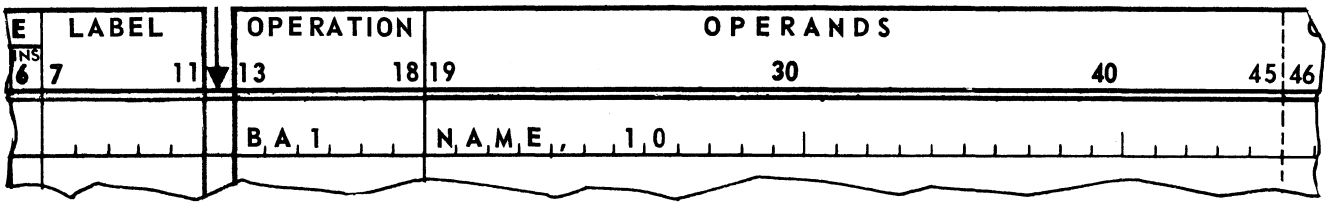
Notes:

a. L is a decimal number ranging from 1 to 16, or an equivalent expression.

b. The zone bits of all characters are transferred, and no sentinel is inserted.

Example:

Bring the 10 character field NAME into the 10 least significant positions of AR1.



AR1 (before)	=	7	9	4	2	4	5	5	2	7	3	8	6	0	8	2	+4
NAME	=							J		W	I	L	L	I	A	M	S
AR1 (after)	=	7	9	4	2	4	5	J		W	I	L	L	I	A	M	S

3.2.3. STORE ARITHMETIC REGISTER

Format: **SAa** M, L, X

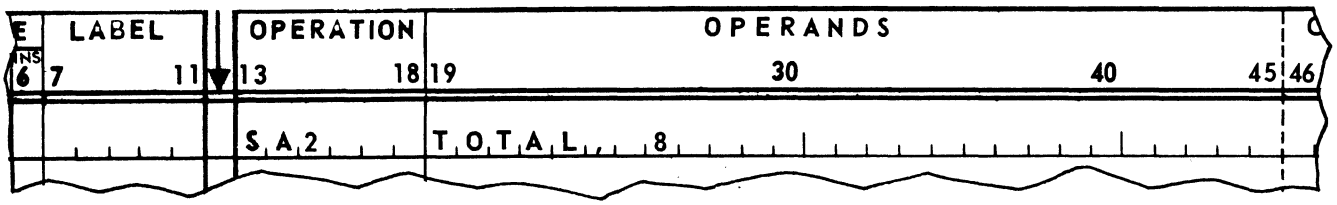
Function: Store the L least significant characters of AR1 or 2 in the L consecutive character positions whose least significant character is at M_X.

Note:

L is a decimal number ranging from 1 to 16, or an equivalent expression.

Example:

Store 8 characters from AR2 into TOTAL.



3.2.4. STORE BOTH ARITHMETIC REGISTERS

Format: **SAR** M,,X

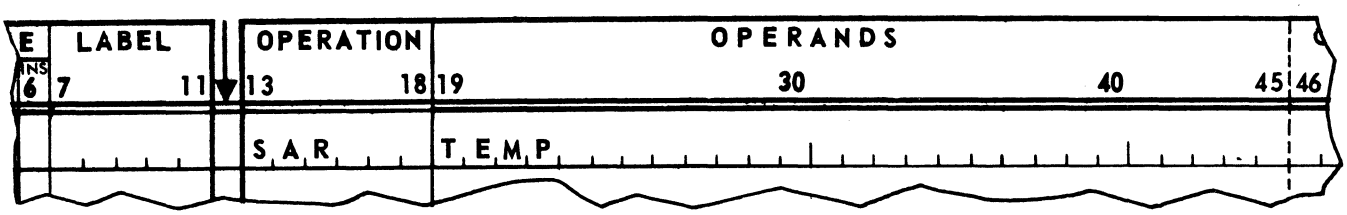
Function: Store the contents of arithmetic registers 1 and 2 in the 32 consecutive store positions whose least significant character is at M_X.

Note:

This instruction stores every position of both arithmetic registers, making the L portion of the instruction superfluous.

Example:

Store the contents of both arithmetic registers in TEMP.



3.2.5. STORE CHARACTER

Format: **SC** M, C, X MVI $M(X), C

Function: Store the six bit character C in location M_x

Notes:

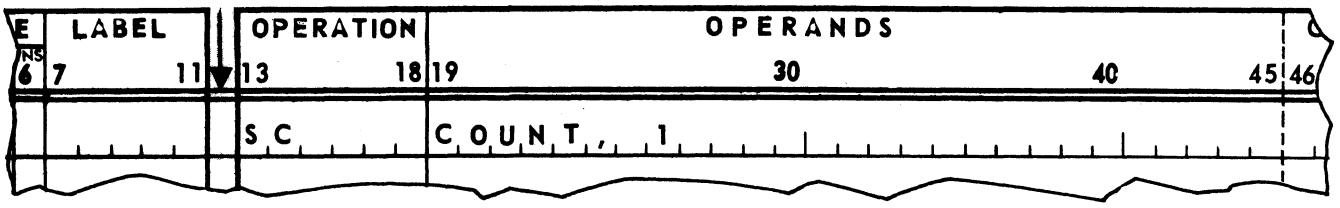
a. This instruction stores the six bit character C in M_x. The arithmetic registers are not involved in the operation, unless M_x refers to some position in an arithmetic register.

b. C is either

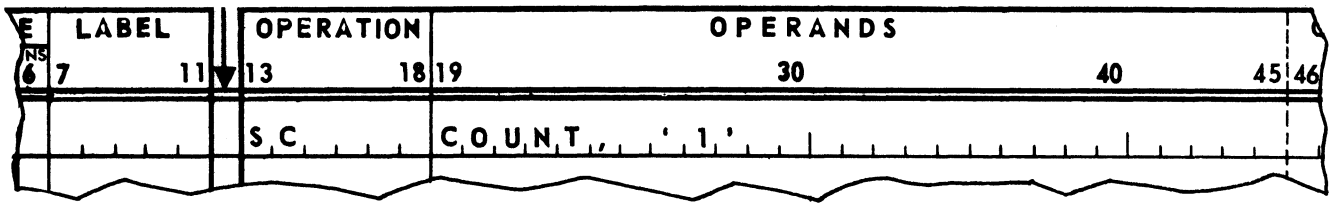
- a decimal number ranging from 0 through 63, or
- an octal number ranging from 0 through 077, or
- a single character bounded by apostrophes.

Examples:

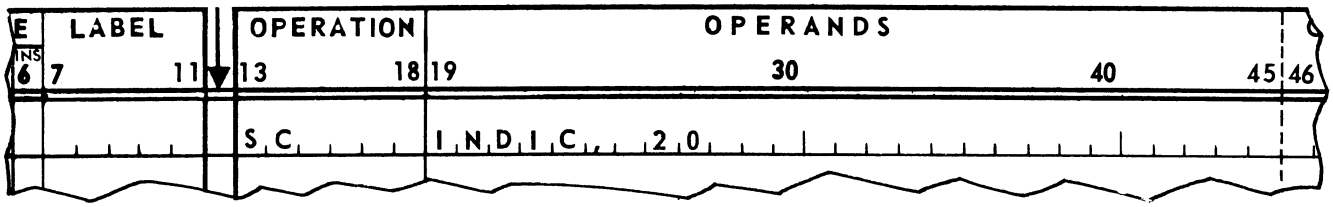
- Store a binary 1 in COUNT.



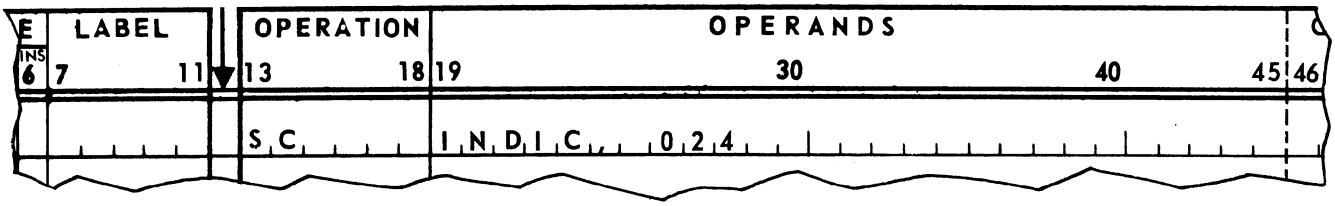
- Store the UNIVAC 1050 six bit code for the digit 1 in COUNT.



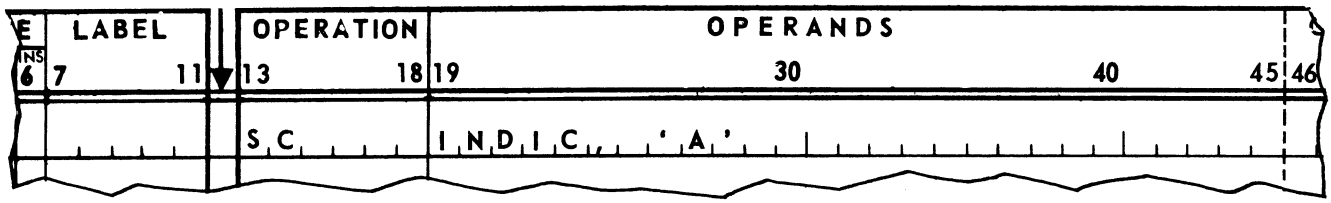
- Store the six bit configuration 010100 in INDIC. This may be written in any one of three ways:



because 010100 is the binary representation of 20;



because the binary number 010100 is noted octally as 024; or



because 010100 is the UNIVAC 1050 six bit code for the letter A.

3.2.6. TRANSFER BLOCK FROM STORE

Format: **TFR** (reset)
TFI (increment) M , , X

Function: Transfer a block of consecutive characters beginning with the most significant character at M_x to that area in store whose most significant character position is stored in DST (tetrad 16).

Notes:

a. In the transfer block instructions, M_x addresses the *most significant* character position of the sending field.

b. Prior to the execution of the TFR or TFI instruction, the binary count of characters to be transferred must be program set in the ten least significant bits of TCT (tetrad 18). The maximum number of characters that may be transferred is 1024. If the ten least significant bits of TCT are binary zeros, 1024 characters will be transferred.

   The difference between the TFR and TFI instructions is that the address in DST is reset to its original value after the TFR (Transfer From, Reset) instruction is executed. After a TFI instruction the address in DST is incremented by the number of characters specified by TCT. If TCT contains zeros, DST is incremented by 1024. The original content of TCT is not disturbed by execution of this instruction.

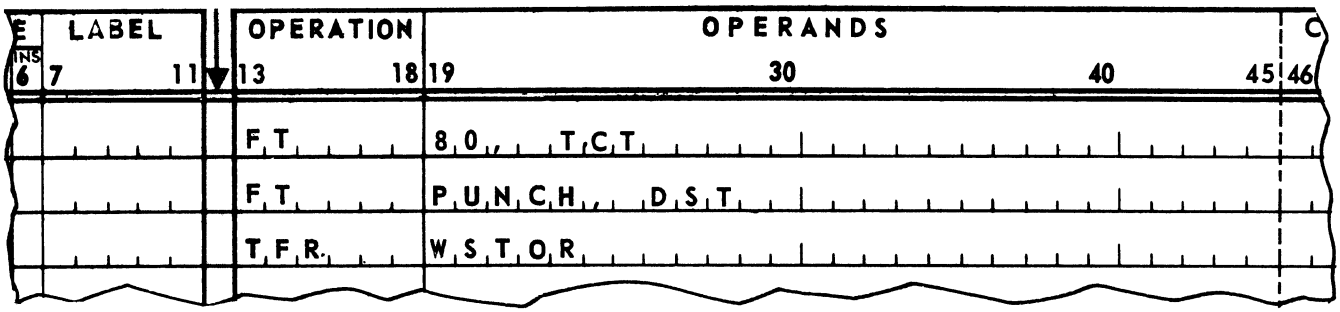
c. Prior to the execution of the TFR or TFI instruction, the address of the *most significant* position of the receiving field must be program set in DST (tetrad 16).

d. After the TFR instruction has been executed, the address in DST is reset to its original value.

e. After the TFI instruction is executed, the address in tetrad 16 is set to a value one greater than the address of the latest character of the sending field.

Example:

Transfer a block of eighty consecutive characters from the area whose most significant character position is labeled WSTOR, to the area whose most significant character position is labeled PUNCH. The sequence of instructions required to effect this transfer, using the TFR instruction, is as follows:



3.2.7. TRANSFER BLOCK TO STORE

Format: **TTR** (reset) / **TTI** (increment) M, , X

Function: Transfer a specified number of characters, the address of whose most significant character is stored in ORG (tetrad 17), to that area in store whose most significant character is M_x.

Notes:

a. In the transfer block instructions, M_x addresses the *most significant* character position of the receiving field.

b. Prior to the execution of the TTR or TTI instruction, the binary count of characters to be transferred must be program set in TCT (Tetrad 18). The maximum number of characters that may be transferred is 1024. If the ten least significant bits of TCT are binary zeros, 1024 characters are transferred.

   The difference between the TTR and TTI instructions is that the address in ORG is reset to its original value after the TTR (Transfer To, Reset) instruction is executed; after the TTI (Transfer To, Increment) instruction is executed, the address in ORG is incremented by the number of characters specified by TCT. If TCT contains zeros, ORG is incremented by 1024. On completion of the instruction TCT contains its original value.

c. Prior to the execution of the TTR or TTI instruction, the address of the most significant position of the sending field must be program set in ORG (tetrad 17).

d. After the TTR instruction has been executed, the address in ORG (tetrad 17) is reset to its original value.

e. After the TTI instruction is executed, the address in tetrad 17 is set to a value one greater than the address of the latest character of the sending field.

Example:

Using the TTI instruction, transfer a block of 80 consecutive characters from the area whose most significant character position is labeled WSTOR, to the area whose most significant character position is labeled PUNCH. After the transfer, leave ORG set to refer to WSTOR + 80.

E INS	LABEL	OPERATION	OPERANDS	C
6	7 11	13 18	19 30 40 45	46
		FT	80, TCT	
		FT	WSTOR, ORG	
		TTI	PUNCH	

3.3. ARITHMETIC INSTRUCTIONS

The UNIVAC 1050 System adds and subtracts in both the decimal mode and the binary mode, and performs multiplication and division in the decimal mode. Decimal arithmetic operations are governed by the following general rules:

a. The length of an operand in an arithmetic register is specified by the sentinel character & (110011) immediately to the left of the most significant character of the operand.

b. The length of an operand in store is specified by the instruction.

c. Operands in the arithmetic registers must always occupy the least significant character positions of the register.

d. Except for the sign bit and the zone bits of the sentinel character, the zone bits of operands are ignored and do not appear in the result.

e. If the result of a decimal arithmetic operation generates a carry beyond the most significant character position of the result field, decimal overflow occurs. This terminates the instruction, sets a testable indicator, and initiates a Class II interrupt.

In decimal add and subtract operations, the four characters (blank, +, @, ≠) having the internal form xx0000 will be converted to XS 3 zeros (000011) before the operation. Decimal operations should not be performed with any of the following invalid numeric digits:

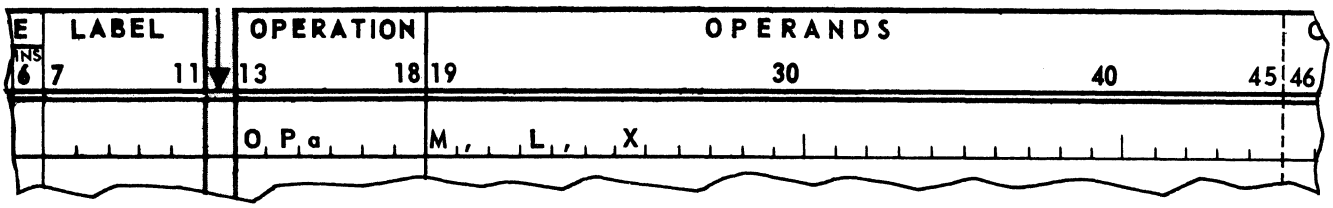
BINARY VALUE	SOURCE CHARACTERS
xx0001	] : * (
xx0010	– . $,
xx1101	= %)
xx1110	; < ' >
xx1111	[# Δ ¤

Binary arithmetic operations are governed by the following general rules:

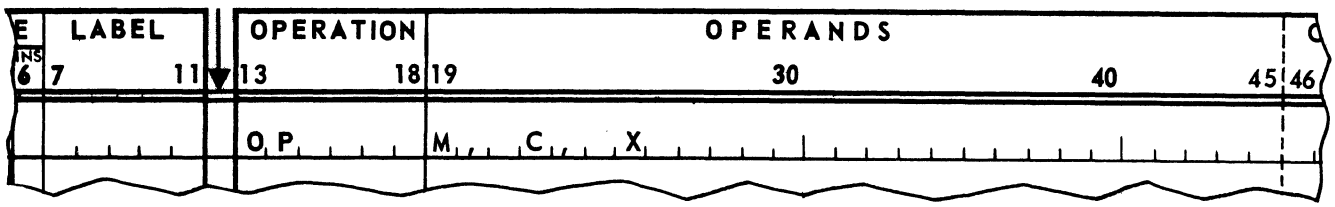
a. No algebraic signs are associated with an operand.

b. If the result of a binary arithmetic operation generates a carry beyond the most significant bit position of the result field, binary overflow occurs, which terminates the instruction and sets a testable indicator. Unlike decimal overflow, binary overflow does not initiate any interrupt.

The formats of arithmetic instructions are



and



where .

OP is the mnemonic *operation code*,

a is 1 or 2, specifying the *arithmetic register* to be used,

M is an *operand address*,

L is usually a decimal or an octal number specifying the *length* of one of the operands,

C may be

- a single *character* enclosed in apostrophes,
- a *decimal number* ranging from 0 through 63,
- an *octal number* ranging from 0 through 077, or
- a symbolic *expression*.

X is an *index register* expression.

3.3.1. ADD DECIMAL

Format: **ADa** M, L, X

Function: Perform a decimal algebraic addition of the L character field whose least significant character is in M_x to the contents of AR1 or 2 and store the result in AR1 or 2

Notes:

a. If no sentinel character appears in ARa, the working length of ARa is sixteen characters. Otherwise, the sentinel character specifies the working length of ARa.

b. Blanks (00 0000) in either operand are treated as decimal zeros (00 0011).

c. Zone bits other than the sign bit of the operand and the zone bits of the sentinel are ignored and do not appear in the result.

d. If the length of ARa is equal to or greater than L, the instruction is terminated when L characters have been added to ARa.

e. If the length of ARa is less than L, decimal zeros are substituted for the first sentinel encountered in ARa and for all higher order positions of ARa, up to and including the Lth position. A sentinel is then inserted into the position immediately to the left of the Lth position of ARa, and addition proceeds.

f. Carries are propagated up to the sentinel position. A carry into the sentinel does not alter the sentinel, but causes decimal overflow.

g. When decimal overflow occurs, the AD instruction is terminated, and an interrupt is initiated which causes a transfer of control to the decimal overflow interrupt entry, a fixed hardware location.

h. Decimal overflow interrupt can be inhibited either manually on the system console, or by programmed instruction. If interrupt has been inhibited, a testable indicator is set when overflow occurs.

i. The result of an AD instruction is recorded in testable indicators as follows:

If sum = 0, KZR (Indicator 37) is set to 1

If sum $\neq$ 0, KZR (Indicator 37) is set to 0

If sum is +, KM (Indicator 38) is set to 0

If sum is -, KM (Indicator 38) is set to 1

If overflow, KDF (Indicator 40) is set to 1

If no overflow, KDF (Indicator 40) is set to 0

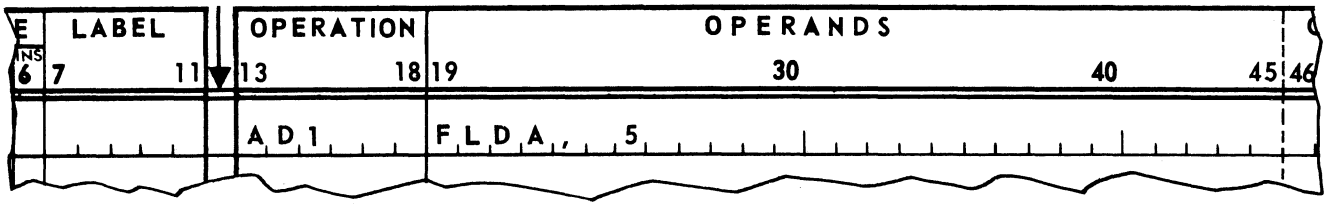
j. A decimal zero result is always positive, with the following exceptions:

(1) $-0 + (-0) = -0$

(2) a false zero result (such as that obtained by adding 99 and 1, which should yield 100 but, on account of the sentinel, results in &00) will carry the sign of the full result.

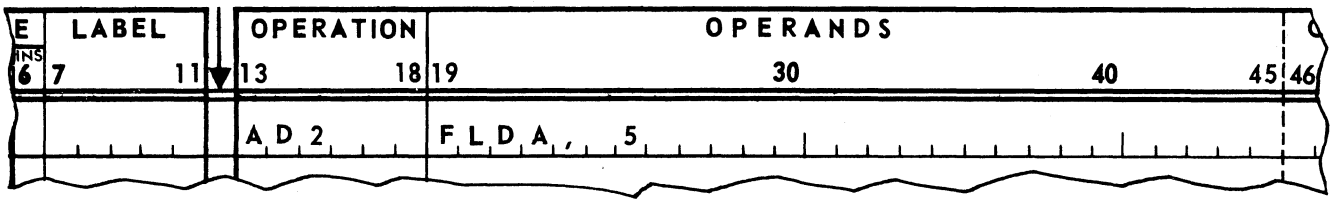
Examples:

- Add the five digit field labeled FLDA to arithmetic register 1.



AR1 (before) =	3	2	6	9	8	&	1	2	&	0	0	3	4	5	1	+6
FLDA =												9	2	3	0	+4
AR1 (after) =	3	2	6	9	8	&	1	2	&	0	1	2	6	8	2	+0

- Add the five digit field labeled FLDA to arithmetic register 2.



AR2 (before) =	3	2	6	9	8	&	1	2	0	0	0	3	4	&	1	+6
FLDA =												9	2	3	0	+4
AR2 (after) =	3	2	6	9	8	&	1	2	0	0	&	9	2	3	2	+0

3.3.2. SUBTRACT DECIMAL

Format: **SDa** M, L, X

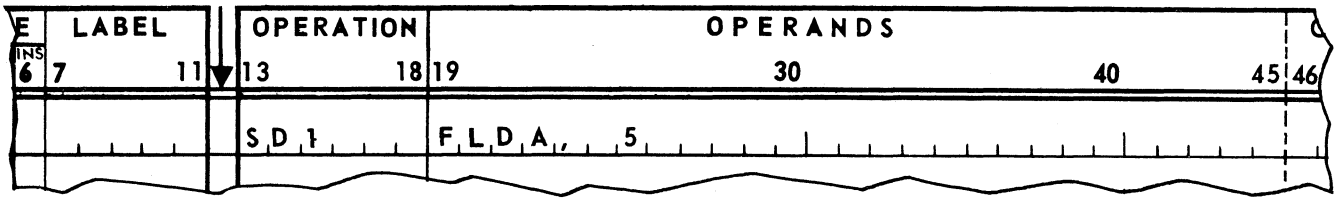
Function: Perform a decimal algebraic subtraction of the L character field whose least significant character is in M_x from the contents of AR1 or 2. and store the result in AR1 or 2.

Note:

This instruction operates identically to the ADa instruction, with the sole exception that the operation is a subtraction. Otherwise, the notes under the ADa instruction apply.

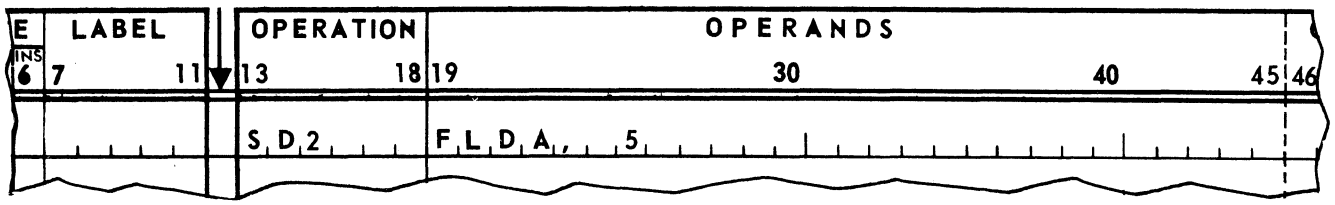
Examples:

- Subtract the five digit field labeled FLDA from arithmetic register 1.



AR1 (before) =	3	2	6	9	8	&	1	2	&	0	0	3	4	&	1	+6
FLDA =												9	2	3	0	+4
AR1 (after) =	3	2	6	9	8	&	1	2	&	0	&	9	2	2	8	−8

- Subtract the five digit field labeled FLDA from arithmetic register 2.



AR2 (before) =	3	2	6	9	8	&	1	2	&	0	0	9	5	0	0	+6
FLDA =												9	2	3	0	+4
AR2 (after) =	3	2	6	9	8	&	1	2	&	0	0	0	2	7	0	+2

3.3.3. ADD TO MEMORY

Format: **AMa** M, L, X

Function: Perform a decimal algebraic addition of the L least significant characters of AR1 or 2 to the L consecutive characters in store whose least significant character is M_x, and place the sum in the field at M_x.

Notes:

a. Addition is terminated when L characters have been added from the arithmetic register.

b. If a sentinel is encountered in the arithmetic register before the Lth character is added, addition proceeds as though the sentinel and all characters to the left of the sentinel, up to the Lth position, were decimal zeros. The contents of the arithmetic register, however, are unchanged.

c. Carries are allowed to propagate up to the Lth character in store. A carry occurring when the Lth character is added terminates the addition, and decimal overflow occurs, causing an interrupt and setting KDF (Indicator 40) to 1. The carry is lost.

d. Except for the sign bit, zone bits are ignored, and they do not appear in the result.

e. A zero result is always positive, except for the following cases:

(1) $-0 + (-0) = -0$

(2) A false zero result occurring when a carry is lost carries the sign of the full true result.

f. The results of the AM instruction are recorded in testable indicators as follows:

If the sum = 0, KZR (Indicator 37) is set to 1

If the sum $\neq$ 0, KZR (Indicator 37) is set to 0

If the sum is +, KM (Indicator 38) is set to 0

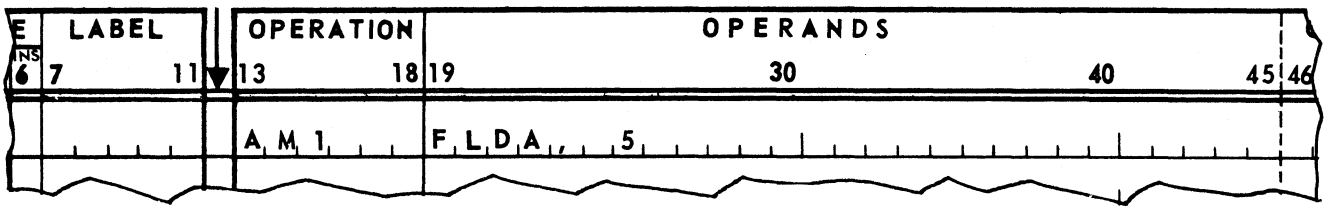
If the sum is −, KM (Indicator 38) is set to 1

If overflow, KDF (Indicator 40) is set to 1

If no overflow, KDF (Indicator 40) is set to 0

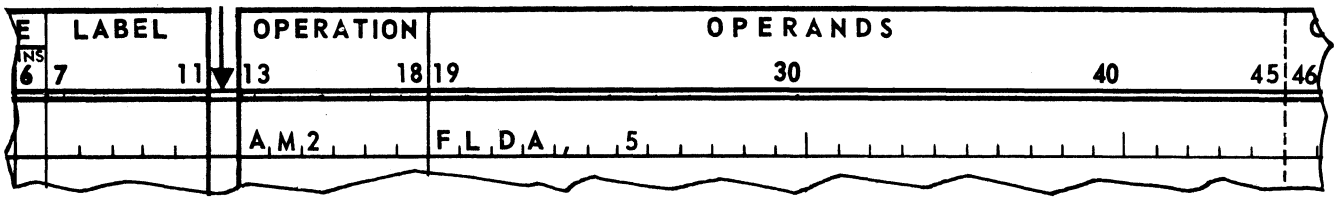
Examples:

- Add the 5 least significant characters of arithmetic register 1 to the field labeled FLDA.



AR1 (before & after) =	3	2	6	9	8	&	1	2	&	0	0	0	4	5	1	+6
FLDA (before) =												9	2	3	0	+4
FLDA (after) =												9	6	8	2	+0

- Add the 5 least significant characters of arithmetic register 2 to the field labeled FLDA.



AR2 (before & after) =	3	2	6	9	8	7	1	2	0	0	0	3	4	&	1	+6
FLDA (before) =													9	2	3	9	+4
FLDA (after) =													9	2	4	1	+0

3.3.4. SUBTRACT FROM MEMORY

Format: **SMa** M, L, X

Function: Perform a decimal algebraic subtraction of the L least significant characters of AR1 or 2 from the L characters whose least significant character is at M_X, and store the difference in the field at M_X.

Note:

This instruction operates identically to the AMa instruction, except that the operation is a subtraction. Otherwise, the notes under the AMa instruction apply.

Example:

Subtract the 5 least significant characters of AR2 from the 5 characters at BLNCE.

```
E  LABEL      OPERATION      OPERANDS
NS
6  7      11  13      18  19          30          40      45 46
           SM2            BLNCE,5
```

AR2 (before & after) =	5	6	3	0	4	8	7	7	3	4	&	4	0	2	6	+3
BLNCE (before) =												8	7	9	4	+7
BLNCE (after) =												4	7	6	8	+4

3.3.5. MULTIPLY NON-CUMULATIVE

Format: **MPN ,L**

Function: Clear arithmetic register 1 to decimal zeros; multiply the multiplicand in arithmetic register 2 by the L least significant characters of MLR (tetrads 20 and 21); store the product, without sentinel, in arithmetic register 1.

Notes:

a. Both the multiplicand and the multiplier must be positioned by previous instructions. The multiplier must be stored in the least significant character positions of tetrads 20 and 21 (MLR) and must be preceded by decimal zeros if less than eight characters. This field is eight characters long and is treated as one field. The L specifies the L least significant characters of the field.

b. The length of the multiplicand is determined by the sentinel in AR2. This implies that the multiplicand must be loaded in AR2 by means of a BD2 instruction, rather than a BA2 instruction. No blanks should appear in the multiplicand field.

c. The number of characters in the multiplicand plus the number of characters in the multiplier must not exceed 16. The product is limited to the sixteen character positions of AR1. If the number of characters in the product exceeds 16, undetected overflow may occur. *A carry* from the 16th position of AR1 will cause a detected decimal overflow which will set indicator 40 and cause a class II interrupt unless interrupt is inhibited.

The following are permissible combinations in multiplication.

L (number of characters) in MLR	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

d. The sign of the product is governed by normal algebraic rules. Like signs yield a positive product, and unlike signs yield a negative product.

e. If a sentinel appears in the least significant character position of AR2, the multiplicand is considered to be –0 and, depending on the sign of the multiplier, AR1 is cleared to either minus zeros or plus zeros.

f. Multiplication destroys the contents of MLR (tetrads 20 and 21) but leaves the multiplicand in AR2 unaltered.

g. The result of the MPN instruction is recorded in the testable indicators as follows:

If product = 0, KZR (Indicator 37) is set to 1

If product ≠ 0, KZR (Indicator 37) is set to 0

If product is +, KM (Indicator 38) is set to 0

If product is −, KM (Indicator 38) is set to 1

If overflow occurs, KDF (Indicator 40) is set to 1.

h. The index register and M portions of the instruction are ignored. When an MPN instruction is coded, a blank expression must be written for the M portion.

Example:

Multiply a five digit multiplicand by a one digit multiplier. The first two instructions position the multiplicand and the multiplier.

E NS	LABEL 6 7 11	OPERATION 13 18	OPERANDS 19 30 40 45 46
		BT	K5,MLR
		BD2	RATE,5
		MPN	,1

(Note: In the illustrations below, the x's represent characters the values of which are immaterial to the MPN instruction.)

Before the MPN instruction is executed:

AR1	=	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
AR2	=	x	x	x	x	x	x	x	x	x	x	&	6	6	3	7	+7
MLR	=											0	0	0	0	0	0	0	+5

The multiplicand is 66377, and the multiplier is 5.

After the MPN instruction is executed:

AR1	=	0	0	0	0	0	0	0	0	0	0	3	3	1	8	8	+5
AR2	=	x	x	x	x	x	x	x	x	x	x	&	6	6	3	7	+7
MLR	=											x	x	x	x	x	x	x	x

3.3.6. MULTIPLY CUMULATIVE

Format: **MPC ,L**

Function: Multiply the multiplicand in arithmetic register 2 by the L least significant characters of MLR (tetrads 20 to 21), and add the product to the contents of arithmetic register 1.

Notes:

This instruction operates identically to the MPN instruction, with the following differences:

a. AR1 is not cleared to zero.

b. Any sentinel in AR1 is treated as a decimal zero.

c. When the product is added to the contents of AR1, the absolute values of the products and of AR1 are added, and AR1 takes the sign of the product.

d. KZR is not affected if the cumulative product is 0.

e. If a carry occurs beyond the most significant character position of AR1, KDF (Indicator 40) is set to 1 and decimal overflow occurs. The carry is lost.

f. Blanks in either AR1 or AR2 will result in an erroneous product.

Example:

Multiply a five digit multiplicand by a one digit multiplier. AR1 contains the value 7163398238.

E INS 6	LABEL 7 — 11	OPERATION 13 — 18	OPERANDS 19 — 30 — 40 — 45	46
		BT	K5,MLR	
		BD2	RATE,5	
		MPC	,1	

(Note: In the illustrations below, the x's represent characters the values of which are immaterial to the MPC instruction.)

Before the MPC instruction is executed:

AR1 =	x	x	x	x	x	&	7	1	6	3	3	9	8	2	3	+8
AR2 =	x	x	x	x	x	x	x	x	x	x	&	6	6	3	7	+7
MLR =									0	0	0	0	0	0	0	+5

The multiplicand is 66377, and the multiplier is 5.

After the MPC instruction is executed:

AR1 =	x	x	x	x	x	x	7	1	6	3	7	3	0	1	2	+3
AR2 =	x	x	x	x	x	x	x	x	x	x	&	6	6	3	7	+7
MLR =									x	x	x	x	x	x	x	x

(66377 x 5 = 331885)

3.3.7. DIVIDE

Format: **DV ,L**

Function: Divide the dividend in arithmetic register 1 by the divisor in arithmetic register 2 and store an L character quotient in the L least significant character positions of QTN (tetrads 20 and 21).

Notes:

a. The sign of the quotient is determined by normal algebraic rules.

b. The maximum size of the quotient is eight characters.

c. The length of the divisor is specified by the sentinel in AR2.

d. The length of the dividend must be equal to the length of the divisor plus the quotient, L. If the length of the dividend is less, AR1 must be extended by padding decimal zeros.

e. The absolute value of the divisor shifted L positions to the left must be greater than the absolute value of the dividend. If it isn't, an Improper Divide (Class II) interrupt occurs and KDF (Indicator 40) is set to one.

f. The length of the quotient plus the length of the divisor cannot be greater than 16; otherwise the quotient will be incorrect.

g. If no sentinel is present in AR2, the computer stalls on the DV instruction.

h. If a sentinel is present in the least significant position of AR2, the computer stalls on the DV instruction.

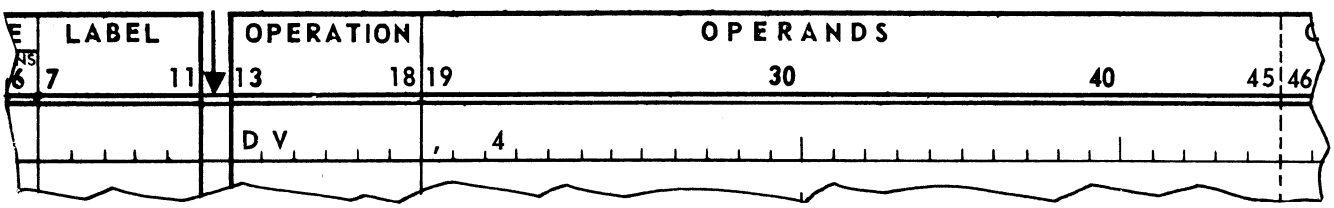
i. The remainder, if any, is stored in AR1, and carries the sign of the original dividend.

j. The M and X portions of the instruction are ignored. However, a blank expression must be coded for the M expression.

k. Blanks cannot be substituted for decimal zeros in this instruction.

Example:

Divide a five digit field in AR1 by a two digit field in AR2, and store a four digit quotient in QTN (tetrads 20 and 21).



(Note: The x's in the illustrations represent characters which are immaterial in the operation of the divide instruction.)

Before the DV instruction is executed,*

AR1 =	0	0	0	0	0	0	0	0	0	0	0	0	7	2	3	+0
AR2 =	x	x	x	x	x	x	x	x	x	x	x	x	x	&	1	+7
QTN =									x	x	x	x	x	x	x	x

after the DV instruction is executed,

AR1 =	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	+5
AR2 =	x	x	x	x	x	x	x	x	x	x	x	x	x	&	1	+7
QTN =									x	x	x	x	0	4	2	+5

* *If the quotinet 042529 were desired, it could be obtained by specifying L as 6 and using a dividend of 723000.*

3.3.8. ADD BINARY

Format: **ABa** M, L, X

Function: Perform a binary addition of the L least significant characters of AR1 or 2 to the L characters in store whose least significant character is in M_x and place the sum in the field at M_x.

Notes:

a. The ABa instruction adds from ARa into memory, i.e., the sum appears in M_x.

b. The contents of the arithmetic register are not changed, unless M_x addresses either arithmetic register in which case the contents of the two registers could be added or the content of one arithmetic register could be added to itself.

c. The instruction specifies L characters; therefore the number of bits involved is always 6L.

d. No algebraic signs are associated with the operands.

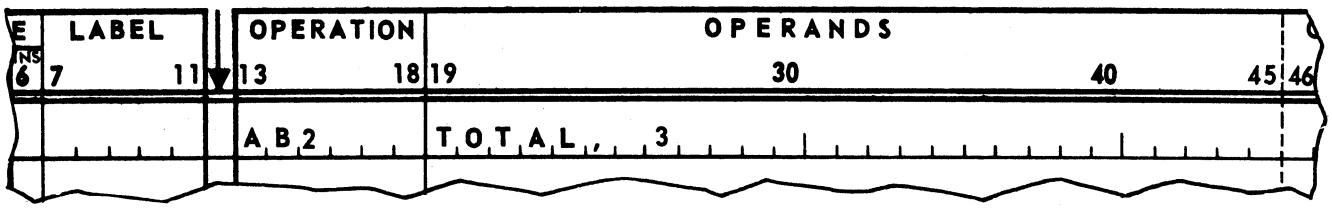
e. A carry beyond the most significant bit of the operand in store is lost, but KNB (Indicator 39) is set to 0.

f. If there is no carry beyond the most significant bit of the operand in store, KNB (Indicator 39) is set to 1.

g. If the contents of the L store positions are binary zeros after the addition, KZR (Indicator 37) is set to 1; otherwise, it is set to 0.

Example:

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



AR2 (before & after)	=	x	x	x	x	x	x	x	x	x	x	x	x	1	1	0	(octal 040403)
TOTAL (before)	=													A	5	W	(octal 241071)
TOTAL (after)	=													E	9	Z	(octal 301474)

3.3.9. SUBTRACT BINARY

Format: **SBa** M, L, X

Function: Perform a binary subtraction of the L least significant characters of AR1 or 2 from the L characters in store (whose least significant character is in M_X), and place the binary difference in the field at M_X.

Notes:

a. The SBa instruction subtracts the contents of ARa from M_X placing the difference in M_X.

b. The contents of the arithmetic register are not changed, unless M_X addresses either arithmetic register.

c. The instruction specifies L characters; therefore the number of bits involved is always 6L.

d. No algebraic signs are associated with the operands.

e. This instruction adds the 2's complement of the value in the arithmetic register to the value in store.

f. Carries propagate up to, but not beyond, the most significant bit of the field in store. A carry beyond the most significant bit is lost, but sets Indicator 39 to 1. If there is no carry KNB (Indicator 39) is set to 0. This differs from the setting described under the ABa instruction, because a carry beyond the most significant bit indicates that the result in M_X is the true difference. If there is no carry, the result is the complement of the true difference.

g. If the contents of the L character positions are binary zeros after the subtraction, KZR (Indicator 37) is set to 1; otherwise it is set to 0.

Example:

Subtract, in binary, the 3 least significant characters of AR2 from QNTY.

E INS 6	LABEL 7 ... 11	OPERATION 13 ... 18	OPERANDS 19 ... 30 ... 40 ... 45	46
		S B 2	Q N T Y , , 3	

AR2 (before & after) = | x | x | x | x | x | x | x | x | x | x | x | x | 6 | 5 | 3 | (octal 111006)

QNTY (before) = | 8 | 6 | K | (octal 131145)

QNTY (after) = | – |] | # | (octal 020137)

3.3.10. ADD CHARACTER

Format: **AC** M, C, X

Function: Add, in binary, the character C to the contents of M_X, and store the sum in M_X.

Notes:

a. The binary value contained in the last six bit positions of the instruction is the increment.

b. The binary sum is stored at M_X.

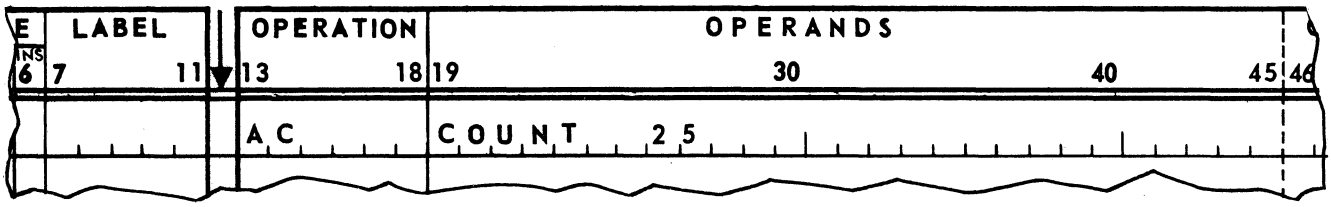
c. Carries are allowed to propagate into M_X-1 and as far as necessary.

d. A carry beyond the most significant bit of M_X sets KNB (Indicator 39) to 0; if there is no carry beyond this position, KNB is set to 1.

e. The arithmetic registers are not affected by this instruction, unless M_X addresses a character in AR1 or AR2.

Example:

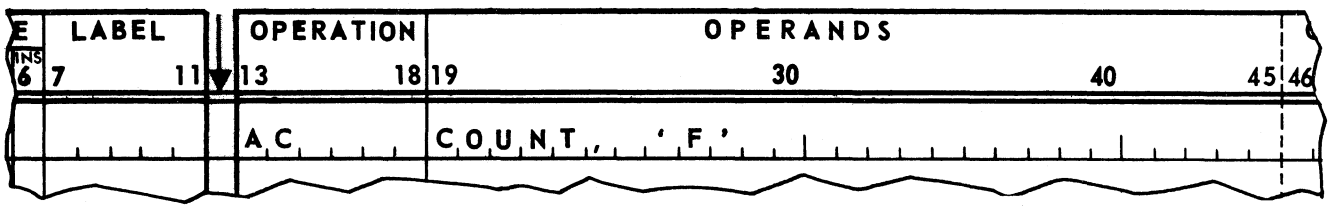
Add the binary equivalent of a decimal 25 (011001) to the character labeled COUNT.



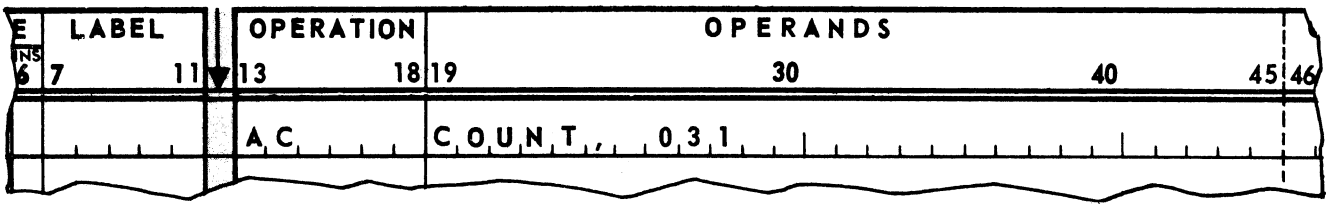
COUNT (before) = decimal 25

COUNT (after) = decimal 50

This instruction may also be written as



Since 011001 is the UNIVAC 1050 character code for the letter F, it may also be written as



In all three cases, the assembler produces the bit configuration 011001.

3.4. COMPARISON INSTRUCTIONS

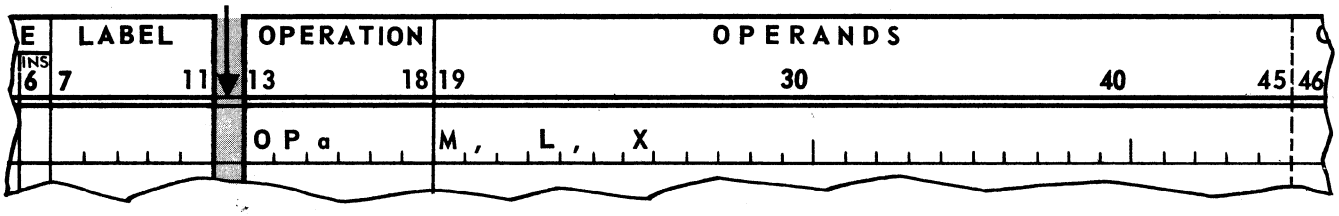
These instructions compare two values, and the result of the comparison is recorded in the following indicators:

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

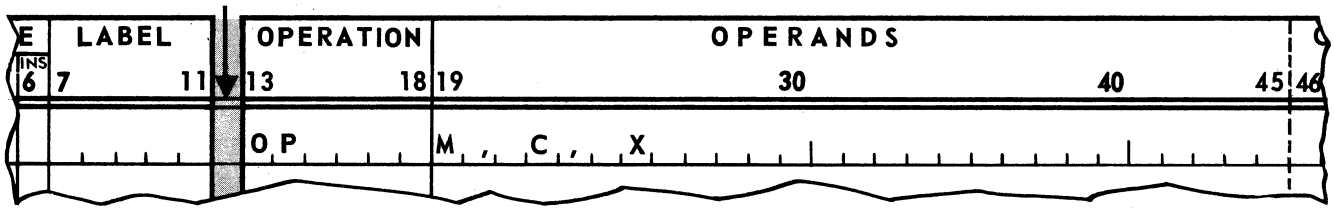
The settings of these indicators can be tested by the program and appropriate action can be taken.

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

The formats of the comparison instructions are



and



where

OP is the mnemonic *operation code*,

a is 1 or 2, specifying the *arithmetic register* to be used,

M is an *operand address*,

L is an expression specifying *operand length*,

C may be

- a single alphanumeric *character* enclosed in apostrophes,
- a *decimal number* ranging from 0 through 63,
- an *octal number* ranging from 0 through 077,
- a *symbolic expression*,

X is an *index register* expression.

3.4.1. COMPARE DECIMAL

Format: **CDa** M, L, X

Function: Compare algebraically a signed number comprising all digits to the right of the rightmost sentinel in AR1 or 2 to a signed numeric field of L (maximum of L is 16) decimal digits, starting with the least significant digit location at M_X. Except for the sign bit zone portions are ignored; all characters are treated as decimal digits.*

Notes:

a. If the signs of the two fields are unlike, the comparison is terminated immediately.

b. If no sentinel is present in the specified arithmetic register, all sixteen characters of the register are used in the comparison.

c. If there is a difference in the field lengths of the two operands, decimal zeros are assumed in the implied high order positions of the shorter field, i.e., if one field is five characters long and the other is eight characters long, the CD instruction assumes that the five character field is preceded by three decimal zeros.

d. Comparison stops upon locating a sign difference or when the most significant character of the longer field has been compared algebraically.

e. The result of the algebraic comparison is stored in testable indicators as follows:

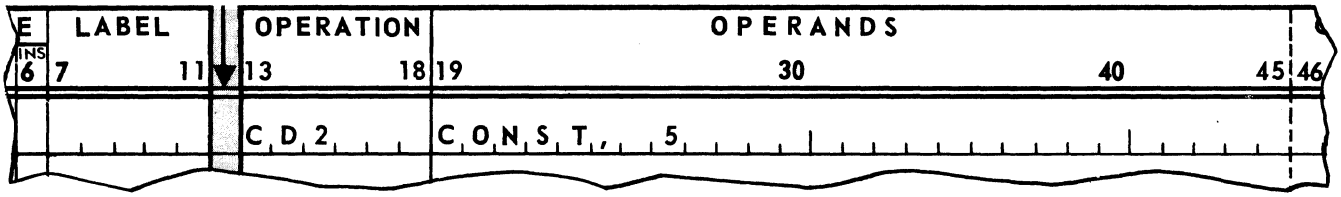
Result of Comparison**	Status of Indicators after Comparison			
Indicator Number (octal)	041	042	043	044
Indicator Number (decimal)	33	34	35	36
Suggested Mnemonic	KHI (High)	KEQ (Equal)	KUQ (Unequal)	KLO (Low)
(ARa) = (M_X)	0	1	0	0
(ARa) < (M_X)	0	0	1	1
(ARa) > (M_X)	1	0	1	0

* *Compare Binary should be employed for comparisons involving alphabetics.*

** *(ARa) means "the contents of ARa", and (M_x) means "the contents of M_x".*

Example:

Compare decimally the five character field at CONST with the seven character field in AR2.



AR2 (before) & (after) =	x	x	x	x	x	x	x	x	&	0	0	1	4	4	0	+ 0
CONST (before) = & (after)												1	3	5	8	+ 2

The CD instruction assumes that CONST is a seven character field and treats it as if it were 0013582. Since the contents of AR2 are greater than the contents of CONST, the KHI and KUQ indicators are set.

3.4.2. COMPARE BINARY

Format: **CBa** M, L, X

Function: Perform an absolute binary comparison of the L least significant character positions of AR1 or 2 to the L characters whose least significant location is at M_X.

Notes:

a. The comparison is an absolute binary comparison; therefore the operation continues until L characters have been compared.

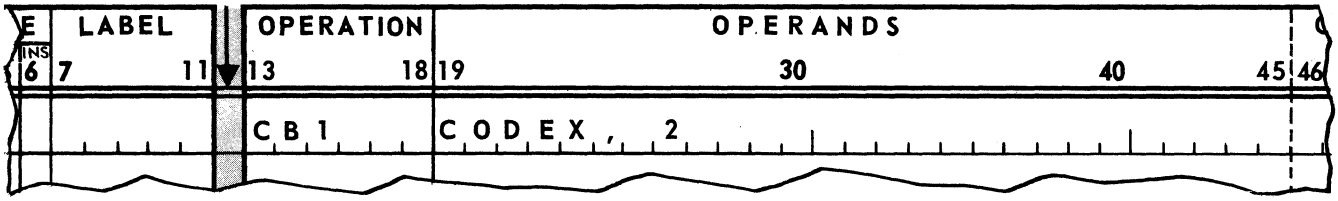
b. Since L specifies a length in terms of characters, the number of bits involved in the comparison is 6 L.

c. The result of the comparison is recorded in the testable indicators as follows:

Result of Comparison	Status of Indicators after Comparison			
Indicator Number (octal)	041	042	043	044
Indicator Number (decimal)	33	34	35	36
Suggested Mnemonic	KHI (High)	KEQ (Equal)	KUQ (Unequal)	KLO (Low)
(ARa) = (M_X)	0	1	0	0
(ARa) < (M_X)	0	0	1	1
(ARa) > (M_X)	1	0	1	0

Example:

Compare the two characters at CODEX against the two least significant characters of AR1.



CODEX = | B | 3 | (010101 000110)

AR1 = | x | x | x | x | x | x | x | x | x | x | x | x | x | x | B | 2 | (010101 000101)

Since the absolute binary value in AR1 is less than that in M_X, the KUQ (35) and KLO (36) Indicators are set to 1.

COMPARE CHARACTER – CC M, C, X

Function: Perform an absolute binary comparison of the character represented by C to the character in M_x.

Notes:

a. C may be

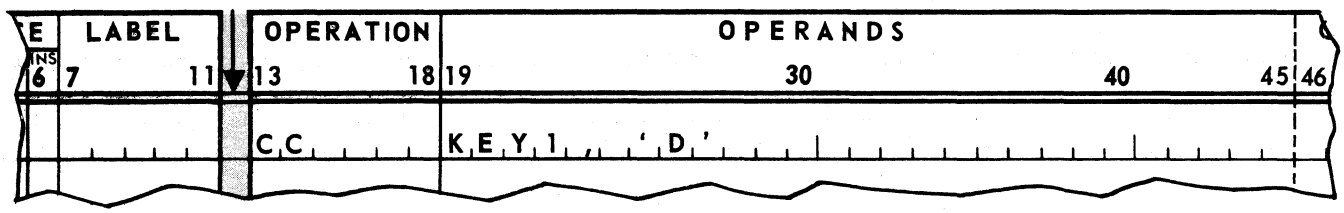
- a single alphanumeric character enclosed in apostrophes,
- a decimal number ranging from 0 through 63,
- an octal number ranging from 0 through 77, or
- a symbolic expression.

b. The result of the comparison is stored in the testable indicators as follows:

Result of Comparison	Status of Indicators after Comparison			
Indicator Number (octal)	041	042	043	044
Indicator Number (decimal)	33	34	35	36
Suggested Mnemonic	KHI (High)	KEQ (Equal)	KUQ (Unequal)	KLO (Low)
$C = (M_x)$	†	1	0	†
$C < (M_x)$	0	0	1	1
$C > (M_x)$	1	0	1	0

Example:

Compare the character at KEY1 against the character D.



If KEY1 contains the character G (011010), the character D (010111) is less than KEY1, the Unequal (35) and Low (36) Indicators are set to 1.

†*Unchanged.*

LOGICAL COMPARE – LC M, C, X

Function: Test the character at M_X for the presence of 1 bit in every bit position that corresponds to those bit positions of C which contain 1 bits.

Notes:

a. Only those corresponding bit positions in M_X and C containing 1 bits are compared. All other bits are ignored.

b. If all bit positions in M_X that correspond to the 1 bits in C are also 1 bits, M_X and C are considered to be equal, and KEQ is set. Otherwise, C is considered to be higher in value.

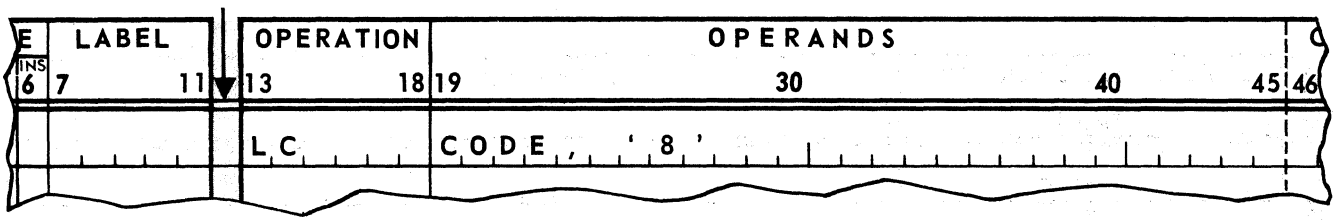
c. If C is binary zeros, M_X and C are considered to be equal, regardless of the contents of M_X.

d. The result of the comparison is recorded in the testable indicators as follows:

Result of Comparison	Status of Indicators after Comparison			
Indicator Number (octal)	041	042	043	044
Indicator Number (decimal)	33	34	35	36
Suggested Mnemonic	KHI (High)	KEQ (Equal)	KUQ (Unequal)	KLO (Low)
$C = (M_X)$	†	1	0	†
$C \neq (M_X)$	1	0	1	0

Example:

Compare the 1 bits of the character '8' with the 1 bits of the character at CODE.



†*Unchanged.*

Since the UNIVAC 1050 bit configuration for the character '8' is 001011, CODE will be considered equal to 8 if the first, second, and fourth bits (counting from the rightmost bit) of CODE are 1 bits. Therefore the following bit configurations will set the Equal Indicator:

001011 (8)

011011 (H)

101011 (Q)

111011 (Y)

001111 ([)

011111 (#)

101111 (Δ)

111111 (¤)

Any other bit configurations will set the KUQ and KHI Indicators.

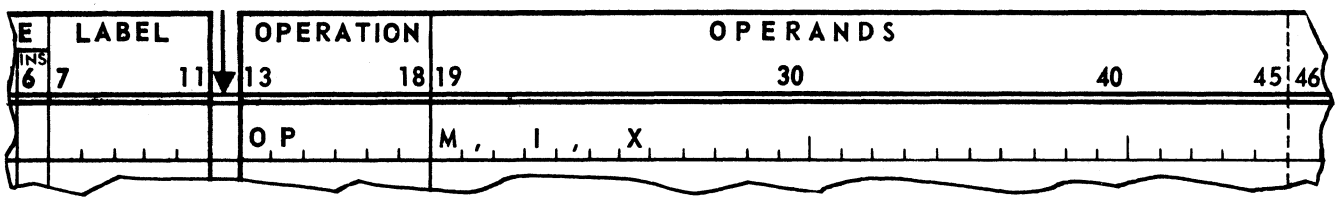
3.5. SEQUENCE CONTROL INSTRUCTIONS

Normally the instructions in a UNIVAC 1050 program are accessed and executed sequentially, i.e., in the order that they appear in main store. Whenever the conditions of the program require a break in this normal sequence, the sequence control instructions are used.

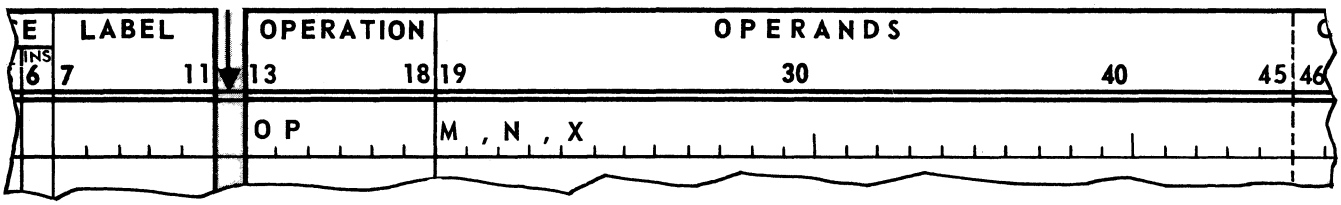
In the normal, sequential execution of instructions, the control counter is automatically incremented by five whenever an instruction is executed. This provides the control unit with the address of the next instruction to be accessed by the control register.

Sequence control instructions override this normal incrementation by changing the contents of the control counter. This transfers program control to some instruction which is not in sequence.

The format of a sequence control instruction is



or



where

OP is the mnemonic *operation code,*

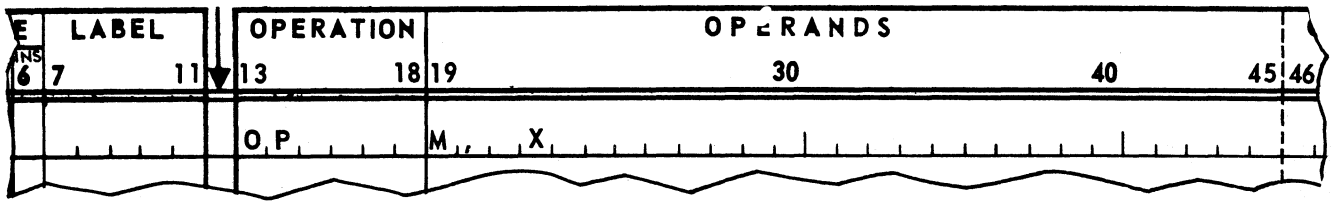
M is the *label* of an instruction,

I is an expression identifying an *indicator,*

N is an expression giving a *number,*

X is an *index register* expression.

In some special forms of sequence control instructions, the I expression is implied by the operation code, in which case the instruction is written as follows:



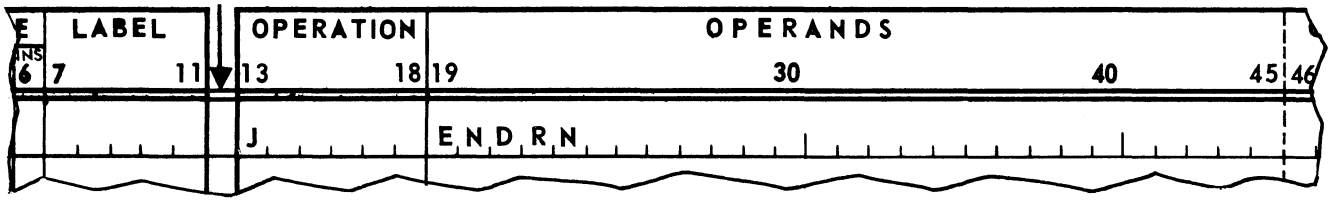
3.5.1. JUMP

Format: **J** M, X

Function: Transfer program control unconditionally to the instruction labeled M_x.

Example:

Transfer program control unconditionally to the instruction labeled ENDRN.



3.5.2. JUMP IF GREATER

Format: **JG** M, X

3.5.3. JUMP IF EQUAL

Format: **JE** M, X

3.5.4. JUMP IF UNEQUAL

Format: **JU** M, X

3.5.5. JUMP IF SMALLER

Format: **JS** M, X

Function: Test the comparison indicator specified by the operation code. If the indicator is set to 1, transfer control to the instruction labeled M_x; if it is set to 0, execute the next instruction in sequence.

Notes:

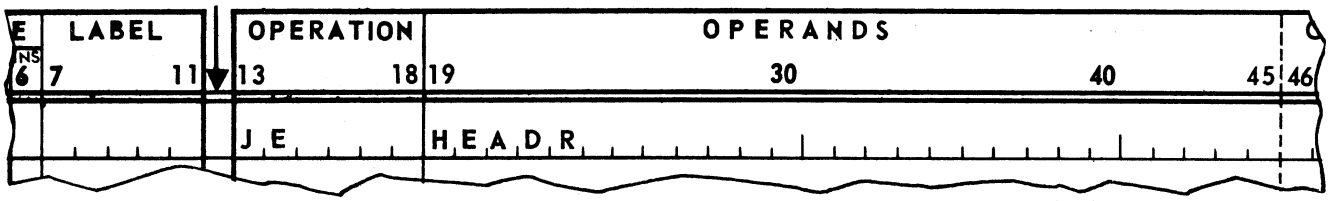
a. These four conditional jump instructions are not available with the PAL Jr. System which is employed on the 4096 character storage capacity Central Processor. The PAL Jr. System uses the Jump Conditional instruction with indicator 33 for Jump if Greater, 34 for Jump if Equal, 35 for Jump if Unequal, and 36 for Jump if Smaller.

b. These instructions are used in conjunction with the comparison instructions (CT, CDa, CBa, CC, and LC). After a comparison instruction has been executed, one or more of the comparison indicators (KHI, KEQ, KUQ, KLO) is set. The comparison jumps test these indicators.

c. If a second expression appears on a line, it is interpreted as an index register expression.

Example:

A comparison instruction has just been executed. If the Equal Indicator was set as a result of the comparison, transfer control to the instruction labeled HEADR.



HALT, THEN JUMP – JHJ M, X

Function: Stop the computer. When the Program Start button on the console is depressed, transfer program control to the instruction labeled M_X.

Notes:

a. This instruction is provided to allow the program to stop the computer and await some action on the part of the operator before processing is resumed.

b. When the computer stops, the control counter already contains the address of the instruction to be executed when the Program Start button is depressed.

JUMP DISPLAY – JD M, X

Function: Stop the computer and display the binary value at M_X on the console display lights. When the Program Start button on the console is depressed, execute the next instruction in sequence.

Note:

The notes under the JHJ instruction apply, except that M_X is ignored and is used for display purposes only. When the Program Start button is depressed, control is transferred to the next instruction in sequence.

JUMP CONDITIONAL* - JC M, I, X

Function: Transfer control according to the specification I.

Notes:

- Conditional

1. The following indicators are associated with arithmetic operations.

Indicator	Control is transferred to M_x if:
37 (KZR)	The result of the last arithmetic operation was zero.
38 (KM)	The result of the last decimal arithmetic operation was negative.
39 (KNB)	No overflow occurred in the last binary add operation or overflow did occur in last binary subtract operation.
40 (KDF)	Decimal overflow occurred since the last test for this condition.**

2. The following values of I test the Sense Indicators. The Sense Indicators are devices which are set and reset by program instructions (See Unconditional, Note 1). The Sense Indicators exist as a convenience for the programmer; while the comparison indicators are set and reset as a result of a comparison, the Sense Indicators may be set and reset arbitrarily.

Indicator	Control is transferred to M_x if:
53	Sense Indicator 1 is set to 1.
54	Sense Indicator 2 is set to 1.
55	Sense Indicator 3 is set to 1.

3. The following indicators test the setting of the Sense Switches, which are set and reset manually. These Sense Switches are on the console.

Indicator	Control is transferred to M_x if:
50	Sense Switch 1 is ON
51	Sense Switch 2 is ON
52	Sense Switch 3 is ON

---

* *A listing of the various values of I and of their significance is provided on page 3-E-4.*

** *KDF is reset to zero when tested. All other indicators are unaffected by testing.*

- Unconditional

a. The following indicators set and reset the testable Sense Indicators (See Conditional, Note b). The Sense Indicators are not tested. After they are set or reset, the instruction causes an unconditional transfer of control to M_x.

Indicator	Function
18	Set Sense Indicator 1 to 1 and jump to M_x.
19	Set Sense Indicator 2 to 1 and jump to M_x.
20	Set Sense Indicator 3 to 1 and jump to M_x.
21	Reset Sense Indicator 1 to 0 and jump to M_x.
22	Reset Sense Indicator 2 to 0 and jump to M_x.
23	Reset Sense Indicator 3 to 0 and jump to M_x.

b. The indicators 00 and 24 cause an unconditional transfer of control to M_x, i.e., they cause the JC instruction to operate identically to the J instruction. The J instruction is actually a JC instruction which the assembler automatically supplies with the indicator 00.

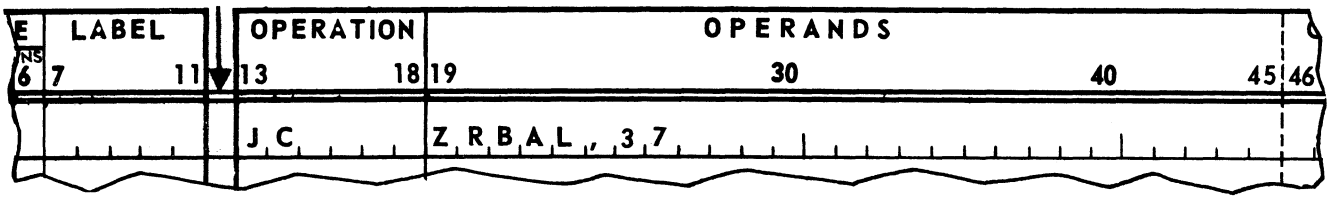
c. The indicators 32 and 56 do not test any of the hardware indicators. Control is always transferred to the next instruction in sequence; in other words, a JC instruction with an I expression of either 32 or 56 is a skip, or a No-Operation instruction.

d. Indicator 41 stores the settings of comparison indicators 33 and 34 and the arithmetic indicators 37-40 in M_x. It is unnecessary to store indicators 35 and 36 (unequal and low) with indicators 33 and 34 (high and equal) stored. These indicators are stored in character M_x in the following order: 40, 39, 38, 37, 34, 33.

Indicator 42 sets indicators 33-40 from M_x. Be careful that indicator 40 (decimal overflow) is set properly. If by setting indicator 42 indicator 40 is set, a Class II interrupt will be caused.

Example:

If the result of the last arithmetic operation was zero, transfer control to ZRBAL.



3.5.9. JUMP RETURN

Format: **JR** M, I, X

Function: Test the indicator specified by I*. If it is set to 1, store the address of the next instruction in sequence in the *address portion* of the instruction at M_X. Program control is then transferred to the instruction immediately following the one at M_X.

Notes:

a. This instruction provides the programmer with the facility of breaking program sequence and executing a subroutine; then it returns program control to the instruction immediately following the JR instruction.

b. In order that control be returned to the instruction immediately following the JR, the last line of the subroutine must be a J to the same label (M_X) as the label to which the JR was executed.

c. The instruction at M_X must be a J instruction with no index register expression. The address portion of this J instruction is usually zero, although any value may be placed in it. This portion is destroyed when the JR to that line is executed.

d. The JR instruction tests the same indicators as those which the JC instruction does. The only difference between a JR and a JC instruction, other than in timing, is that a JR stores the address of the instruction immediately following it in the address portion of the instruction labeled M_X and transfers control to $M_X + 5$, while the JC merely transfers control to M_X.

e. Additional values of I are as follows:

Indicator	Function
16	Stop the computer. When the Program Start button on the console is depressed, store the address of the instruction immediately following in the address portion of the instruction at M_X, and transfer control to $M_X + 5$.
33 (KHI)	If the High Indicator is set, store the address of the instruction immediately following in the address portion of the instruction at M_X, and transfer control to $M_X + 5$.
34 (KEQ)	If the Equal Indicator is set, store the address of the instruction immediately following in the address portion of the instruction at M_X, and transfer control to $M_X + 5$.

* *A listing of the various values of I and of their significance is provided in Table 4-1, Page 4-4.*

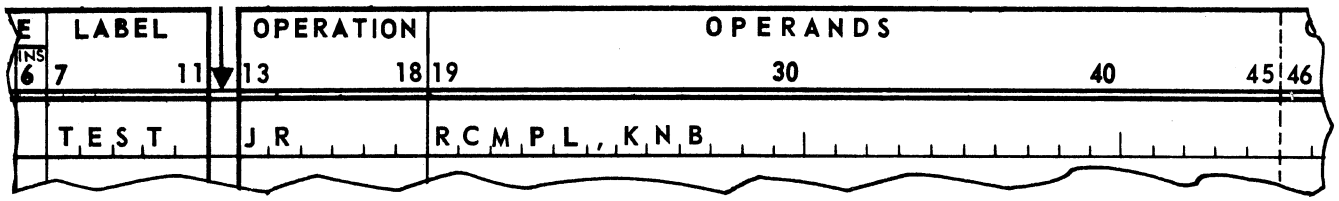
35 (KUQ) If the Unequal Indicator is set, store the address of the instruction immediately following in the address portion of the instruction at M_x, and transfer control to $M_x + 5$.

36 (KLO) If the Low Indicator is set, store the address of the instruction immediately following in the address portion of the instruction at M_x, and transfer control to $M_x + 5$.

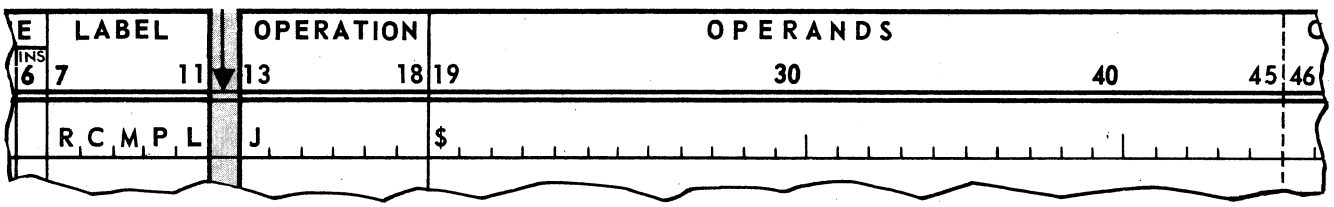
Example:

A binary subtract instruction has just been executed. If no overflow has occurred, the result is the complement of the true result, and must be recomplemented. The subroutine whose first instruction is RCMPL must be performed. In either case, processing must continue whether or not the recomplementation subroutine has been executed.

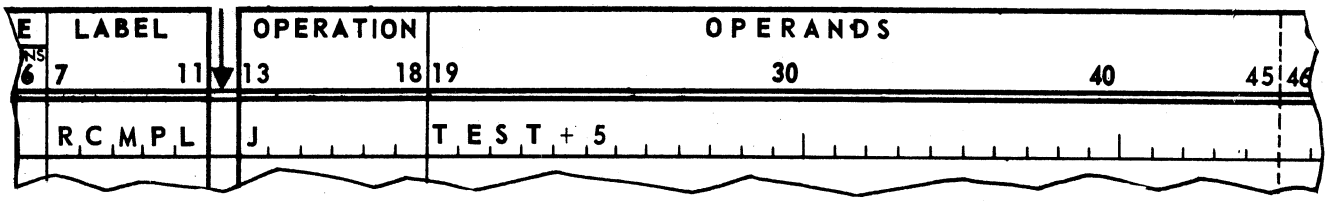
Test for binary overflow; if none has occurred, perform the subroutine whose first line is labeled RCMPL; otherwise, continue processing.



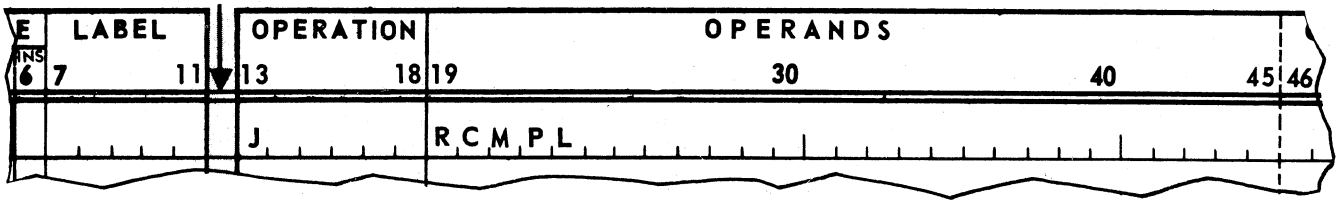
The line labeled RCMPL might be



If the JR instruction above effects a transfer of control, this line will become, effectively,



and the last line of the subroutine must be



which will transfer control to TEST + 5.

JUMP LOOP – JL M, N, X

Function: Test the N portion of the instruction against binary zeros (000000). If equal, execute the next instruction in sequence. If unequal, decrement the N portion by a binary 1 (000001) and restore the new value of N in the N portion of the instruction in main store. If the new value of N is still unequal to 000000, transfer program control to the instruction at M_x; otherwise, execute the next instruction in sequence.

Notes:

a. The N portion of the instruction is never decremented past 000000.

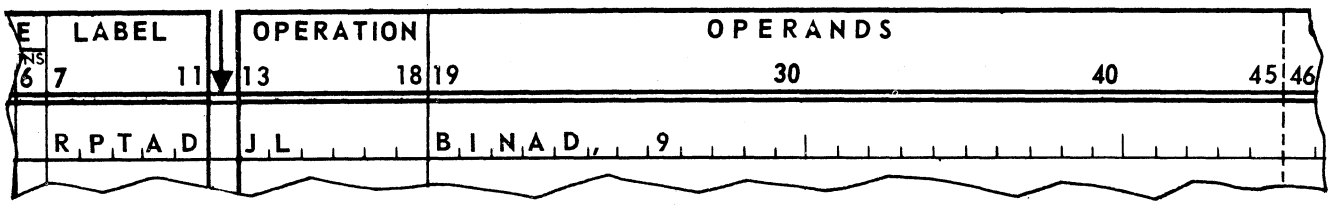
b. The N portion serves as the working counter for the instruction. It is decremented by 000001 every time that the JL instruction is executed.

c. The maximum value of N is 63.

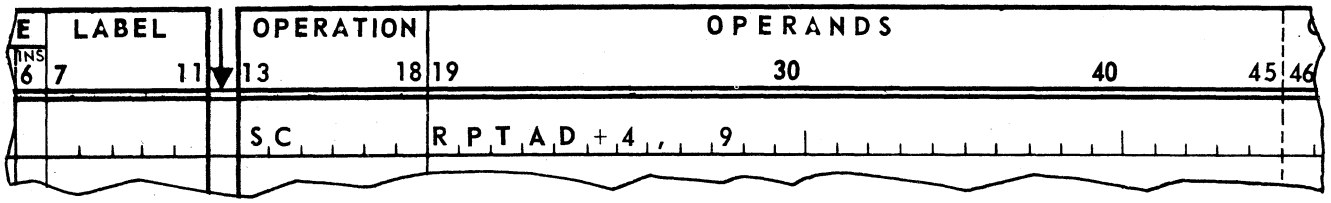
d. After N has been decremented to 000000, N must be reset by a program instruction (usually an SC instruction) to its original value. Otherwise, N will remain at 000000 the next time that the JL instruction is executed.

Example:

Execute the subroutine, the first line of which is labeled BINAD, 9 times. The line of the subroutine may be coded as follows:



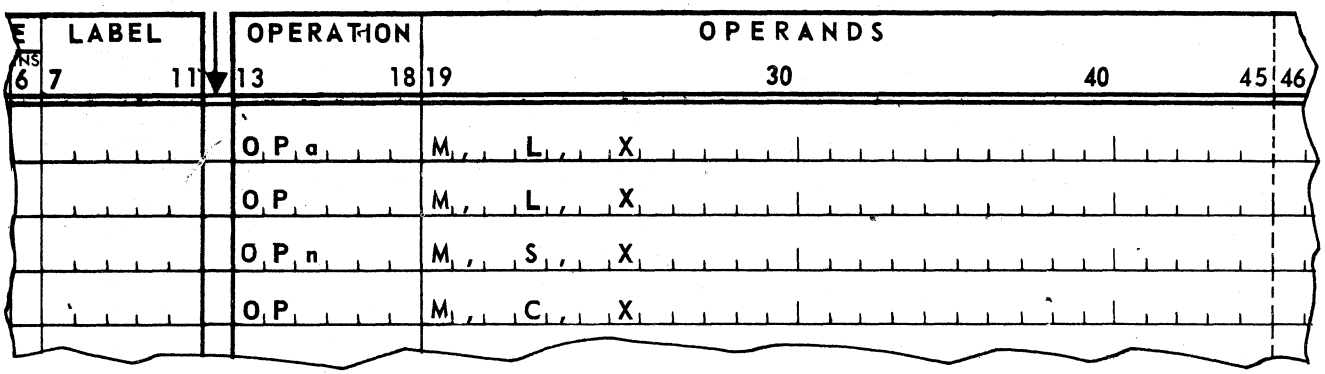
It is recommended that the next line be



so that, when the N portion of the line labeled RPTAD is decremented to 000000, it is reset to its original value of 9.

6. EDITING INSTRUCTIONS

The editing instructions in the UNIVAC 1050 instruction repertoire are used to alter the form of information in store by means other than arithmetic instructions. The formats of editing instructions are:



where

- OP is a mnemonic operation code,
- a is 1 or 2, specifying an arithmetic register,
- n is the number of characters involved in a bit shift,
- M is an expression specifying an operand address,
- L is an expression specifying operand length,
- S is an expression specifying the number of bit positions that an operand is to be shifted,
- C is a six bit editing pattern, and
- X is an index register expression.

TRANSLATE – TR M, L, X

Function: Replace the L characters whose least significant character is in M_x using a translation table.

Notes:

a. The maximum value of L is 64.

b. A translation table may consist of a maximum of 64 characters stored in any row of store from 0-63. The row number must be program set in absolute location 72 (TRO).

c. The M_x expression specifies the location of the least significant character to be translated. Translation works from the least significant to the most significant character, until the number of characters specified by L have been translated.

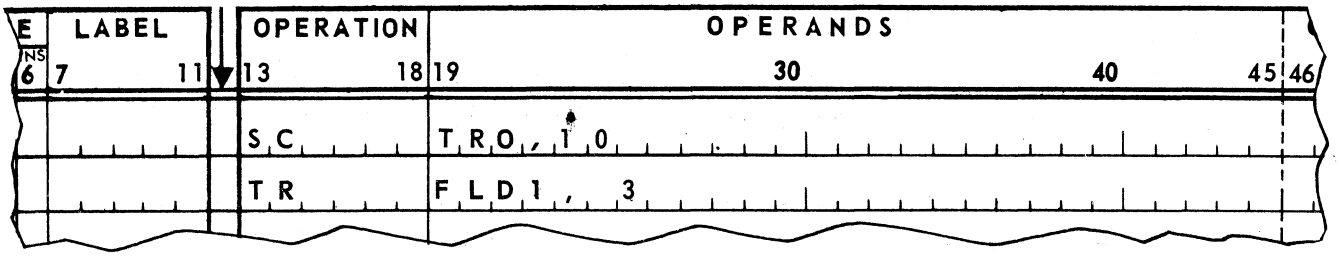
d. The TR instruction replaces each character in the field to be translated with a character selected from the row specified by TRO. The basis for selecting the replacement character is the binary value of the character to be replaced. The binary value of any six bit character ranges from zero (000000) through 63 (111111). This binary value provides the character address of the particular six bit configuration within the specified row which is to replace the character. In other words, a character with a binary value of zero (000000) is replaced by whatever character is prestored in position 0 of the translate row; a character with a binary value of 1 (000001) is replaced by whatever character is prestored in position 1 of the translate row; and so on.

e. The contents of the translate row are not altered by the instruction, unless the translate row itself is translated.

f. If L is greater than 15, L may not be implied by means of a previous definition (cf. AREA Directive).

Example:

A three character field containing the bit configurations 010101 010100 100010 is labeled FLD1. These bit configurations are the 90 column card codes for the characters A B C. FLD1 is to be printed and must be translated from 90 column card code to UNIVAC 1050 XS 3 code. The translation table is in Row 10 (locations 640-703). The first instruction places the row number in TRO.



FLD1 (before) = 010101 010100 100010

90 column equivalent = A B C

Decimal value = 21 20 34

Character position = 20 21 34
Row 10 = . . . (010101) (010100) . . . (010110) . . .

1050 equivalent = B A C

FLD1 (after) = 010100 010101 010110

1050 equivalent = A B C

3.6.2. EDIT

Format: **ED** M, L, X

Function: Edit the L least significant characters of arithmetic register 1 into the store positions whose least significant character is M_x under control of the pattern in AR2.

Notes:

a. The maximum value of L plus E is 16.*

b. The edit instruction facilitates the following operations on a data field

- elimination of the sign bit
- translation of the sign to a form suitable for printing
- insertion of punctuation (any alphanumeric character except @) in the data field:

c. Data in AR1 is placed in the designated storage, in character positions that correspond to the location of the @ and ⌑ characters in AR2.

d. If the least significant character in AR2 is

– a minus sign (000010), two functions will be performed:

- if the field in AR1 is negative, a minus sign is placed in M_x; if the field is positive, a blank is placed in M_x.
- the least significant character in AR1 is transferred to M_x-1 and binary, zeros are placed in the zone bits.

⌑ a lozenge (111111), the numeric bits of the least significant character of AR1 are copied into location M_x, and binary zeros are placed in the zone bits of M_x.

@ an "at" character (100000), the least significant character of AR1 is copied into M_x without alteration.

Any other character appearing in the least significant character position of AR2 is transferred to location M_x unaltered.

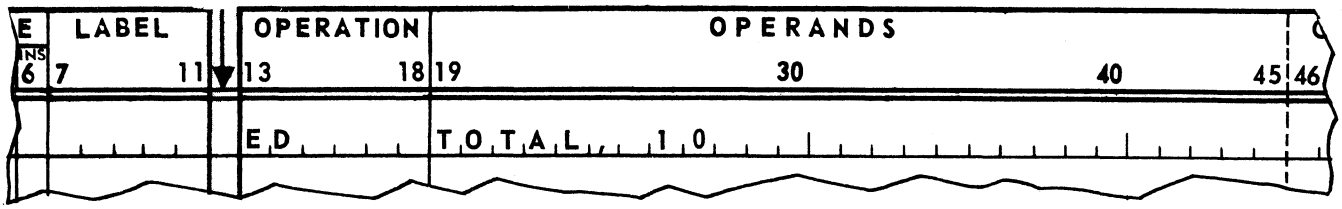
e. Except as noted in note d, any character in AR 2, other than an @, is transferred unaltered to a corresponding position in the designated storage area; an @ causes the corresponding character in AR1 to be transferred to the designated storage area unaltered.

f. The number of @ characters in AR2 must be at least equal to the number of characters specified by L.

** E = number of characters inserted into the edited field.*

Examples:

- Edit the 10 least significant characters of AR1 according to the pattern contained in AR2, placing the edited field in the locations whose least significant character is labeled TOTAL.



AR1 =	0	0	0	0	0	0	1	2	3	4	5	6	7	8	9	+0
AR2 =		@	@	@	,	@	@	@	,	@	@	@	.	@	@	−

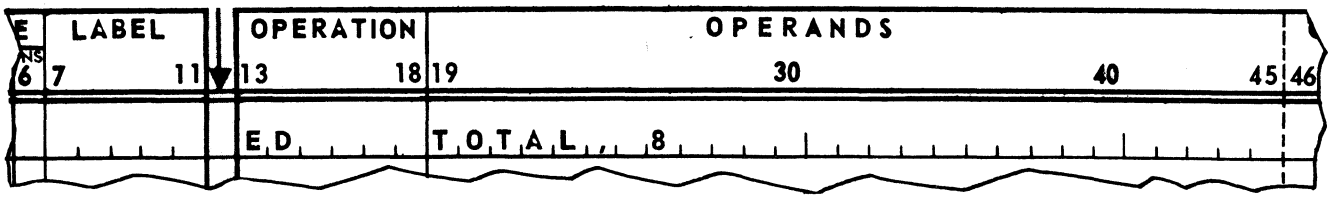
After the instruction is executed, the field labeled TOTAL contains

												TOTAL
X	1	2	,	3	4	5	,	6	7	8	.	9	0	

Character position TOTAL is blank because the field is positive. If it were negative, TOTAL would contain

												TOTAL
X	1	2	,	3	4	5	,	6	7	8	.	9	0	−

- Edit the 8 least significant characters of AR1 according to the pattern in AR2 and store the edited field in the field labeled TOTAL.



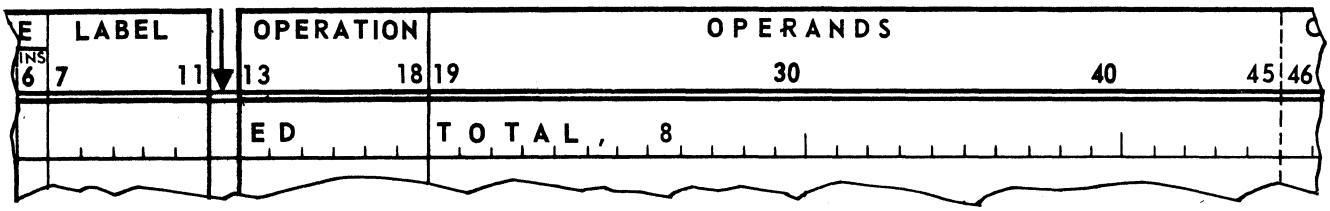
AR1 =	0	0	0	0	0	0	0	0	3	4	5	6	7	8	9	−0
AR2 =	0	0	0	@	@	,	@	@	@	,	@	@	@	.	@	@

After the instruction is executed, TOTAL will contain

											TOTAL
X	X	X	3	4	5	,	6	7	8	.	9	!

because the minus sign appears in the zone bits of the least significant character in AR1, the least significant @ character in AR2 acts as a ¤ and the zone bits are not transferred. (If the zone bits were to be transferred the result would be the 1050 character for the exclamation point.)

- Edit the 8 least significant characters of AR1 according to the pattern in AR2 and store the edited field in the field labeled TOTAL.



AR1 =	0	0	0	0	0	0	0	0	3	4	5	6	7	8	9	+0
AR2 =	@	@	@	,	@	@	@	.	@	@	*	T	O	T	A	L

After the instruction is executed, TOTAL will contain

3	4	5	,	6	7	8	.	9	0	*	T	O	T	A	L

ZERO SUPPRESS – ZS M, L, X
ZS$ M, L, X
ZS* M, L, X

Function: Beginning at location M_X and working to the right on a maximum of L characters, replace blanks, zeros, and commas until a character which is neither a blank, a zero, nor a comma is encountered.

Notes:

a. In this instruction, M_X specifies the most significant character position of the field, as the instruction operates on the field from left to right.

b. The maximum value of L is 16.

c. A ZS instruction replaces all leading blanks, zeros, and commas with blanks.

d. A ZS$ instruction replaces all leading blanks, zeros, and commas with blanks, and inserts a dollar sign ($) in the position immediately to the left of the first character encountered which is neither a blank, a zero, nor a comma.

e. A ZS* instruction replaces all leading blanks, zeros, and commas with asterisks (*).

f. A count (from 0 to 16), expressed in binary, of the number of characters suppressed is stored by the circuitry in ZCT (absolute location 73).

g. If M_X is not a blank, a zero, or a comma, and a ZS$ instruction is executed, a dollar sign is inserted into M_X-1.

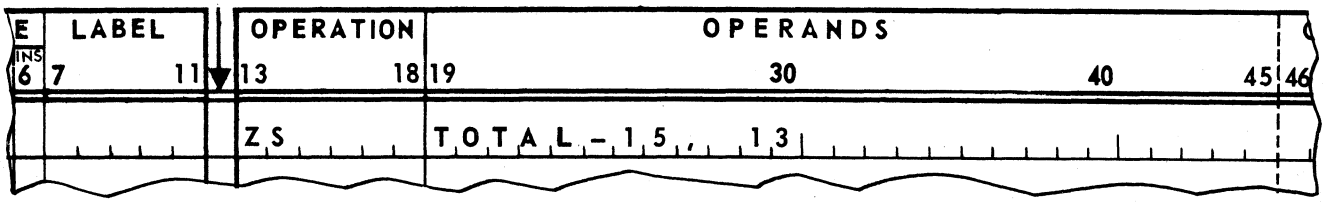
Example:

Suppress leading zeros, commas, and blanks in the field whose most significant character is labeled TOTAL-15.

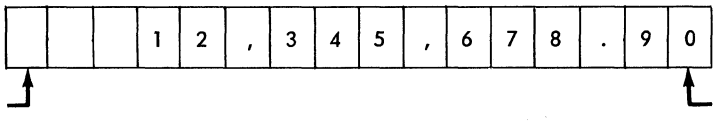
TOTAL-15 through TOTAL =

0	,	0	1	2	,	3	4	5	,	6	7	8	.	9	0

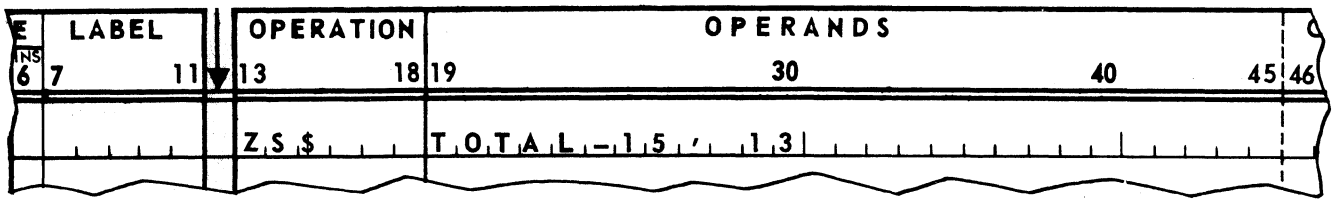
If the instruction is



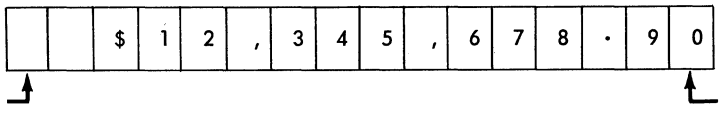
after the instruction is executed, the field TOTAL-15 through TOTAL would contain



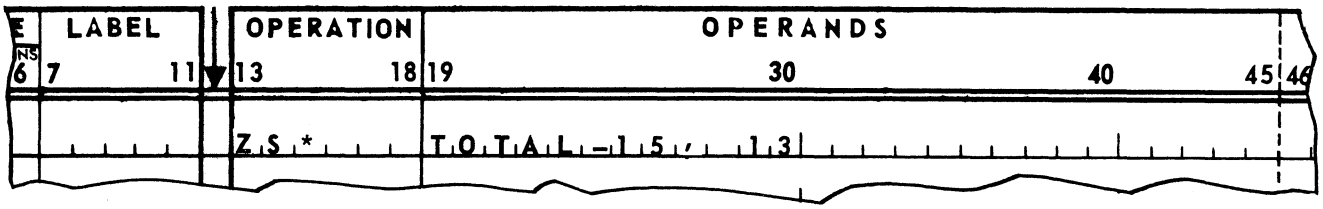
If the instruction is



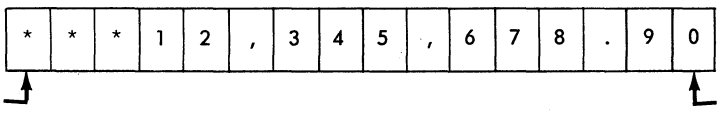
the field would contain



If the instruction is



the field would contain



In all three cases, ZCT would contain a 3 expressed in binary (000011) .

3.6.4. PAD BLANKS

Format: **PD** M, L, X

3.6.5. PAD ZEROS

Format: **PDO** M, L, X

Function: Place decimal zeros (000011) or blanks (000000), as indicated by the operation code, in the L locations whose least significant character is at M_X.

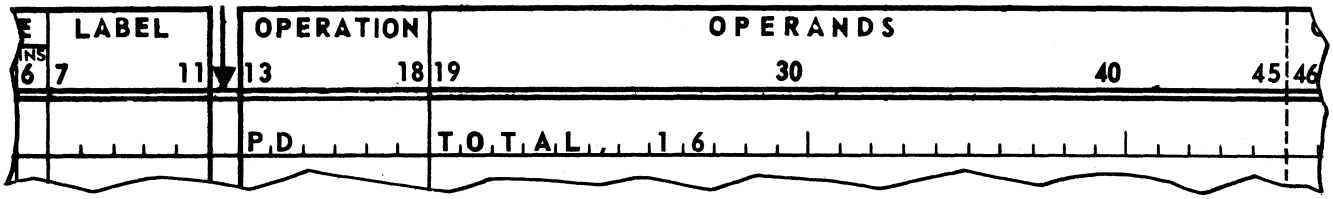
Notes:

a. The arithmetic registers are not involved in the operation of this instruction, unless M_X is the address of a location in either arithmetic register.

b. The maximum value of L is 16.

Example:

Place blanks in the 16 character field whose least significant character position is labeled TOTAL.



TOTAL (before) =	*	*	*	1	2	,	3	4	5	,	6	7	8	.	9	0
TOTAL (after) =																

LOGICAL SUM – LS M, C, X

Function: For every bit position in C containing a one, place a one in the corresponding bit position in M_X.

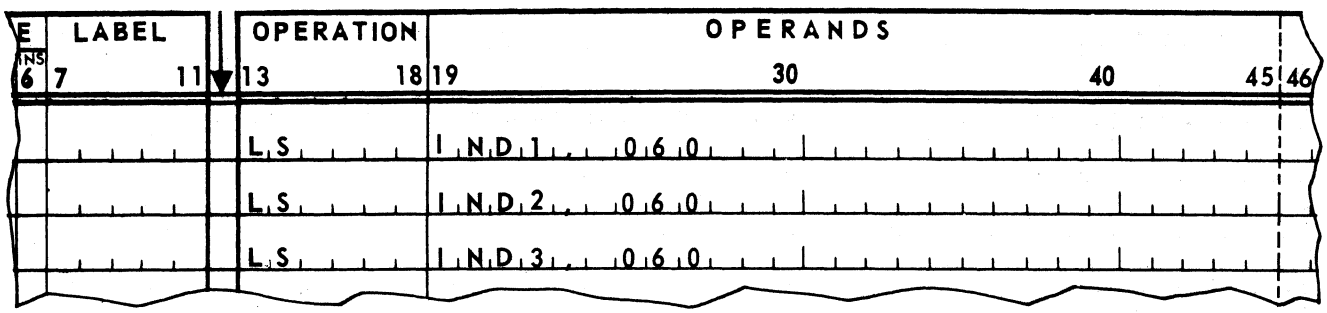
Notes:

a. The bit positions of M_X which correspond to those bit positions of C containing 0 bits are unchanged.

b. C is not altered after the instruction is executed.

Example:

Superimpose the character 110000 on the characters IND1, IND2, and IND3.



IND1 (before) = 001111
IND1 (after) = 111111

IND2 (before) = 000011
IND2 (after) = 110011

IND3 (before) = 100011
IND3 (after) = 110011

LOGICAL PRODUCT - LP M, C, X

Function: For every bit position in C containing a zero, place a zero in the corresponding bit position in M_x.

Notes:

a. The bit positions of M_x which correspond to those bit positions of C containing 1 bits are unchanged.

b. C is not altered after the instruction is executed.

Example:

Extract the three least significant bit positions of the characters IND4, IND5, and IND6. (C must be 111000.)

E INS	LABEL	OPERATION	OPERANDS
6	7 11	13 18	19 30 40 45 46
		LP	IND4 070
		LP	IND5 070
		LP	IND6 070

IND4 (before) = 111111
IND4 (after) = 111000

IND5 (before) = 101101
IND5 (after) = 101000

IND6 (before) = 001011
IND6 (after) = 001000

3.6.8. BIT SHIFT

Format: **BSn** M, S, X

Function: Shift the n characters whose least significant location is M_x, S bit positions left, replacing the S least significant bit positions of M_x with binary zeros. Note that this instruction is a bit shift involving an integral number of characters.

Notes:

a. n may be 1, 2, 3, or 4, specifying the number of six bit characters involved in the shift.

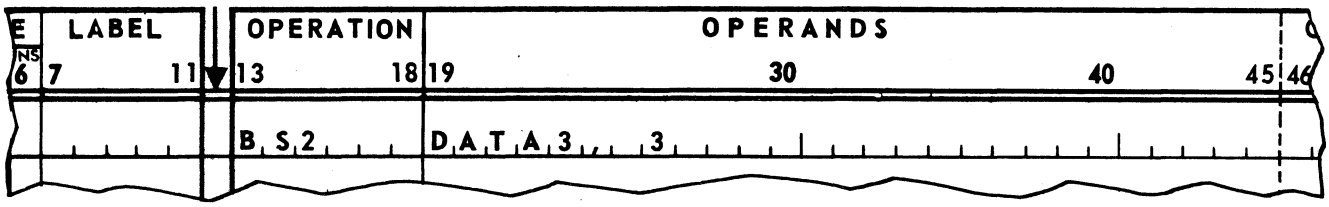
b. S specifies the number of bit positions that the field is to be shifted left. A maximum shift of 7 is possible.

c. Bits shifted beyond the most significant bit position of the most signficant character are lost.

d. Zeros replace the bits shifted out of the least significant bit positions of the least significant character(s).

Example:

Shift the two character field, DATA3, 3 bit positions left.



DATA3-1 through DATA3 (before) = 110101 001111
DATA3-1 through DATA3 (after) = 101001 111000

3.6.9. BIT CIRCULATE

Format: **BCn** M, S, X

Function: Shift the n characters whose least significant location is M_X, S bit positions left. The S most significant bits are moved into the S least significant bit positions of M_X. Note that this instruction is a bit shift involving an integral number of characters.

Notes:

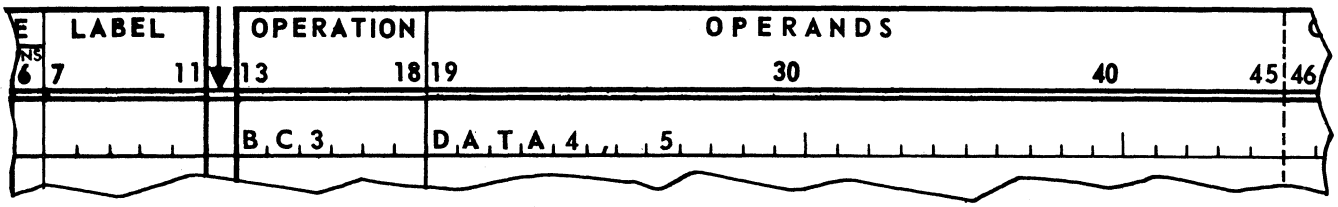
a. n may be 1, 2, 3, or 4, specifying the number of six bit characters involved in the shift.

b. S specifies the number of bit positions that the field is to be shifted left. A maximum shift of 7 is possible.

c. Bits shifted beyond the most significant bit position of the most significant character are entered into the least significant bit positions of the least significant character(s).

Example:

Shift the three character field, DATA4, 5 bit positions left, circularly.



DATA4-2 through DATA4 (before) = 100110 110101 001111
DATA4-2 through DATA4 (after) = 011010 100111 110011

PAGE NO'S.	TYPE	MNEM CODE	OPERANDS	INSTRUCTION	APPROXIMATE INSTRUCTION EXECUTION TIMES IN MICROSECONDS
3-14	DATA TRANSFER	BDa	M, L, X	BRING DECIMAL	36 + 9L
-16		BAa	M, L, X	BRING ALPHANUMERIC	27 + 9L
-6		BT	M, T, X	BRING TETRAD	63
-17		SAa	M, L, X	STORE ARITHMETIC REG.	27 + 9L
-17		SAR	M , , X	STORE BOTH ARITHMETIC REGISTERS	324
-6		ST	M, T, X	STORE TETRAD	63
-18		SC	M, C, X	STORE CHARACTER	40.5
-10		FT	M, T, X	FIX TETRAD	81
-20		TFI	M , , X‡	TRANSFER BLOCK FROM STORE, INCREMENT	103.5 + 9B
-20		TFR	M , , X‡	TRANSFER BLOCK FROM STORE, RESET	90 + 9B
-22		TTI	M , , X‡	TRANSFER BLOCK TO STORE, INCREMENT	103.5 + 9B
-22		TTR	M , , X‡	TRANSFER BLOCK TO STORE, RESET	90 + 9B
3-26	ARITHMETIC	ADa	M, L, X	ADD DECIMAL	49.5 + 13.5 (L + Lc); if $\|M_X\| > \|ARa\|$ and signs ≠, 49.5 + 27L
-38		ABa	M, L, X	ADD BINARY	27 + 13.5L
-29		AMa	M, L, X	ADD TO MEMORY	49.5 + 13.5; if $\|ARa\| > \|Mx\|$ and signs ≠, 49.5 + 31L
-7		AT	M, T, X	ADD TO TETRAD	81
-40		AC	M, C, X	ADD CHARACTER	45 + 13.5Lc
-28		SDa	M, L, X	SUBTRACT DECIMAL	49.5 + 13.5 (L + Lc); if $\|M_X\| > \|ARa\|$ and signs =, 49.5 + 27L
-39		SBa	M, L, X	SUBTRACT BINARY	27 + 13.5L
-31		SMa	M, L, X	SUBTRACT FROM MEMORY	49.5 + 13.5L; if $\|ARa\| > \|Mx\|$ and signs =, 49.5 + 31L
-32		MPN	, L	MULTIPLY NON-CUMULAT.	L (33.75K + 63.5) + 99
-34		MPC	, L	MULTIPLY CUMULATIVE	L (33.75K + 63.5) + 27
-36		DV	, L	DIVIDE	L (74.25K + 128.25) + 13.5K + 49.5
3-44	COMPARISON	CDa	M, L, X	COMPARE DECIMAL	36 + 13.5L'; if ≠, 36
-46		CBa	M, L, X	COMPARE BINARY	27 + 13.5L
-47		CC	M, C, X	COMPARE CHARACTER	40.5
-9		CT	M, T, X	COMPARE TETRAD	81
-48		LC	M, C, X	LOGICAL COMPARE	40.5
3-51	SEQUENCE CONTROL	JE	M, X	JUMP EQUAL	31.5
-51		JG	M, X	JUMP GREATER	31.5
-51		JS	M, X	JUMP SMALLER	31.5
-51		JU	M, X	JUMP UNEQUAL	31.5
-51		J	M, X	JUMP	31.5
-54		JC	M, I, X	JUMP CONDITIONAL	31.5
-59		JL	M, N, X	JUMP LOOP	40.5
-56		JR	M, I, X	JUMP RETURN	45
-53		JD	M, X	JUMP DISPLAY	31.5
-53		JHJ	M, X	HALT, THEN JUMP	31.5
3-72	SHIFT	BCn	M, S, X	BIT CIRCULATE	40.5 + S (9 + 18n)
-71		BSn	M, S, X	BIT SHIFT	40.5 + S (9 + 18n)
3-69	EDIT	LS	M, C, X	LOGICAL SUM	40.5
-70		LP	M, C, X	LOGICAL PRODUCT	40.5
-68		PD	M, L, X	PAD BLANKS	27 + 4.5L
-68		PD0	M, L, X	PAD ZEROS	27 + 4.5L
-66		ZS	M, L, X‡	ZERO SUPPRESS	45 + 9Z
-66		ZS$	M, L, X‡	ZERO SUPPRESS AND FLOATING $ SIGN	49.5 + 9Z
-66		ZS*	M, L, X‡	ZERO SUPPRESS WITH ASTERISK FILL	45 + 9Z
-63		ED	M, L, X	EDIT	36 + 13.5L + 9E
-61		TR	M, L, X	TRANSLATE	36 + 13.5L

XF INSTRUCTION TIME IS 72 MICROSECONDS.
B = NUMBER OF CHARACTERS TRANSFERRED
E = NUMBER OF CHARACTERS INSERTED INTO EDITED FIELD
K = DIVISOR OR MULTIPLICAND LENGTH
L = OPERAND LENGTH OR LENGTH OF QUOTIENT
L = LENGTH OF THE LONGER OF TWO FIELDS
Lc = CARRIES BEYOND L^{TH} DIGIT
$|M_X|$ = ABSOLUTE VALUE OF M_X
N = NUMBER OF CHARACTERS SHIFTED
S = BIT POSITIONS SHIFTED
Z = NUMBER OF CHARACTERS SUPPRESSED

Table 3-4. Instruction Execution Times

4. AUTOMATIC PROGRAM INTERRUPT

4.1. General Description

Automatic program interrupt is a concept incorporated into the control circuitry of the UNIVAC 1050 System which enables the system to operate at optimum overall efficiency. The automatic program interrupt feature permits the efficient utilization of all input/output devices operating under control of the Central Processor without sacrificing any processing time within the program cycle – an essential consideration in the maintenance of maximum input/output speeds.

Basically, automatic program interrupt consists of the generation of a signal to the Central Processor upon the recognition of a condition that requires immediate attention from the program. These interrupt signals are assigned a priority within a hierarchy of interrupts in order to facilitate their processing.

Associated with automatic program interrupt is interrupt inhibit, which prevents the acceptance of an interrupt signal when it is generated. However, the interrupt signal is stored in an indicator that can be tested subsequently by a program instruction.

Interrupt results from one of two general classes of occurrences: first, an error, fault, or emergency condition occurring either in the Central Processor or in an input/output device; and, second, successful completion of an input/output function or, in some cases, when an input/output device is ready to accept an input/output command.

Upon the occurrence of an interrupt, and if interrupt has not been inhibited, control is transferred to one of ten fixed store locations which must contain the starting address of a routine that processes the interrupt.

Programs that use the PAL Assembler library of input/output routines supplied by UNIVAC are relieved from the burden of controlling and coordinating interrupts since comprehensive interrupt coding is included in these routines.

For the benefit of the programmer who wishes to write his own input/output and interrupt coordinating routines, the following subsection presents the considerations attendant upon interrupt programming.

4.2. Programming Considerations

4.2.1. Classes of Interrupt

There are three classes of interrupt which are named in the order of their priority: Class I, Class II, and Class III.

When a Class I interrupt occurs, a Class I Interrupt Inhibit bit is set automatically. While this bit is set, the processing (but not the storage) of all subsequent interrupts is prohibited. If a Class I interrupt occurs while the Class I Interupt Inhibit bit is set, the Central Processor stalls.

When a Class II interrupt occurs, a Class II Interrupt Inhibit bit is set automatically. While this bit is set, the processing (but not the storage) of subsequent Class II and Class III interrupts is prohibited. A Class I interrupt, however, will be processed in spite of the inhibition of Class II interrupts.

When a Class III interrupt occurs, a Class III Interrupt Inhibit bit is set automatically. While this bit is set, the processing (but not the storage) of subsequent Class III interrupts is prohibited. Class I and Class II interrupts, however, will be processed in spite of the inhibition of Class III interrupts.

4.2.1.1. Class I Interrupt

A Class I interrupt occurs upon the recognition of a main store parity error when the control circuits of the Central Processor obtain and execute instructions. Such an error is known as an internal parity error. Parity errors occurring while input/output devices are accessing main store are excluded from this definition.

4.2.1.2. Class II Interrupt

A Class II interrupt is caused by either

- decimal overflow or improper division, both of which set the Decimal Overflow Indicator (Indicator 40), or
- the depression of the Operator Request Switch on the console, which sets the Operator Interrupt Indicator (Indicator 44).

4.2.1.3. Class III Interrupt

A Class III interrupt is generated by the Synchronizers associated with the input/output devices of the UNIVAC 1050 System upon the occurrence of any of the following:

- Successful completion of an input/output function, which may result from
  - the normal termination of a requested input/output function without detected errors, or
  - an interrupt request from a demand device without detected errors. A demand device is one that is expected to generate an interrupt request at fixed time intervals whether or not an instruction has been issued to it.
- Error conditions when
  - normal termination of a requested input/output function is accompanied by the detection of an error or errors; or
  - an error occurs while an input/output function is in progress which will prevent normal termination.
- Off normal conditions resulting from
  - the issuance of an input/output instruction to a device that has not completed a previously requested operation;
  - the detection of an error or fault condition in a device that is not in use; or
  - the existence of a condition whereby the acceptance of the instruction would violate the rules governing the simultaneous use of input/output channels – a condition known as Storage Overload. The purpose of these rules is to prevent the occurrence of an input/output data transfer rate that exceeds the main store data transfer rate.

When an instruction requests an off normal device, an interrupt request is generated and the instruction is disregarded.

4.2.2. Programmed Interrupt Inhibit

Class II and Class III interrupts may be inhibited by program instruction. The following rules govern programmed interrupt inhibit.

- Operator Interrupt may be inhibited by instruction. Such inhibit can only be released by instruction or by the depression of the CLEAR button on the console.
- Decimal Overflow Interrupt may be inhibited by instruction. Such inhibit can only be released by instruction.
- Setting of a programmed Decimal Overflow Interrupt Inhibit sets an indicator (Indicator 47), which may be tested by a program instruction.
- Class III Interrupt from all input/output channels may be inhibited by instruction. Such inhibit can only be released by program instruction. Setting the Class III Interrupt Inhibit sets Indicator 45, which may be tested by a program instruction. This inhibits all Class III Interrupts.

This general Class III interrupt inhibit is distinct from the channel interrupt inhibit specifiable in an XF instruction which inhibits further interrupts only from the specified channel and which is released by a subsequent XF instruction to that channel.*

The setting and resetting of programmed interrupt inhibit does not affect, nor is affected by, any other class of interrupt.

4.2.3. Instructions Associated with Interrupt Control

The Jump Conditional and Jump Return instructions are used to control the processing of interrupts of all classes. Table 4-1 lists all the indicators; those associated with interrupts and used by the Jump Conditional and Jump Return instructions are marked with a dagger.

4.2.4. Fixed Interrupt Locations

Associated with each class of interrupt, and with each input/output channel on the UNIVAC 1050 System, is a group of eight consecutive character positions through which communication with the interrupt routines is maintained. The foldout Figure 3–1 on page 3–2 shows the location of these fixed interrupt addresses.

** The XF instruction will be fully explained in the applicable 1050 peripheral subsystem manual.*

The indicators in the table below are divided into two groups: testable and nontestable. The nontestable indicators (00-31) cause a certain function to be performed and an unconditional jump. The conditional jump indicators (32-63) are tested and cause a jump only if the indicator has been set.

00-31 Unconditional Jump to M Address

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 and jump
†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 and jump
†*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

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 — These four indicators are affected by the comparison instructions: CC, LC, CD, CB, CT.
34 (KEQ)	Equal
35 (KUQ)	Unequal
36 (KLO)	Low
37 (KZR)	Result of last arithmetic operation was zero
38 (KM)	Result of last decimal arithmetic operation was negative.
39 (KNB)	No overflow in last add binary operation or overflow did occur in the last binary subtract 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)
46	Test and reset inquiry typewriter request
† 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

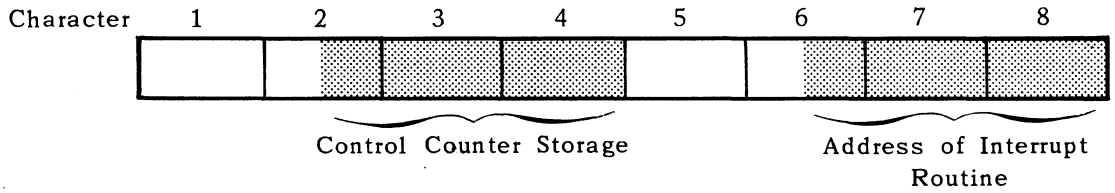
*RESETS the inhibit automatically generated when the interrupt occurred.

†See Section 4.2.3.

§Also inquiry typewriter, if preset.

Table 4-1. Indicator List

The format of each eight character group is as follows:



When an interrupt occurs, the contents of the control counter are stored in the 15 least significant bits of characters 2, 3, and 4; zeros are placed in the three most significant bit positions of character 2. The 15 least significant bits of characters 6, 7, and 8 are then read into the control counter. These characters should contain the starting address of the interrupt routine associated with the particular fixed interrupt locations.

The following sequence of events takes place when an interrupt request is accepted:

a. The instruction currently being executed by the Central Processor is completed. Exception: When a Class I interrupt occurs, an ending pulse is immediately generated for the instruction currently being executed.

b. The address of the next instruction in the program being interrupted (contents of the control counter) is stored in characters 2, 3, and 4 of the fixed locations associated with the channel which initiated the request.

c. The 15 least significant bits of characters 6, 7, and 8 (starting address of the interrupt routine) are read into the control counter. In addition, a signal is generated which prevents the Central Processor from accepting additional interrupt requests from channels of the same or lower classes. This signal persists until an Automatic Interrupt Inhibit Release instruction for this class is executed.

d. The Central Processor does not recognize additional interrupt requests of any kind during the time required to execute steps 2 and 3. Beginning with the completion of step 3, interrupt requests from higher classes will be accepted.

Control is returned to the interrupted program by means of the address stored in characters 2, 3, and 4. Character 1 should contain the operation code for a JC instruction, and character 5 should contain the indicator releasing the automatic class interrupt inhibit for the class with which this channel is associated.

With the exception of characters 2, 3, and 4 of the area, the area must be preset by the initializing subroutine of each program.

It should be noted that Class I and Class II interrupts each have a single fixed interrupt area. It is a function of the Class II interrupt routine to determine whether the interrupt was caused by decimal overflow or by an operator interrupt request.

Class III interrupts have eight fixed interrupt areas, one for each input/output channel. Fixed priorities within the class are assigned to each channel to avoid conflict by simultaneous interrupt requests. Once an interrupt request has been accepted, however, it cannot be interrupted by another request from a channel of higher priority within the same class until an instruction releasing the Automatic Class III Interrupt Inhibit has been executed.

5. CENTRAL PROCESSOR CONSOLE OPERATION

The Central Processor Console provides a communication link between the Central Processor and the operator. The console contains display indicators that allow the operator to determine normal and abnormal conditions, and to access registers and selected positions in main storage. In addition, the console contains switches that allow the operator to correct or override error conditions, debug programs online, and manually set sense switches for program use.

5.1. NORMAL OPERATION

5.1.1. 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.

5.1.2. 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. 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 condition.

5.1.3. Operating Mode

The six mutually exclusive mode control switches are used to control the operation of the processor in conjunction with the PROGRAM START button.

5.2. PANEL CONTROLS AND INDICATORS

a. Switch/Indicators

All Switch/Indicators Are Momentary Action Switches.

DESIGNATION	DESCRIPTION
SYSTEM ON *(Green)*	Turns system on. Lights when depressed; extinguishes when SYSTEM OFF indicator is depressed.
SYSTEM OFF *(Red)*	Turns system off. Lights when depressed; extinguishes when SYSTEM ON switch is depressed.
CLEAR *(White)*	General clear of testable indicators, counters, and stored interrupts; and initiates a console lamp test. Lights when depressed; extinguishes when released.
PROC ABNORMAL *(Red)*	Lights when a second parity error is recognized and a previous parity error has not been cleared. Depressing this button when illuminated turns it off, but does not clear the parity error. May also be on due to maintenance operations.
PARITY *(Yellow)*	Indicator only. Lights when a parity error is detected in a character read from storage. Parity errors cause an immediate processor halt. (See Section 5.3.4 on how to clear a parity error).
CLASS III INHIBITED *(Yellow)*	Indicator only. Lights when a Class III Interrupt Inhibit is stored; extinguishes when the inhibit is released by the program.
OVERFLOW INHIBITED *(Yellow)*	Indicator only. Lights when a Decimal Overflow Interrupt Inhibit is set. Extinguishes when the inhibit is cleared by the program.
CHANNEL ABNORMAL 0–7 *(8 Red)*	Lights when fault develops in corresponding I/O channel. Extinguish by clearing error at peripheral unit and then depressing switch.
SENSE 1 2 3 *(3 White)*	Lights when depressed; extinguishes when depressed again. Used in conjunction with programmed tests and program operating instructions.
OPERATOR REQUEST *(White)*	Lights under program control. Can be depressed when illuminated to stop program by causing a Class II interrupt. Depressing the OPERATOR REQUEST button when extinguished has no effect.

Table 5-1. Control Console Switch and Indicator Description

DESIGNATION	DESCRIPTION
NEXT INSTRUCTION M *(White)*	Directs processor to M portion of the current instruction. Lights when depressed or under program control; control is transferred to the M portion of the current instruction. Extinguishes when CC switch is depressed or under program control.
CC *(White)*	Forces the processor to obtain the next instruction from the address in the control counter; that is, it overrides a jump instruction. Lights when depressed or under program control. Extinguishes when M switch is depressed or under program control. Either the M or the CC button will always be illuminated. *Note: During operation in the continuous mode, the Next Instruction switch/indicator will light according to the instruction sequence and depending on the result of programmed tests and comparisons. Consequently, these buttons should not be depressed during continuous operations. Doing so could force an incorrect instruction sequence.*
PROGRAM START *(Green)*	Lights when depressed. Depressing when extinguished initiates execution of program under control of Mode switches. This indicator may flash on and off rapidly during the execution of a long instruction, but this should be ignored unless light stays off for longer than a few seconds.
PROGRAM STOP *(Red)*	Lights when depressed or when program stops. Extinguishes when PROGRAM START switch is depressed. When both PROGRAM START and PROGRAM STOP buttons extinguish and will not illuminate, processor is stalled. Depress ONE INST Mode and PROGRAM START buttons.
DISPLAY *(White)*	When depressed causes the contents of the storage location represented in the M portion of the Display/Alter switches to be displayed in lights in the C portion. Lights when depressed. Extinguishes when released.
ALTER *(White)*	When depressed causes the character represented by the setting of the C alter switches to be stored into the address specified in the M portion of the Display/Alter switches. Lights when depressed. Extinguishes when released.

Table 5-1. Control Console Switch and Indicator Descriptions (continued)

b. Illuminating Pushbuttons

Depress To Set; Depress To Release; Lights When Depressed.

DESIGNATION	DESCRIPTION
MODE LOAD CARD	The MODE pushbuttons operate one at a time. One of these buttons must be illuminated at all times. When depressed and illuminated, depressing the PROGRAM START button causes one card to be read from the reader into octal location 400. Depressing PROGRAM START again, with the CONTinuous mode button depressed, causes control to be transferred to octal location 400, when UNIVAC standard code cards are used.
LOAD TAPE	When depressed and illuminated, depressing the PROGRAM START button will cause one block of tape to be read from logical tape unit 0 into storage starting at octal location 400. Changing to the continuous mode and depressing PROGRAM START immediately after, will transfer control to this location.
ONE CYCLE	When depressed and illuminated, depressing the PROGRAM START button will cause instructions to be executed one instruction cycle at a time. This mode is generally used by UNIVAC Field Engineering personnel only.
ONE INSTR.	When depressed and illuminated, depressing the PROGRAM START button will cause one instruction to be executed and the next instruction accessed.
CONT	When set and the PROGRAM START switch is depressed, a program will run in the normally used continuous mode.
DISPLAY/ALTER SELECTION Q1 Q2	In order to display or alter the contents of storage the Display/Alter Selection buttons are used in conjunction with the display lights and alter switches. The functions of the Display/Alter Selection buttons are as follows: When set, these buttons display internal registers and indicators on the control console. They are primarily for UNIVAC Field Engineering use.
CC	When set, the contents of the control counter are displayed in the M portion of the display lights.
INST.	When set, the contents of the instruction register (the next instruction to be executed) are displayed in the 30 display lights.
OP/CH	When set, the entire instruction is displayed but only the operation code and the channel (index register) designation portions of the instruction register are alterable.
M	When set, the entire instruction is displayed but only the operand address (M portion) of the instruction register is alterable.
C	When set, the entire instruction register is displayed but only the C portion (detail field) of the instruction register is alterable.
MEM	When set, it causes the contents of the storage location specified by the M alter switches to be displayed when the DISPLAY switch is depressed, and altered when the ALTER switch is depressed.
SEQ	When set, it is used in conjunction with the ALTER or DISPLAY switch to display or alter the contents of sequential memory locations.

Table 5-1. Control Console Switch and Indicator Descriptions (continued)

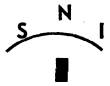
DESIGNATION	DESCRIPTION
TRACE MODE	Switches OP, CC, PROC, WRITE, and I/O operate one at a time; TRACE STOP may be used with any one of the five Trace Mode switches. If TRACE STOP is not used, a program testable indicator is set.
OP	When set with TRACE STOP, the computer will stop when the program operation code matches the settings of the five most-significant-digit alter switches (OP portion of the top row of indicators).
CC	When set with TRACE STOP, the computer will stop when the contents of the control counter match the settings of the Trace Address switches.
PROC	When set with TRACE STOP, the computer stops when an operand address matches the settings of the Trace Address switches.
WRITE	When set with TRACE STOP, the computer stops when a character is to be written into an address location which matches the settings of the Trace Address switches.
I/O	When set with TRACE STOP, the computer will stop when a control unit reference to a storage address matches the settings of the Trace Address switches.
TRACE STOP	When set, the program stops at a specified location if one of Trace Mode switches is also set.

c. Toggle Switches

ALTER SWITCHES	Not labeled as such; they are the row of 30 switches immediately below the control indicators in groups labeled OP, CH, M, and C. They are two-position toggle switches with up and center positions only: a switch in the up position represents a binary 1; in the center position, a binary 0. A binary pattern can be stored in these switches; this pattern can then be used to alter the area of the processor that is designated by the Display/Alter election pushbuttons. Alteration occurs only when the DISPLAY or ALTER switches are depressed.
TRACE ADDRESS	Sixteen three-position toggle switches which correspond to the M portion of the instruction: up represents a binary 1; down represents a binary 0; the center position is either a 1 or a 0 and will compare with both. For example, a switch pattern of up down middle will trace either 101 or 100. The trace address positions correspond to parts of the instruction and are repeatedly compared to the instruction for equality. If equality is detected and one of the Trace Mode pushbuttons is depressed, an indicator is set; if the TRACE STOP button is also depressed the program stops.

Table 5-1. Control Console Switch and Indicator Descriptions (continued)

d. Rotary Switches

DESIGNATION	DESCRIPTION
CLASS I 	Three-position rotary switch; used for manual control of Class I interrupts. Positions are: S – STALL N – NORMAL I – INHIBIT In the STALL position the processor stops each time a parity error is detected. It is restarted by depressing PROGRAM START. In the NORMAL position interrupt requests are processed through the interrupt entry channel. The processor will not stall unless another Class I interrupt occurs while in the interrupt mode. In the INHIBIT position Class I interrupts are ignored. However, it is recommended that the program stop on a Class I interrupt since recovery without operator intervention is not specified.
OF 	Operates same as above but for Class II interrupts: Overflow (OF), improper division, operator request.
CLASS III 	Operates same as above but for Class III interrupts.*

* *See Section 4 for a complete description of Class I, II, and III interrupts.*

Table 5-1. Control Console Switch and Indicator Descriptions (continued)

5.3. PROGRAM DEBUGGING AND TESTING

5.3.1. Use of Display Lights and Switches

The 30 display lights and corresponding toggle switches at the top of the console are a primary means of communication between the operator and a running program. These lights and switches must be read as octal numbers. To do this they are interpreted in groups of three binary digits. A binary digit, or bit, can have a value of either 0 or 1; in this case, an illuminated (on) display light represents a 1, while an extinguished (off) display light represents a 0. Similarly, an octal digit can have a value from 0 to 7, and any octal digit can be represented by three binary bits. The bit patterns, or groups of display lights, representing all the octal digits are as follows:

Octal Number	Bit Pattern
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

By interpreting the 30 display lights as 10 groups of three each, any display can be read as 10 octal digits.

The setting of the Display/Alter Select buttons at the middle of the console determines what will be displayed in the 30 lights. Normally, the INST Display/Alter Select button remains depressed so an entire 30-bit instruction will be displayed. Various portions of the instruction are delimited by the labels on the console between the display lights and their corresponding switches. The first five bits on the left comprise the operation code;* the next three bits are the channel number or index register used, if any; the next sixteen bits specify the storage address referred to; and the final six bits on the right comprise the detail field of the instruction.

When the computer comes to a programmed display stop, the instruction OP code will always be 30, and the detail field will always be 60 or 20; when the detail is 20 a blank is displayed. The configuration displayed in the M portion of the instruction is the "message" for that stop, and should be explained in the operating instructions for the run being executed. Display stops are usually defined only in terms of this M portion; the OP code of 30 and detail of 60 being understood. Consequently, a stop of 3007000160 would probably be written as stop 070001.

5.3.1.1. Display Contents of a Storage Location

a. Set the address of the location to be displayed in the M portion of the alter switches.

b. Depress the MEM Display/Alter Selection button.

c. Depress the DISPLAY button.

d. The contents of the selected storage location will be displayed in the six rightmost display lights (above the C notation on the console), and the address of the location displayed +1 will appear in the M display lights.

** When reading the OP code, a sixth least significant bit, which is always zero, is implied. As a result all octal OP codes are even numbers.*

If sequential storage locations are to be displayed, the above procedure should be followed to display the first character. Then, to display subsequent locations

(1) Depress the SEQ Display/Alter Selection button.

(2) Depress the DISPLAY button to display the next location.

Every time the DISPLAY button is now depressed, the M address will be increased by one, and the contents of Location M-1 will be displayed in the C lights.

5.3.1.2. Altering the Contents of Storage

a. Depress the MEM Display/Alter Selection button.

b. Set the address of the memory location to be altered in the M alter switches.

c. Set the bit configuration of the character to be stored into this location in the six C alter switches.

d. Depress the ALTER button.

The procedure for altering sequential locations is analogous to that for displaying them

a. Alter the first location as outlined above.

b. Depress the SEQ Display/Alter Selection button.

c. Set the new bit configuration to be stored in the next character location in the six C alter switches.

d. Depress the ALTER button.

e. Steps 3 and 4 should be repeated until all sequential character locations have been altered.

5.3.1.3. Altering the Next Instruction

To alter the instruction register (as displayed in the lights) the procedure is the same as that for altering the contents of storage except that the INST, OP/CH, M, or C Display/Alter Selection buttons may be used in place of the MEM and SEQ buttons.

5.3.1.4. Manual Instruction Execution

Although the CONTinuous Mode is the normal operating mode, during program testing it may occasionally be more desirable to execute one instruction at a time in order to follow the exact path taken in a particular phase of processing. This is accomplished by using the ONE INSTruction Mode button. When operating in this mode, the INST Display/Alter Selection button usually remains depressed, but the CC button may also be used to display the contents of the control counter, thereby determining the location within the program of the instruction following the one about to be executed.

With the INST Display/Alter Select button depressed, operation in the one instruction mode will cause a single instruction to be executed each time the PROGRAM START button is depressed. The instruction displayed is always the next instruction to be executed; a subsequent push of the PROGRAM START button will cause this instruction to be executed and display the next one in the 30 display lights. If the CC Display/Alter Select button is depressed, the address of the next instruction in sequence, following the one displayed by pushing the INST button, will appear in the 16 lights of the M portion of the display. Note that the instruction specified by the address in the control counter is not always the next instruction to be executed. A jump or conditional jump instruction may cause a different path to be taken.

5.3.1.5. Next Instruction Switches

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, will force a Jump Conditional instruction into the instruction register. The following parts of the instruction word are affected:

a. The operation code is staticized to Jump Conditional.

b. The C portion is changed to a value that initiates the unconditional skip associated with the Jump Conditional operation code.

When the processor is restarted, the new instruction will be performed. In this manner, any instruction may be skipped.

If a Jump Conditional or Jump Return instruction is staticized, and the processor is stopped, depressing the M button will force the processor to take its next instruction from the address in the M portion. In this manner, any jump instruction may be forced to follow the M path.

Thus, if the M Next Instruction button is illuminated when a Jump Conditional instruction is being displayed, the condition tested for has been met. The next instruction to be executed is at the address specified in the M portion of the display lights. If the CC Next Instruction button is illuminated, the next instruction to be executed is the one following the jump instruction. Its address may be displayed by pushing the CC Display/Alter Select button.

5.3.1.6. Altering Instruction Sequence

The sequence of program instructions may be altered by using the Next Instruction switch/indicators. If a programmed comparison has been made, and the M button light is lit along with the Jump Conditional instruction display, it may be desirable to see what would happen if the program would take the other path. This may be done by depressing the CC button to illuminate it, and depressing the PROGRAM START button to execute the next instruction (which is now the one specified by the control counter). This procedure does not in any way alter the contents of storage; the next time these instructions are executed, they will be unchanged. Any instruction (with the exception of Jump Loop, in which the control counter is only incremented by four instead of five) may be executed manually while the computer is in the one instruction mode. The following procedure must be followed:

a. Depress the CC Display/Alter Selection button to obtain the value of the control counter, if this value must be recorded for later use.

b. Depress the INST Display/Alter Select button.

c. Set the bit configuration of the instruction to be executed in the 30 alter switches.

d. Depress the ALTER button. The new instruction will be displayed.

e. Depress the PROGRAM START button.

f. The computer will execute the new instruction and stop, displaying the next instruction to be executed. The new contents of the control counter may be displayed by depressing the CC Display/Alter Selection button. Storage has not been altered; the next time this sequence of instructions is to be executed, the original instructions will be performed.

5.3.1.7. Tracing

Frequently during program execution, and especially during program testing, it is desirable to search (Trace) through the running program for a particular instruction, location or operation. This is accomplished through the Trace Mode buttons and Trace Address switches. The Trace Mode buttons are used to specify the type of trace being performed. The TRACE STOP button must be depressed in order to stop the computer if the traced value sought is found. However, whether or not the TRACE STOP button is depressed, a program-testable indicator is set when the trace conditions are met. If the trace is on a particular address, the value of that address must be set in the Trace Address toggle switches. These are three-position switches in which the down position indicates a 0, the up position indicates a 1, and the middle position can stand for either one. This latter feature enables tracing on several addresses at once. The five Trace Mode buttons and their uses are described in section b. of Table 5-1.

5.3.2. Error Indicators

There are two types of Central Processor errors that will cause an interrupt: a Class I interrupt, which is a parity error in a character read from storage, and a Class II interrupt (decimal overflow), caused by improper division or too great a carry in decimal addition.

A Class III interrupt is a normal interrupt of central processor operation that allows for the completion of input/output functions in the peripheral units. This class of interrupt, as well as Class II decimal overflow interrupt, may be permitted or inhibited by program instructions. If either type of interrupt is inhibited during a running program, the associated indicator will light. (During the operation of most programs, these lights may be seen flickering on and off.) All three classes of interrupt may be inhibited manually by setting the rotary switches at the bottom of the console to I (inhibit), however this setting is not recommended to anyone except UNIVAC Field Engineering personnel.

Errors that occur in the peripheral units will be indicated by a red light on the appropriate Channel Abnormal switch/indicator. Each input/output device is assigned one of the Central Processor's eight I/O channels.

The following are the standard channel assignments:

Channel 0 - Printer
Channel 1 - Card Reader
Channel 2 - Card Punch
Channel 3 - Communications
Channel 4 - Tape Read
Channel 5 - Tape Write
Channel 6 - FASTRAND
Channel 7 - Unassigned

If a red Channel Abnormal light goes on, the operator should check the unit(s) associated with that channel for error conditions and clear them before attempting to continue the run. Clearing the error at the peripheral location will usually cause the error light on the console to go out, and program operation can be resumed.

All standard UNIVAC 1050 software routines have display stops in them which will occur simultaneously with a channel abnormal error stop to indicate the nature of the problem in the peripheral unit. (For detailed descriptions of these stops, refer to the software routine's operating instructions.)

5.3.3. Sense Switches and Operator Request

Immediately above the five Mode buttons in the lower right-hand portion of the console is a row of four switch/indicators. The leftmost three are Sense switches, which may be used by programs to determine one among alternative courses of action (for example, to produce output in punched card format rather than a printed listing). The use of Sense switches for any given program should be outlined in the operating instructions for that program. If no mention is made of Sense switches in the operating instructions, it is understood that they should all be off (light extinguished).

The OPERATOR REQUEST button can be depressed when illuminated to interrupt a running program. Operator request is another form of Class II program interrupt. For tape systems running under the Executive Routine, depressing this button causes a unique display stop at which one of several courses of action may be selected. (For a detailed discussion of these alternatives, see the tape system software operating instructions.)

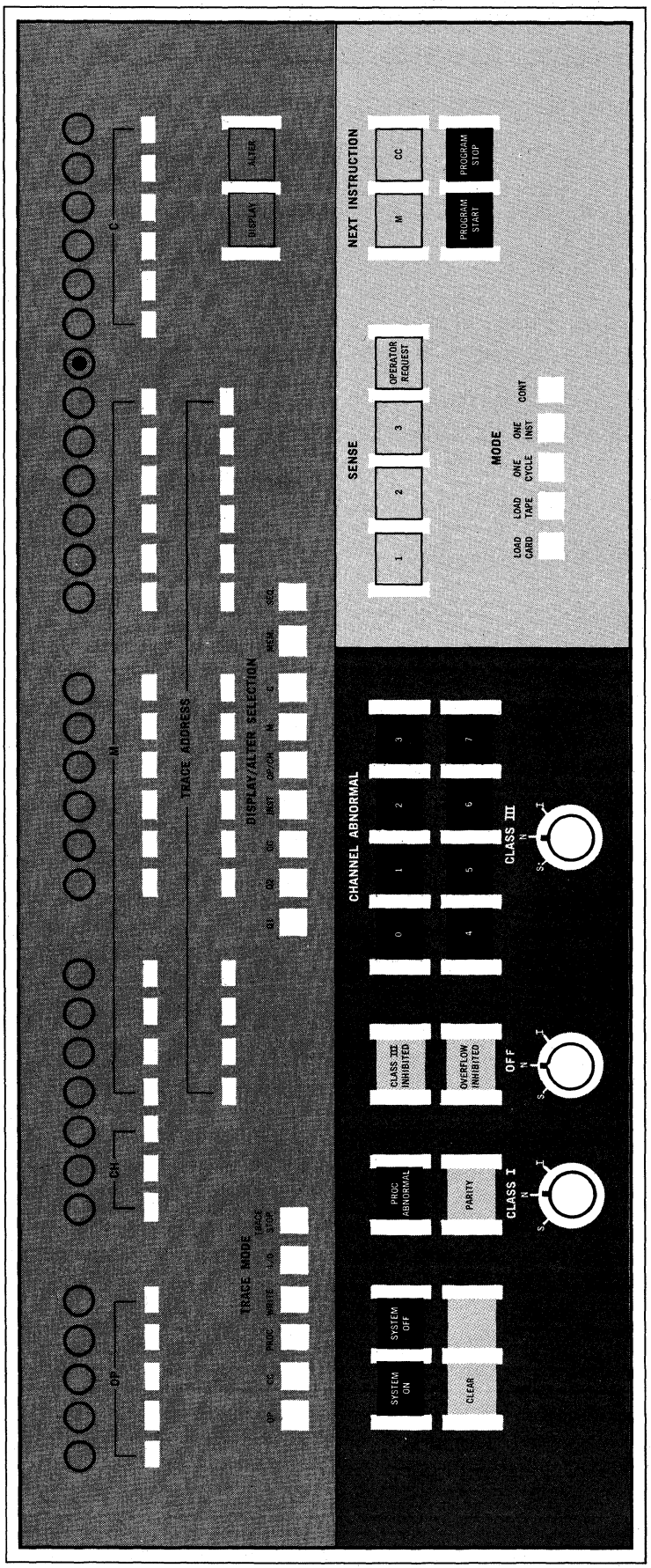


Figure 5-1. Central Processor Console.

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 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 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 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

DIVISION OF SPERRY RAND CORPORATION