File No. 1620-13
Form A26-5764-0

IBM Systems Reference Library

IBM 1620 Data Processing System, Model 2 Binary Capabilities and Index Registers

This publication explains the 1620 Model 2 special features, Binary Capabilities and Index Registers. The instructions for both features are described and illustrated with data flow diagrams. The Binary Capabilities section includes a brief explanation of the binary, octal, and decimal number systems.

Copies of this and other IBM publications can be obtained through IBM Branch Offices. Comments concerning the contents of this publication may be addressed to: IBM, Product Publications Department, San Jose, California

CONTENTS

PREFACE

The Index Register and Binary Capabilities features increase the programming flexibility and performance of the 1620 Model 2. The fourteen 5-position index registers are valuable for modification of instructions and simplification of program loops. The Binary Capabilities feature allows direct input and output of binary data via the IBM 1621 Paper Tape Reader and the IBM 1624 Tape Punch. No intermediate steps are necessary. Once binary data is read into the 1620, it can be processed in octal or decimal form.

Index registers offer savings in program steps,core storage, and computer processing time. In addition, the programming of repetitive calculations or operations is greatly simplified.

An index register allows modification of the P and Q addresses of 1620 program instructions without actually changing the addresses in the instructions themselves — the address modifications take place in the address registers. The contents of the index register are algebraically added to the specified addresses during execution of the instruction.

With the 1620 Model II, two bands of seven (total of 14) index registers (IX) are available. The first seven (Band 1) are located in core storage positions 00305 through 00339; Band 2 uses locations 00345 through 00379.

PROGRAMMING ADVANTAGES

The index register feature offers many programming advantages, the greatest of which is the ability to modify instructions in a program loop. For example, without index registers, to add four fields to another field, it is easier (and as efficient) to use four separate add instructions rather than to program a loop to use one add instruction four times. With index registers, however, the operation can be programmed with one add instruction and one instruction to modify the index register.

When more than one instruction address in a program loop requires modification, an even greater savings in core storage and programming time can result with the use of an index register. For example, assume that core storage contains two sets of ten quantities, A1 through A10 and B1 through B10. The problem is to program a loop to add A1 to B1, A2 to B2, . . . A10 to B10. Without index registers, and disregarding initialization, this requires a minimum of four instructions.

With the indexing feature, the 1620 Model 2 can be programmed to solve the above problem in two steps — one add instruction, and a combination compare-modify-branch instruction (see the Branch Conditionally and Modify Index Register instruction).

As stated previously, two bands of 7 index registers are available. An instruction is provided to select the seven index registers (IX) in Band 1, or the seven in Band 2, or to select the "no IX" mode.

Only one IX of a band is used at a time. The individual IX is selected by a combination of flag bits in the tens, hundreds, and thousands positions of the P and/or Q addresses of the instruction to be modified. As shown in Table 1, an instruction address with a flag bit in the tens position specifies IX 1 (of the band previously selected). Likewise, an instruction address with flag bits in all three positions specifies IX 7 of the selected band.

An instruction address that contains an index register flag is said to be indexed.

Table 1. Index Register Identification

Address	IX
000$\bar{0}$0	1
00$\bar{0}$00	2
00$\bar{0}\bar{0}$0	3
0$\bar{0}$000	4
0$\bar{0}$0$\bar{0}$0	5
0$\bar{0}\bar{0}$00	6
0$\bar{0}\bar{0}\bar{0}$0	7

EFFECTIVE ADDRESS COMPUTATION

Computation of the effective address uses five digits from the index register regardless of any flags in the IX. Flags in the IX are not added to the effective address.

All indexed addresses are considered positive even if a flag is over the units position; the IX value, however, can be negative; and when the algebraic sum of the two is negative, the result is expressed in 10's complement form (see Figure 1). An invalid effective address, such as that shown in Figure 1, will result in a MAR check.

NOTE: The computation of an effective address does not alter the condition of the High/Positive, Equal/Zero, or Overflow indicators.

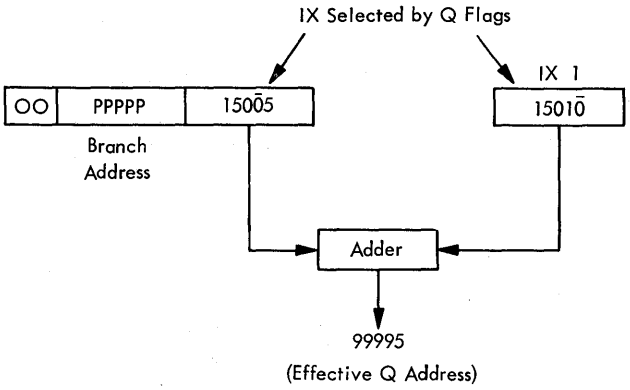


Figure 1. Indexing, Negative Result after Modification

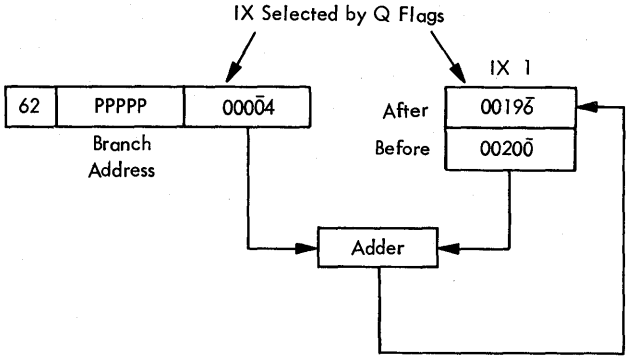


Figure 2. Index Register Modification

INDEXING EXECUTION TIME

An additional 5 memory cycles or 50 microseconds are required during I-time for each address that is indexed.

INDEXING AND INDIRECT ADDRESSING

Any P or Q address that is a core storage location can be indexed. Any address that can be indexed can also be an indirect address. When an address is both indirect and indexed, the indexing takes precedence — the resulting effective address then becomes an indirect address. The address specified by the indirect address can also be indexed and/or indirect. This chain can continue until the final effective address is not an indirect address.

INDEX REGISTER MODIFICATION

The new contents of a modified IX is the algebraic sum of the former contents and the modifying amount (see Figure 2). When a sign change occurs, the correct result is in the IX. Negative amounts are expressed in true form, e.g., minus 196 as 00196.

When the "no index register" mode is selected and an IX operation code (61-67) is to be executed, the operation is treated as invalid. When no IX is selected for an IX operation code (due to no Q flags), the operation is performed for IX "zero." Band 1 IX "zero" is located at positions 00300-00304; Band 2, at positions 00340-00344. The contents of IX "zero" are not used to compute an effective address.

PROGRAM INDICATORS

Three additional indicators come with the IX feature. These indicators provide the ability to determine, by programming, which band of IX is selected. The condition of the indicators can be tested by the Branch Indicator or Branch No Indicator instructions. The indicators are not affected by the testing.

The indicators as shown in Table 2 are considered to be ON when the condition shown exists.

Table 2. IX Band Program Indicators

Indicator	Condition
30	Neither Band Selected
31	Band 1 Selected
32	Band 2 Selected

INDEX REGISTER INSTRUCTIONS

Eight additional instructions are provided with the index register feature: seven for loading, storing, and modifying, and one Move Address instruction. Provision is made through the Branch and Select (BS-60) instruction to select Band 1, Band 2, or "no band."

The Q addresses of the new instructions either specify the location of data or contain data (immediate instructions). Flag bits are used in the Q_8, Q_9, and Q_{10} positions to specify the IX (see Table 1). The P address is a branch address.

Five of the instructions are unconditional branch instructions — after IX loading, storing, or modifying, the computer branches to the P address. Two instructions are of the conditional branch type — after modification of an IX, a branch is initiated only if the contents of the modified IX are not zero or have not changed sign.

Branch and Select (BS-60)

Description. The Branch and Select instruction provides for IX band selection or no band selection. The selected band remains selected until another Branch and Select instruction is executed or until power is removed from the system. The Reset key has no effect on band selection. The "no band" mode is selected automatically when power is applied to the system. The Q_7-Q_{10} positions of the instruction are not used. The selection desired is indicated by the Q_{11} digit as shown below:

0 - IX Band 0 (no band)
1 - IX Band 1 (IX 1-7)
2 - IX Band 2 (IX 1-7)

The P address specifies the next instruction to be executed.

Execution time. 60 microseconds

Branch and Modify Index Register (BX-61)

Description. The BX instruction is used for IX modification. The field designated by the Q address is added to the selected IX. The IX is selected by flags in the Q_8-10 positions of the instruction (Figure 3).

The field to be added to the IX is not limited to five digits; thus, one or more IX can be modified with one instruction. It is the high order flag bit in the P and Q fields that defines the length of the field to be modified and the length of the modifying field.

The High/Positive or Equal/Zero indicator is set according to the results of the modification. The Overflow Check indicator is turned ON only if the Q field is longer than the IX field. If the length of the Q field is less than or equal to the length of the IX field, an overflow may result from a "carry" out of the high order position of the IX field. However, in this case the Overflow Check indicator is not turned ON and the overflow digit is lost.

The P address specifies the next instruction to be executed.

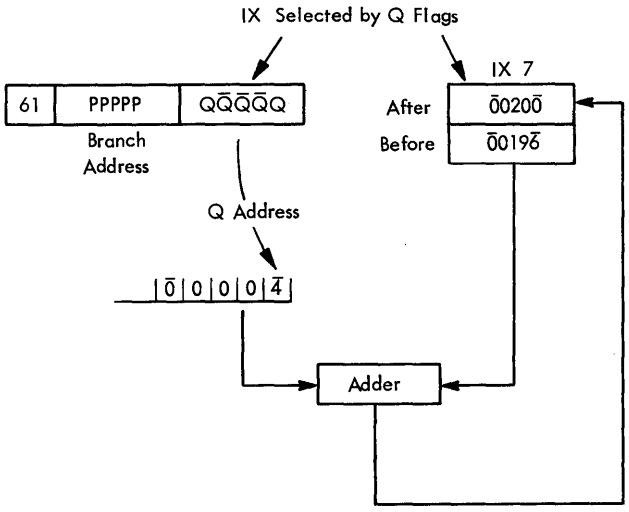


Figure 3. Branch and Modify Index Register (BX-61)

Execution time. 6.5 + .5L_q + L_x (average) memory cycles, where L_q is the length of the Q field, L_x is the length of the IX field. (A memory cycle is 10 microseconds.)

Branch and Modify Index Register Immediate (BXM-62)

Description. The BXM instruction is similar to the Branch and Modify Index Register instruction except that the five digits in the Q portion of the instruction are used as the modifier (see Figure 4). Flags in the Q field specify the IX to be modified.

Execution time. 140 microseconds.

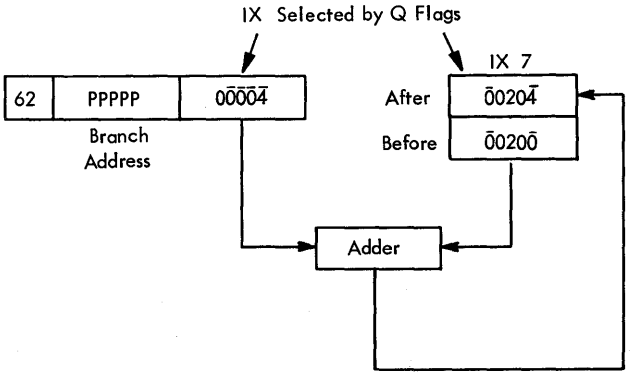


Figure 4. Branch and Modify Index Register Immediate (BXM-62)

Table 3. Conditional Branch Examples

Modifier	IX (before)	IX (after)	Branch
00005	00015	00010	Yes
00010	00010	00000	No
00010	00008	00002	No
00005	00025	00020	Yes
00020	00020	00000	No
00010	00005	00005	No
00006	99995	00001	No
00040	99970	99930	Yes
00035	99985	00020	No

Branch Conditionally and Modify Index Register (BCX-63)

Description. The BCX instruction is similar to the Branch and Modify Index Register instruction except that a branch to the P address is conditional. The P address specifies the next instruction to be executed if (after modification) the sign of the IX has not changed or the result is not zero. If the sign has changed or the result is zero the next sequential instruction is executed. If an overflow would result from a carry out of the high order position of the IX field, the next sequential instruction is executed (see Table 3).

Execution time. $6.5 + .5L_q + L_x$ (average) memory cycles, where L_q is the length of the Q field, and L_x is the length of the IX field. (A memory cycle is 10 microseconds.)

Branch Conditionally and Modify Index Register Immediate (BCXM-64)

Description. The BCXM instruction is similar in operation to the Branch Conditionally and Modify Index Register instruction except that the five digits in the Q portion of the instruction are used as the modifier (see Figure 4). Flags in the Q field specify the IX to be modified.

Upon completion of the IX modification, a branch to the P address is conditional as described for the BCX instruction.

Execution time. 140 microseconds.

Branch and Load Index Register (BLX-65)

Description. The Branch and Load Index Register instruction loads the five digits of data specified by the Q address to the selected IX. The flags in Q_{8-10} positions select the IX.

A flag bit in the units position of the Q field is transferred to the IX and a flag is automatically placed over the high order position of the IX field. No other flags are transferred (see Figure 5).

The P address specifies the next instruction to be executed.

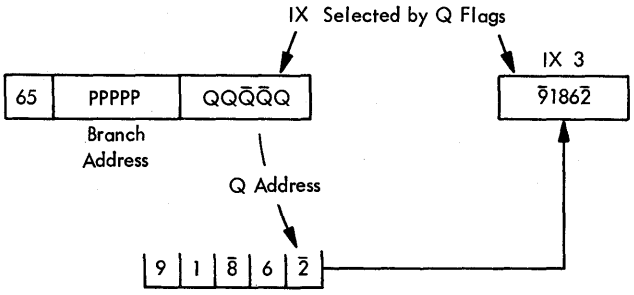


Figure 5. Branch and Load Index Register (BLX-65)

Execution time. 140 microseconds

Branch and Load Index Register Immediate (BLXM-66)

Description. The BLXM instruction is similar to the Branch and Load Index Register instruction except that the Q portion of the instruction is used as the data.

Execution time. 140 microseconds.

Branch and Store Index Register (BSX-67)

Description. The BSX instruction stores the five digits of the selected IX at the Q address (see Figure 6). If the contents of the IX are negative, a flag is stored with the units position. A flag is

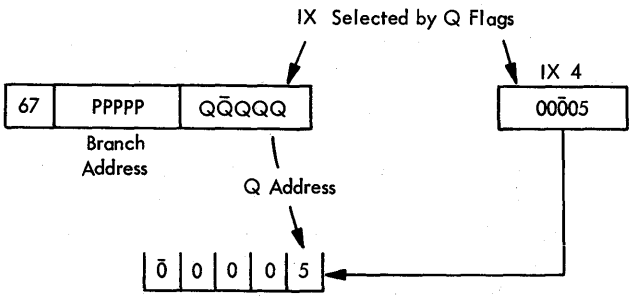


Figure 6. Branch and Store Index Register (BSX-67)

automatically placed in the high order position of the stored data. No other flags are transmitted.

Execution time. 140 microseconds.

Move Address (MA-70)

Description. The Move Address instruction causes the five digits specified by the Q address of the instruction to be moved to the P address (see Figure 7). Flags in the P field remain unchanged.

Execution time. 140 microseconds.

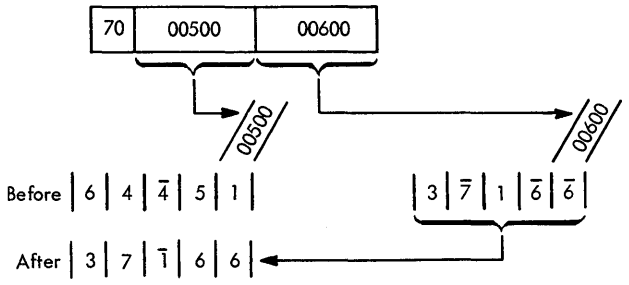


Figure 7. Move Address (MA-70)

BINARY CAPABILITIES

The Binary Capabilities feature enables the 1620-2 to process binary data for such applications as process control, missile and satellite tracking, and weather observation. Binary data is collected in octal form, can be converted to decimal form for computation, and processed for binary, octal, or decimal output. A brief treatise of each number system follows.

Decimal

Mankind was endowed with ten fingers — the probable reason for universal acceptance and use of the decimal or base ten system. The decimal number 1375, for example, can be expressed as $1(10^3) + 3(10^2) + 7(10^1) + 5(10^0)$ which equals 1000 + 300 + 70 + 5. (Any number with a zero exponent equals one.)

Binary

Binary data consists of two digits, zero and one. Thus, the decimal digits 0 through 9 are expressed in binary form, as follows:

Decimal	Binary	Decimal	Binary
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

Note that a one in the low-order position of the binary number is equivalent to a decimal one; a one in the second position is equivalent to a decimal 2; a one in the third position, to a 4; a one in the fourth position, an 8; a fifth position would be 16; the sixth, 32, and so on. The decimal equivalent of each position is twice that of the position to its right.

Octal

Octal data is expressed to the base 8 just as decimal and binary data are expressed to the base 10 and 2. Thus, equivalent octal numbers progress as follows:

Decimal	Octal	Decimal	Octal
1	1	17	21
2	2	23	27
7	7	24	30
8	10	31	37
9	11	32	40
15	17	39	47
16	20	40	50

The octal number 2537, for example, is equal to the decimal number 1375; this relationship is explained in the following section.

Numbers in systems other than decimal are identified by subscripts. Thus, the preceding octal number 2537 is expressed as 2537_8.

Numbers without subscripts are assumed to be decimal.

NUMBER CONVERSION

The Read Binary instruction of the Binary Capabilities feature reads binary data from the 1621. The binary data is stored in octal form as it enters core storage. Although this conversion is done automatically, a brief comparison of the two forms is given to show their relationship.

Binary to Octal Conversion

The first three positions of the binary number system can have a maximum decimal equivalent of 7 (111_2=7). Because seven is the highest number in the octal system, a binary number can be separated into 3-position groups — each group representing the equivalent octal number. 010101011111_2 for example, can be separated as follows:

$$\underbrace{010}_{2}\ \underbrace{101}_{5}\ \underbrace{011}_{3}\ \underbrace{111}_{7}$$

Thus, $010101011111_2 = 2537_8$

Octal to binary conversion is, of course, simply the reverse of binary to octal.

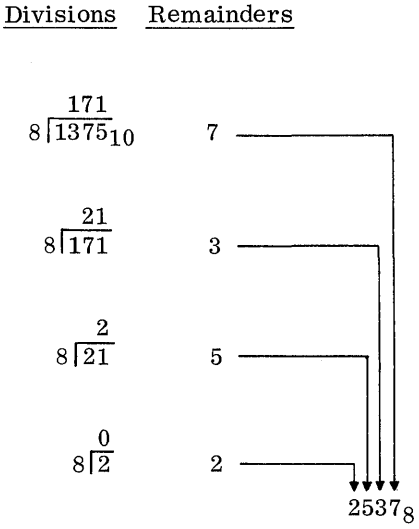
Octal to Decimal Conversion

Octal numbers are converted to decimal numbers by expanding each position, as follows:

2537_8 equals $2(8^3) + 5(8^2) + 3(8^1) + 7(8^0)$
$2(512) + 5(64) + 3(8) + 7(1)$.
$2537_8 = 1375_{10}$

Decimal to Octal Conversion

Decimal numbers are converted to octal numbers by successive divisions by eight until a zero quotient is obtained. The remainders, in the reverse order, form the octal number. For example, 1375_{10} is converted to 2537_8 as follows:



INSTRUCTIONS

Ten instructions are provided with the Binary Capabilities feature. Octal fields have the same flexibility and restrictions that exist for decimal fields. Generally, the field length must be at least two digits in length, a flag is used to specify the high-order digit of the field, etc.

Branch On Bit (BBT-90)

Description. The Q_8 through Q_{11} digits of the instruction specify a four digit core storage address (0000-9999). The bit configuration of the data at this address is compared with the bit configuration of the Q_7 digit of the instruction. If any bit of the Q_7 digit corresponds to any bit in the data specified by Q_8-Q_{11} (the C bit is excluded from this comparison), the 1620-2 branches to the P address. If no successful comparison occurs, the next instruction in sequence is executed. Both the P address and the Q_8-Q_{11} address may be indirect addresses. The Q_8-Q_{11} address is restricted to the first 10,000 positions of core storage. An indirect Q_8-Q_{11} address, however, makes possible the placement of data at any core storage address.

Figure 8 shows that data at 00500 is compared with the Q_7 digit. Since there is no 1-bit in the data at 00500, the branch does not occur.

Execution time. 70 microseconds.

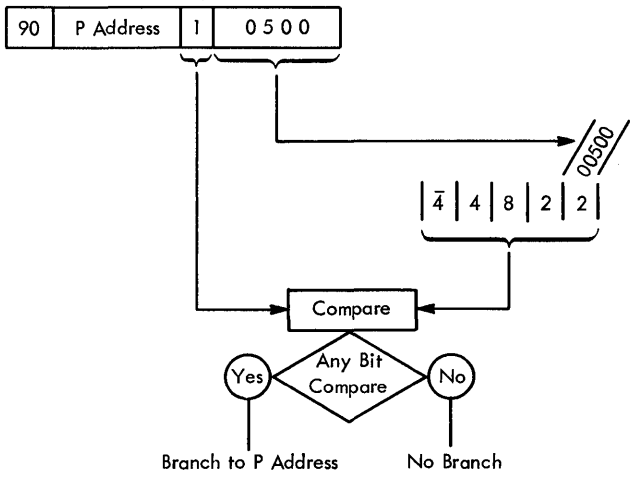


Figure 8. Branch on Bit (BBT-90)

Branch on Mask (BMK-91)

Description. The Q_8 through Q_{11} digits of the instruction specify a four digit core storage address (0000-9999). The bit configuration of the data at this address is compared with the bit configuration of the Q_7 digit of the instruction. A branch to the P address occurs when the 8, 4, 2, and 1 bits that make up the Q_7 digit are also present in the field specified by the Q_8-Q_{11} address. The flag bit is also considered in the comparison only if the mask digit at Q_7 contains a flag bit. If the mask does not contain a flag bit, the presence of a flag bit in the data at Q_8-Q_{11} is of no consequence. If the comparison is not successful, the next instruction in sequence is executed.

Although the Q_8-Q_{11} address is restricted to the first 10,000 positions of core storage, an indirect Q_8-Q_{11} address makes possible the placement of data at any core storage address. Both the

P address and the Q_8-Q_{11} address may be indirect addresses.

Execution time. 70 microseconds.

OR and AND Logic

OR and AND logic consists of two or more inputs and a resulting output based on the presence or absence of input data. As shown in Figure 9, the OR logic provides an output which is a composite of all the inputs.

Input Bits	Output Bits
0011	0111
0100	

Figure 9. OR Logic

The AND logic (Figure 10) provides an output which consists only of those bits that appear on each and every input line.

Input Bits	Output Bits
0011	0001
0101	

Figure 10. AND Logic

OR to Field (ORF-92)

Description. The 4, 2, and 1 bits of each digit of the P field and the 4, 2, and 1 bits of the corresponding digits of the Q field become OR logic inputs. C bits and 8 bits are ignored. The resulting output replaces the P field data. C bits are added where required for correct parity. Flag bits in the P field are retained. Eight bits in the P field are destroyed. The first flag bit in the Q field terminates the operation. Q field data is not altered.

Figure 11 shows data flow for the instruction 92 00500 00600. The resulting data 3767 replaces the original P data 9127. The flag bit in the tens position of the P field is not disturbed. The Equal/Zero indicator is turned on if the resultant field has no numerical bits.

Execution time. $60+20L_q$ microseconds.

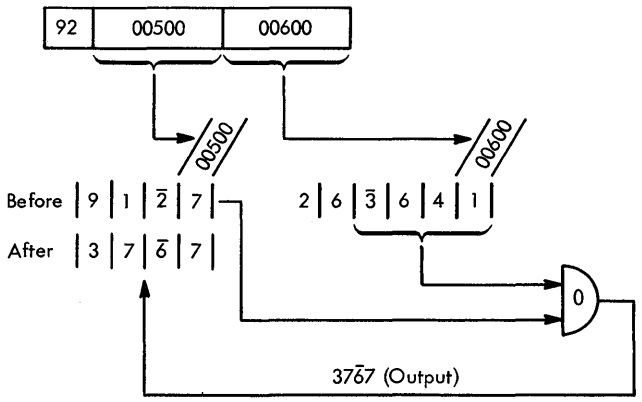


Figure 11. OR to Field (ORF-92)

AND to Field (ANDF-93)

Description. The ANDF instruction operates in the same manner as the ORF instruction except that AND logic is used in place of OR logic.

For example, using the same data as that used in Figure 11, the resulting output is 1001, as shown in Figure 12. The Equal/Zero indicator is turned on if the resultant field has no numerical bits.

Execution time. $60 + 20L_q$ microseconds.

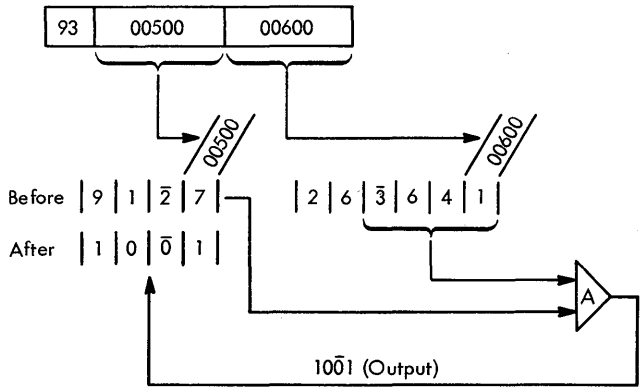


Figure 12. AND to Field (ANDF-93)

Exclusive OR to Field (EORF-95)

Description. The operation of the Exclusive OR to Field instruction is similar to that of the ORF (92) instruction. The only difference is that the octal data in the P and Q fields is exclusive ORed — that is, the 1, 2, and 4 bits that are in either (not both) the P field or Q field are stored in the resultant P field. For example:

OR to Field

Input Bits	Output Bits
0011	0111
0101	

Exclusive OR to Field

Input Bits	Output Bits
0011	0110
0101	

The Equal/Zero indicator is turned on if the resultant field has no numerical bits.

Execution time. $60 + 20L_q$ microseconds.

Complement Octal Field (CPLF-94)

Description. The Q data is complemented on an octal basis and transmitted to the P field. Conversion and transmission are terminated by the first flag bits detected in the Q data field even if a flag bit is present in the units position. The flag in the Q field is transmitted.

Since the octal system has no digit higher than 7, only the 4, 2, and 1 bits of each digit are complemented and transmitted. The 8 bits are neither complemented nor transmitted. The C bit is added to each complemented digit when required for parity. For example, a 4 digit is complemented to a 3 digit, and a C bit is added for parity. All replaced P data is lost. The Equal/Zero indicator is turned on if the resultant field has no numerical bits.

Figure 13 shows that Q data at 00600 is complemented to 76035 and stored at the P address. The original Q data remains at the Q address. Table 4 shows how the numerical bits of each digit are complemented. Although the 8 bit is shown with the 4, 2, and 1 bits of each digit, it, like the C and F bits, is not complemented. Note, however, that the 8, 4, 2, and 1 bits actually form the binary representation of the decimal digit.

Execution time. $60 + 20L_q$ microseconds.

Table 4. Complementing Octal

Q Data (Decimal)	8	9	7	4	2
Bit configuration of each Q digit	1000	1001	0111	0100	0010
Q Data (Octal)	0	1	7	4	2
Bit configuration after complementing (8 bits disregarded)	0111	0110	0000	0011	0101
Resulting P Data (Octal)	7	6	0	3	5

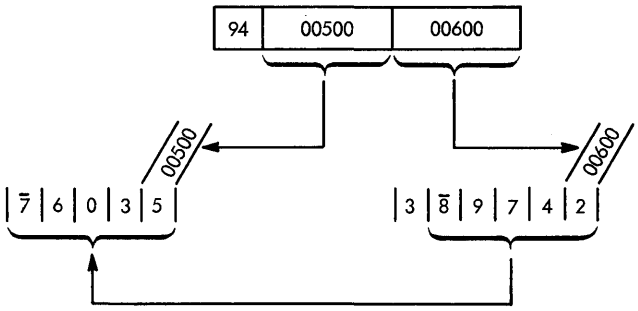
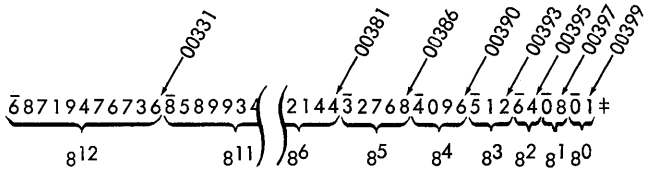


Figure 13. Complement Octal Field (CPLF-94)

Octal to Decimal Conversion (OTD-96)

Description. The principle of octal to decimal conversion has been previously explained. The 1620-2 accomplishes this conversion by using a table of numbers to the base eight. (A table of any number system less than or equal to nine can be used to convert numbers to base ten.)

If the octal fields to be converted do not exceed 13 digits, the table need not contain any powers of eight higher than 12, as follows:



From the table it can be seen that 8^{12} equals 68,719, $476,736_{10}$ which equals 1000000000000_8 (13 digits). The larger the size of the octal field, the higher the power of eight number that must be stored. A 14 digit octal number would require that the table be expanded to include 8^{13}. The table may be stored anywhere in core storage.

The core storage address of the units digit of each power of eight number must be known to the programmer. If the 8^0 number (01) is stored at 00399, the 8^1 units digit would be stored at 00397, the 8^2 units digit at 00395, the 8^3 units digit at 00393, etc.

The P address of the OTD instruction specifies the address of the units digit of the 8^0 number in the table -- 00399 in the example above.

The Q address specifies the address of the units digit of the octal field to be converted. A flag bit determines the length of the Q field.

The decimal equivalent of the octal number is developed from the power-of-eight table, as follows: Recall the principle of octal to decimal conversion,

where $2437_8 = 2(8^3) + 5(8^2) + 3(8^1) + 7(8^0)$. Note that each digit of the octal number correlates with a power-of-eight number — the units digit with the 8^0, the tens digit with 8^1, the hundreds digit with 8^2, etc. Each power-of-eight number is multiplied by its associated octal digit. The multiply table is used and the products of these multiplications are developed and summed in the product area. The units digit and sign of the converted number is located at 00099; the high-order digit is automatically flagged.

Execution time. $280 + 10L_q\ (2L_q-1)$ microseconds maximum).

Decimal to Octal Conversion (DTO-97)

Description. The principle of decimal to octal conversion has been previously explained. The 1620-2 accomplishes decimal to octal conversion in a similar manner, as follows:

The P address of the DTO instruction specifies the address where the high-order digit of the resultant octal number is to be stored.

The Q address specifies the address of the units digit of the power-of-eight number to be used in the first subtraction. Recall that decimal to octal conversion is a series of divisions. Automatic Division is accomplished in the 1620 by successive subtractions. The power-of-eight number that must be addressed is one less than the number of digits that will be in the resultant octal field. If the octal number is to be five digits long, the fifth table entry (8^4 or $\bar{4}096$) is addressed. The programmer must know the size of the octal field that will be developed.

The decimal field to be converted must be located at 00099 before the DTO instruction is executed. The divisor, specified by the Q address, is successively subtracted from the decimal number. Because the Q address specifies a divisor whose value is in close proximity to the value of the dividend, the remainder has no significance and no more than seven successful subtractions should occur. An eighth successful subtraction would turn on the Overflow indicator.

The quotient digit is developed in the Multiplier Quotient register and automatically transferred to the P address. This first quotient digit, which is the high-order digit of the octal number is automatically flagged. The next lower power-of-eight number is then subtracted from the remaining dividend and the second quotient digit is developed and transferred to the P address plus one. This series of divisions of power-of-eight numbers into remaining dividends continues until the last entry (8^0) has been used as a divisor. The last entry is defined by a ‡ to the right. Figure 14 shows how the decimal number

Instruction: 97 00500 00386		
Decimal number at 00095-00099	$\bar{9}9999$	Stored octal digits
8^5 number at 00382-00386	$\bar{3}2768$	
First subtraction	67231	$\bar{1}$
	32768	
Second subtraction	34463	$\bar{2}$
	32768	
Third subtraction	01695	$\bar{3}$
8^4 number at 00387-008390	$\bar{4}096$	
No successful subtraction	01695	$\bar{3}0$
8^3 number at 00391-00393	512	
First subtraction	01183	$\bar{3}01$
	512	
Second subtraction	00671	$\bar{3}02$
	512	
Third subtraction	00159	$\bar{3}03$
8^2 number at 00394-00395	64	
First subtraction	00095	$\bar{3}031$
	64	
Second subtraction	00031	$\bar{3}032$
8^1 number at 00396-00397	08	
After three subtractions	00007	$\bar{3}0323$
8^0 number at 00398-00399	01	
After seven subtractions	00000	$\bar{3}03237$

Figure 14. Decimal to Octal Conversion

$\bar{9}9999$ is converted to the octal number $\bar{3}03237$. The instruction 97 00500 00386 causes the 8^5 number to be subtracted from the decimal number. Three successful subtractions occur and the quotient digit 3 is stored at the P address 00500. The 8^4 number cannot be successfully subtracted from the remaining dividend (overdraws and corrections are not shown) and a zero is transferred to 00501. The 8^3 number is successfully subtracted three times, and a 3 is transferred to 00502. Two successful subtractions occur with the 8^2 number and a two is transferred to 00503. The 8^1 number and the 80 number are successfully subtracted 3 and 7 times, respectively. The detection of the ‡ at 00400 stops the instruction execution. The developed quotient digits form the octal number $\bar{3}03237$ at core storage locations 00500-00505.

Read Binary Paper Tape (RBPT-37, $Q_{8,9}$-33)

Description. The Read Binary Paper Tape instruction operates similar to the Read Alphamerically (paper tape) instruction. Both instructions trans-

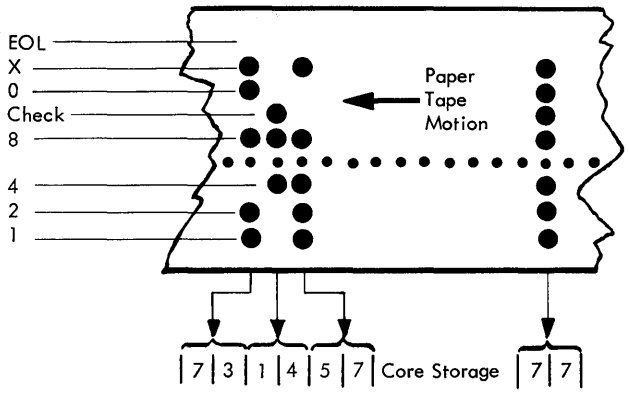


Figure 15. Read Binary Translation

late a single column to the 1620 two-digit form. The difference is that the RBPT instruction transfers the 8, 0, and X tracks to even-numbered core storage locations as 1, 2, and 4 bits, respectively. The 1, 2, and 4 paper tape tracks are entered as 1, 2, and 4 bits into odd numbered core storage positions. C bits are added to the core storage positions wherever necessary to maintain correct parity.

The P address of the RBPT instruction must refer to an odd numbered storage location. Reading continues until an End of Line (EOL) character is interpreted. Note that the tape feed code is converted to 77 in core storage as shown in Figure 15.

Execution time. Depends on size of input record.

Write Binary Paper Tape (WBPT-37, $Q_{8,9}$-32)

Description. The Write Binary Paper Tape instruction causes the 1, 2, and 4 bits of two adjacent core storage positions to be punched into one column of paper tape. The 1, 2, and 4 bits of odd-numbered core storage positions are punched into the 1, 2, and 4 tracks, respectively. The 1, 2, and 4 bits of even numbered core storage positions are punched into the 8, 0 and X tracks, respectively.

The P address of the RBPT instruction must refer to an odd numbered position of core storage. Writing (punching) continues until an alphameric record mark (0‡) character is encountered in core storage. The record mark is translated to an EOL (end-of-line) punch in paper tape.

Execution time. Depends on size of output record.

IBM 1620 Printed in U. S. A. A26-5764-0

IBM®
International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, New York