NCR REFERENCE MANUAL

An Educational Publication

number: 12
page: 1 of 59
date: Jan. 76

ST-9402-08
BINDER NO. 0141

NCR CENTURY 251 AND 300 HARDWARE COMMANDS

This publication, intended as a supplement to the "NCR Century 251 Processor" and "NCR Century 300 Processor" publications, contains functional descriptions of the hardware commands available for both processors. The following table contains a list of the commands, the mnemonics for the commands, the command code representations, and the page numbers of the command descriptions.

COMMAND TABLE AND COMMAND DESCRIPTION INDEX

COMMAND	MNEMONIC	CODE			PAGE
		DEC. REP.	HEX. REP. 8 BYTE	HEX. REP. 4 BYTE	
BINARY COMMANDS					
WORD ADD BINARY	WADD	66	42	C2	4
WORD SUBTRACT BINARY	WSUB	67	43	C3	5
WORD COMPARE BINARY	WCOMP	51	33	B3	5
WORD SHIFT BINARY	(*)	53	35	B5	6
WORD MULTIPLY BINARY	WMULT	54	36	B6	7
WORD DIVIDE BINARY	WDIV	55	37	B7	7
ADD BINARY	BADD	96	60	E0	8
SUBTRACT BINARY	BSUB	97	61	E1	8
COMPARE BINARY	BCOMP	101	65	E5	9
BINARY TO DECIMAL CONVERSION	CVD	114	72	F2	9
DECIMAL TO BINARY CONVERSION	CVB	115	73	F3	10
FLOATING POINT COMMANDS					
F. P. ADD SINGLE	FADD	116	74	F4	11
F. P. ADD DOUBLE	FADDD	117	75	F5	12
F. P. SUBTRACT SINGLE	FSUB	118	76	F6	12
F. P. SUBTRACT DOUBLE	FSUBD	119	77	F7	13
F. P. MULTIPLY AND ADD SINGLE	FMADD	120	78	F8	13
F. P. MULTIPLY AND ADD DOUBLE	FMADDD	121	79	F9	14
F. P. COMPARE SINGLE	FCOMP	122	7A	FA	14
F. P. COMPARE DOUBLE	FCOMPD	123	7B	FB	15
F. P. MULTIPLY SINGLE	FMULT	124	7C	FC	15
F. P. MULTIPLY DOUBLE	FMULTD	125	7D	FD	16
F. P. DIVIDE SINGLE	FDIV	126	7E	FE	16
F. P. DIVIDE DOUBLE	FDIVD	127	7F	FF	17
DECIMAL COMMANDS					
ADD SIGNED	PADD	64	40	C0	19
SUBTRACT SIGNED	PSUB	65	41	C1	19
COMPARE SIGNED	PCOMP	69	45	C5	20
MULTIPLY SIGNED	PMULT	93	5D	DD	21
DIVIDE SIGNED	PDIV	113	71	F1	22
PACK	PACK	76	4C	CC	24
UNPACK	UNPACK	77	4D	CD	26
ADD UNSIGNED	UADD	98	62	E2	27
SUBTRACT UNSIGNED	USUB	99	63	E3	28

COMMAND TABLE AND COMMAND DESCRIPTION INDEX

COMMAND	MNEMONIC	CODE			PAGE
		DEC. REP.	HEX. REP.		
			8 BYTE	4 BYTE	
MOVE DATA COMMANDS					
MOVE B RIGHT TO LEFT	MVBR	68	44	C4	29
MOVE A LEFT TO RIGHT	MVAL	84	54	D4	30
MOVE A RIGHT TO LEFT	MVAR	100	64	E4	30
LOGIC COMMANDS					
EDIT	EDIT	73	49	C9	31
DECODE TO DELIMITER	DCODD	78	4E	CE	33
DECODE ALL	DCODA	79	4F	CF	35
SCAN ON KEY LESS THAN	SCANL	85	55	D5	36
SCAN ON KEY EQUAL	SCANE	86	56	D6	38
SCAN ON KEY GREATER THAN	SCANG	87	57	D7	40
TABLE COMPARE	TCOMP	92	5C	DC	42
LOGIC	LOGIC	94	5E	DE	43
TRANSFER COMMANDS					
COUNT	COUNT	74	4A	CA	45
JUMP	JUMP	75	4B	CB	45
TEST CHARACTER EQUAL	TESTCE	81	51	D1	46
TEST CHARACTER UNEQUAL	TESTCU	82	52	D2	46
TEST BIT	TESTB	83	53	D3	47
BRANCH OVERFLOW	BROV	104	68	E8	48
BRANCH LESS	BRL	105	69	E9	48
BRANCH EQUAL	BRE	106	6A	EA	48
BRANCH LESS OR EQUAL	BRLE	107	6B	EB	48
BRANCH GREATER	BRG	108	6C	EC	48
BRANCH LESS OR GREATER	BRU	109	6D	ED	48
BRANCH GREATER OR EQUAL	BRGE	110	6E	EE	48
BRANCH UNCONDITIONAL	BR	111	6F	EF	48
SPECIAL COMMANDS					
SET IP ON	IPON	70	46	C6	49
SET IP OFF	IPOFF	71	47	C7	50
RESTORE	RESTOR	72	48	C8	51
OPTION SWITCHES INPUT **	SWIN	80	50	D0	52
REPEAT	REPEAT	102	66	E6	53
WAIT **	WAITI	103	67	E7	54
INOUT (Send PAF)**	INOUT	112	70	N/A	54
INOUT (Load PR) **	INOUT	112	N/A	F0	55
LOAD T	LOADT	49	31	B1	55
LOAD BAR **	LDBAR	88	58	D8	56
LOAD MONITOR REGISTER **	LDMONR	89	59	D9	57
LOAD TRACE	LDT	90	5A	DA	57
STORE TRACE	STTR	91	5B	DB	57
MOVE EFFECTIVE B ADDRESS	MVEB	36	24	A4	58
TEST AND SET LOCK	TESTSL	50	32	B2	58

* The mnemonics for this command are described later in this publication under "WORD SHIFT BINARY" command description.

** These commands are privileged and can be executed only when the processor is in the supervisor state. An attempt to execute any of these commands when the processor is in the user state causes a command code trap.

The following symbols and notations are used in the command descriptions:

A = The effective A address (after command setup).
B = The effective B address (after command setup).
T = The effective T value (after command setup).
( ) = The contents of; e.g., (B) = the contents of B after command setup.
X = The value X after command execution; e.g., T, (B).

NOTE

T equals T, except after the execution of a RESTORE or LOAD T command. B is specified for each command; however, the specification is not necessarily correct if termination for a malfunction occurs.

The address of the leftmost character of every command must be 0 modulo 4. Attempting to execute a command that violates this requirement, or trying to access a memory location greater than physical memory size, results in a PE interruption.

All command descriptions assume that the command setup phase has been completed. Commands are described in terms of preserved values, both at the end of the setup phase and at the end of the execution phase.

If the A and B operands overlap, the results of the operation may be other than those expected by the programmer.

FIXED POINT BINARY COMMANDS

In the NCR Century 251 and 300 binary information may be represented in two forms:

- Variable length, positive, unsigned
- Fixed word length, signed

In either form, negative integers are represented as two's complements of the integers. The sign bit is in the 32nd (most significant) bit position of the 4-byte data word used with the signed binary commands.

Commands specifying unsigned binary fields may be used for address modification, clearing fields in memory, operating on variable length fields, etc.

Commands specifying signed binary fields may be used for high-speed computation, space conservation, shifting, etc.

WORD ADD BINARY (WADD)

COMMAND CODE: 66 (42) (C2)

Command Description

(A) is added to (B), with the result replacing (B).

The contents of A and B are 32-bit words: a 31-bit integer and a sign in the 32nd bit position. Negative integers are represented in two's complement form.

All 32 bits of the two operands are added. If the carry/no carry generation from the integer position (31st bit) does not equal the carry/no carry generation from the sign position (32nd bit), the overflow flag (OF) is turned ON.

If the addresses specified by A and B are not zero modulo four (0 mod 4), a programming error (PE) results.

T is not used.

B = B

WORD SUBTRACT BINARY (WSUB)

COMMAND CODE: 67 (43) (C3)

Command Description

(A) is subtracted from (B), with the result replacing (B).

The contents of A and B are 32-bit words: a 31-bit integer and a sign in the 32nd bit position. Negative integers are represented in two's complement form.

The contents of the memory location specified by A are converted to two's complement form and added to the contents of the memory location specified by B. All 32 bits of the two operands are added. If the carry/no carry generation from the integer position (31st bit) does not equal the carry/no carry generation from the sign position (32nd bit), the overflow flag (OF) is turned ON.

If the addresses specified by A and B are not zero modulo four (0 mod 4), a programming error (PE) results.

T is not used.

B = B

WORD COMPARE BINARY (WCOMP)

COMMAND CODE: 51 (33) (B3)

Command Description

(A) is compared algebraically to (B). Both operands are 32-bit words containing a 31-bit integer, and a sign in bit position 32. As a result of the comparison, the proper flag (L, E, or G) is turned ON to indicate whether the A operand is less than, equal to, or greater than the B operand. The other two flags are turned OFF. The A and B operands retain their original contents after command execution.

The conditions (A) > (B) and (A) = (B) cause the repeat indicator (RI) to be turned OFF following the command execution.

If the addresses specified by A and B are not zero modulo four (0 mod 4), a programming error (PE) results.

T is not used.

B = B

WORD SHIFT BINARY (*)

COMMAND CODE: 53 (35) (B5)

Command Description

(A) is shifted into (B) according to the contents of the T character of the command.

The T character serves two functions: Bits 1 through 6 determine the number of bit positions the contents of A are shifted, ranging from 0 (000000) through 63 (111111). Bits 7 and 8 designate the direction and the type of shift, as shown in the following illustration.

FUNCTIONS OF WORD SHIFT BINARY

b8	b7	*	OPERATION
0	0	SLL	SHIFT LEFT -- (A) is shifted left into (B) the number of bits specified by bits 1 through 6 of the T character. Binary zeros are inserted in vacated positions. Both (A) and (B) are 4-byte words.
0	1	SRL	SHIFT RIGHT -- (A) is shifted right into (B) the number of bits specified by bits 1 through 6 of the T character. Binary zeros are inserted in vacated positions. Both (A) and (B) are 4-byte words.
1	0	SLLD	SHIFT LEFT DOUBLE -- A left shift is performed. Both (A) and (B) are 8-byte double words.
1	1	SRLD	SHIFT RIGHT DOUBLE -- A right shift is performed Both (A) and (B) are 8-byte double words.

An attempt to shift more than 31 bit positions results in a programming error (PE), unless double word shifting is specified.

If the addresses specified by A and B are not zero modulo four (0 mod 4), a programming error (PE) results.

WORD MULTIPLY BINARY (WMULT)

COMMAND CODE: 54 (36) (B6)

Command Description

Binary multiplication is performed with (A) specified as the multiplier and (B) specified as the multiplicand. The multiplier and multiplicand are 32-bit words containing a 31-bit integer and a sign in the 32nd bit position.

The product, a 64-bit double word containing a 63-bit integer and a sign in the 64th bit position, is stored in the memory accumulator at locations 320 through 327. The sign of the product is determined algebraically.

Negative integers are represented in two's complement form.

If the addresses specified by A and B are not zero modulo four (0 mod 4), a programming error (PE) results.

T is not used.

B = 320

WORD DIVIDE BINARY (WDIV)

COMMAND CODE: 55 (37) (B7)

Command Description

Binary division is performed with (A) specified as the divisor and (B) specified as the dividend. The divisor is a 32-bit word containing a 31-bit integer and a sign in the 32nd bit position. The dividend is a 64-bit double word containing a 63-bit integer and a sign in the 64th bit position.

The quotient and the remainder are both 32-bit words containing a 31-bit integer and a sign in the 32nd bit position. The quotient, whose sign is determined algebraically, is stored in the memory accumulator at locations 320 through 323. The remainder, whose sign is identical to the sign of the dividend, is stored in the memory accumulator at locations 324 through 327.

If the magnitudes of the divisor and the dividend are such that the quotient does not fit into a 32-bit signed word, a programming error (PE) results and the contents of the accumulator are indeterminate.

If the divisor equals zero, the division is not performed, a PE occurs, and the accumulator remains unchanged.

If the addresses specified by A and B are not zero modulo four (0 mod 4), a programming error (PE) results.

T is not used.

B = 320

ADD BINARY (BADD)

COMMAND CODE: 96 (60) (E0)

Command Description

(A) is added binarily to (B), with the result replacing (B). The addition is performed from right to left. The overflow is not affected by this command. Both fields are considered unsigned and positive.

The A field retains its original contents, unless the A and B fields overlap. If these fields overlap, the results may be different from the results obtained if these fields do not overlap.

The T portion of the command specifies the number of bytes in each field. T may be any number from 0 through 255, with 0 being equivalent to 256.

B = B

SUBTRACT BINARY (BSUB)

COMMAND CODE: 97 (61) (E1)

Command Description

(A) is subtracted binarily from (B), with the result replacing (B). Subtraction is performed from right to left. Both fields are considered unsigned and positive.

Subtraction is actually performed by complementary addition. The two's complement of the A character is formed and added to the B character. The overflow is not affected by this command. If the contents of the A field are greater than the contents of the B field, the final result stored is the two's complement of (A) - (B).

The A field retains its original contents, unless the A and B fields overlap. If these fields overlap, the results may be different from the results obtained if these fields do not overlap.

The T portion of the command specifies the number of bytes in each field. T may be any number from 0 through 255, with 0 being equivalent to 256.

B = B

COMPARE BINARY (BCOMP)

COMMAND CODE: 101 (65) (E5)

Command Description

(A) is compared binarily to (B). As a result of the comparison, the proper flag (L, E, or G) is turned ON to indicate whether the A operand is less than, equal to, or greater than the B operand. The other two flags are turned OFF. The A and the B operands retain their original contents.

The conditions (A) > (B) and (A) = (B) cause the repeat indicator (RI) to be turned OFF following the command execution.

The T portion of the command specifies the number of bytes in each field. T may be any number from 0 through 255, with 0 being equivalent to 256.

B = B

NOTE

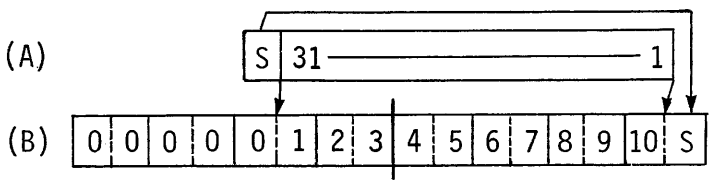
The sign bits of signed binary fields are compared just like the other bits in the field; therefore, negative numbers always appear greater than positive numbers.

BINARY TO DECIMAL CONVERSION (CVD)

COMMAND CODE: 114 (72) (F2)

Command Description

(A) is a 4-byte word containing the signed, binary number, right justified. (B) is an 8-byte double word which contains, after command execution, the signed, packed, decimal equivalent of (A), right justified, with the sign in the least significant digit position. The converted result is always 10 decimal digits, plus the associated sign digit, zero-filled to the left of the last significant digit, as shown in the following illustration.



Signed, 4-byte (31 bits plus sign bit) binary number

Signed, 8-byte (ten packed decimal digits plus sign digit) decimal number

If the addresses specified by A and B are not zero modulo four (0 mod 4), a programming error (PE) results, with (B) unchanged.

T is not used.

B = B

DECIMAL TO BINARY CONVERSION (CVB)

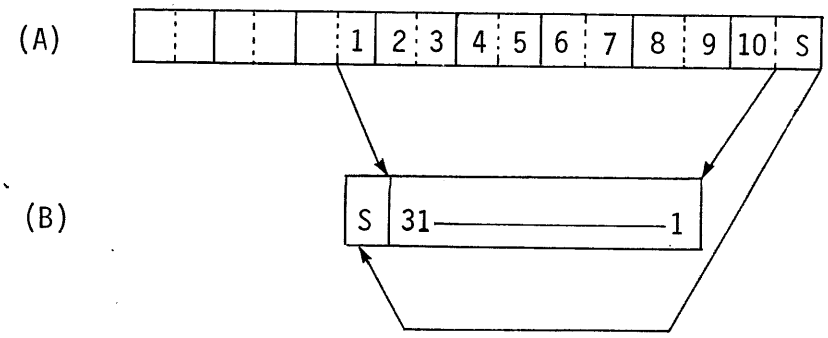
COMMAND CODE: 115 (73) (F3)

Command Description

A packed, signed, decimal number is converted to a signed, binary number.

(A) is an 8-byte double word containing ten packed, signed, decimal digits, right justified, with the sign in the least significant digit position. (B) is a 4-byte word which contains, after command execution, the signed, binary equivalent of (A), right justified. The converted binary field is zero filled to the left of the last significant digit. IF (A) is a negative decimal number, (B) is stored in two's complement.

The following illustration demonstrates, graphically, the decimal to binary conversion.



Signed, 8-byte (ten packed decimal digits plus sign digit) decimal number

Signed, 4-byte (31 bits plus sign bit) binary number

If the magnitude of (A) is greater than can be accommodated in 31 bits of (B), the overflow flag (OF) is turned ON and indeterminate results are stored in (B).

If any of the digits in (A) are non-decimal, indeterminate results are stored in (B).

If the addresses specified by A and B are not zero modulo four (0 mod 4), a programming error (PE) results, with (B) remaining unchanged.

T is not used.

B = B

NOTE

A decimal zero, whether positive or negative, is converted to a positive binary zero.

FLOATING POINT COMMANDS

The floating point hardware feature for the NCR Century 251 and 300 systems automatically scales the numbers involved in a computation and maintains the precision of the result of the computation. The feature comprises 12 commands that provide for addition, subtraction, comparison, multiplication, multiplication-addition, and division. For detailed descriptions of the floating point feature, refer to the "NCR Century 251 Processor" and "NCR Century 300 Processor" publications under this tab.

FLOATING POINT ADD SINGLE (FADD)

COMMAND CODE: 116 (74) (F4)

Command Description

(A) is added to (B), with the result replacing (B). Each field is considered to contain a single precision number (signed fraction and signed characteristic) An intermediate sum, derived by adding the two fractions, is normalized and truncated to yield the final sum.

In performing the operation, the characteristics of the two operands are compared. If the difference is less than seven, the fraction with the smaller characteristic is shifted right the number of hexadecimal digit positions specified by that difference. The fractions are then added algebraically to form the intermediate sum. This includes checking the signs and performing complementary addition (subtraction) if the signs are unlike. If an overflow results from the addition of the fractions, the intermediate sum is shifted right one digit position to accommodate the carry, and the characteristic is increased by one. If this increase causes a characteristic overflow, a programming error (PE) results with the erroneous information stored in (B) and bits 8 and 1 of memory location 036 are turned ON.

If the difference between the characteristics of the two operands is seven or greater, the fraction with the larger characteristic becomes the intermediate sum. (If the fraction with the smaller characteristic were shifted to the right seven or more positions, the vacated positions would be completely zero filled and, when added to the fraction with the larger characteristic, would yield the same result.)

Before addition, a fraction which has been shifted right can be considered to contain seven hexadecimal digits. The rightmost (seventh) digit is a guard digit and it participates in the addition. If no shift occurs, a guard digit of zero is assumed. The resulting intermediate sum of the fractions also consists of seven hexadecimal digits, including the possible overflow carry. If these digits are all zero, a true zero (32 zero bits) is generated for the final result (sign, characteristic, and fraction are all zeros).

If a true zero is not generated, the intermediate sum of the fractions, including the guard digit, is shifted left as necessary to form a normalized fraction. Vacated low-order (rightmost) digit positions are filled with zeros

and the characteristic is reduced by the number of shifts. If this normalization causes the characteristic to underflow, a true zero is generated (32 zero bits), and bit 7 of memory location 036 is turned ON.

The intermediate sum is truncated to the proper length (rightmost digit removed) and the result with its characteristic (32 bits altogether) is stored in (B).

The sign of the sum is determined algebraically, except when a true zero results or is generated, in which case the sign is always positive.

One of the L, E, or G flags is turned ON to reflect the relationship of the result to zero: less than, equal to, or greater than.

T is not used.

B = B

FLOATING POINT ADD DOUBLE (FADDD)

COMMAND CODE: 117 (75) (F5)

Command Description

The FLOATING POINT ADD DOUBLE command functions the same as the FLOATING POINT ADD SINGLE command, except there is no guard digit in the intermediate sum and the fields are double precision.

FLOATING POINT SUBTRACT SINGLE (FSUB)

COMMAND CODE: 118 (76) (F6)

Command Description

(A) is subtracted from (B), with the normalized result replacing (B). Each field is considered to contain a single precision number (signed fraction and signed characteristic). An intermediate difference, derived by subtracting one fraction from the other, is normalized and truncated to yield the final difference.

In performing the operation, first the sign of the A operand is inverted, then the characteristics of the two operands are compared. If the difference is less than seven, the fraction with the smaller characteristic is shifted right the number of hexadecimal digit positions specified by that difference. The fractions are then added algebraically to form an intermediate sum. If an overflow results from the addition of the fractions, the intermediate sum is shifted right one digit position to accommodate the carry, and the characteristic is increased by one. If this increase causes a characteristic overflow, a programming error (PE) results with the erroneous information stored in (B) and bits 8 and 1 of memory location 036 are turned ON.

If the difference between the characteristics of the two operands is seven or greater, the fraction with the larger characteristic becomes the intermediate sum. (If the fraction with the smaller characteristic were shifted to the right seven or more positions, the vacated positions would be completely zero filled and when added to the fraction with the larger characteristic, would yield the same result.)

Before addition, a fraction which has been shifted right can be considered to contain seven hexadecimal digits. The rightmost (seventh) digit is a guard digit and it participates in the addition. If no shift occurs, a guard digit of zero is assumed. The resulting intermediate sum of the fractions, after possible re-complementing, also consists of seven hexadecimal digits, including the possible overflow carry. If these digits are all zero, a true zero (32 zero bits) is generated for the final result (sign, characteristic, and fraction are all zeros).

If a true zero is not generated, the intermediate sum of the fractions, including the guard digit, is shifted left as necessary to form a normalized fraction. Vacated low-order (rightmost) digit positions are filled with zeros and the characteristic is reduced by the number of shifts. If this normalization causes the characteristic to underflow, a true zero (32 zero bits) is generated, and bits 7 and 1 of memory location 036 are turned ON.

The intermediate sum is truncated to the proper length (rightmost digit removed) and the result with its characteristic (32 bits altogether) is stored in (B).

The sign of the difference is determined algebraically, except when a true zero results or is generated, in which case the sign is always positive.

One of the L, E, or G flags is turned ON to reflect the relationship of the result to zero: less than, equal to, or greater than.

T is not used.

B = B

FLOATING POINT SUBTRACT DOUBLE (FSUBD)

COMMAND CODE: 119 (77) (F7)

Command Description

The FLOATING POINT SUBTRACT DOUBLE command functions the same as the FLOATING POINT SUBTRACT SINGLE command, except there is no guard digit in the intermediate sum and the fields are double precision.

FLOATING POINT MULTIPLY AND ADD SINGLE (FMADD)

COMMAND CODE: 120 (78) (F8)

Command Description

(A) is multiplied by (B). The normalized product (32 bits) is stored in the

memory accumulator at locations 328 through 331. The product is then added to the contents of memory accumulator locations 320 through 323.

With the exception noted above, this command functions as the sequential execution of the FLOATING POINT MULTIPLY SINGLE and the FLOATING POINT ADD SINGLE. Refer to the descriptions of these commands for a detailed explanation.

T is not used.

B = 320

FLOATING POINT MULTIPLY AND ADD DOUBLE (FMADDD)

COMMAND CODE: 121 (79) (F9)

Command Description

(A) is multiplied by (B). The normalized product (64 bits) is stored in the memory accumulator at locations 328 through 335. The product is then added to the contents of memory accumulator locations 320 through 327.

With the exceptions noted above, this command functions as the sequential execution of the FLOATING POINT MULTIPLY DOUBLE and the FLOATING POINT ADD DOUBLE. Refer to the description of these commands for a detailed explanation.

T is not used.

B = 320

FLOATING POINT COMPARE SINGLE (FCOMP)

COMMAND CODE: 122 (7A) (FA)

Command Description

(A) is compared to (B). One of the L, E, or G flags is turned ON to reflect the relationship of (A) to (B). The comparison is performed algebraically, taking into account the sign, the characteristic, and the fraction of each number. The guard digit is included in the comparison.

Comparison is accomplished by following the rules for floating point subtraction. Whenever the subtraction results in, or generates a true zero, the operands are equal.

Neither operand is changed as a result of the operation. A characteristic overflow cannot occur.

T is not used.

B = B

FLOATING POINT COMPARE DOUBLE (FCOMPD)

COMMAND CODE: 123 (7B) (FB)

Command Description

The FLOATING POINT COMPARE DOUBLE command functions the same as the FLOATING POINT COMPARE SINGLE command, except there is no guard digit and the fields are double precision.

T is not used.

B = B

FLOATING POINT MULTIPLY SINGLE (FMULT)

COMMAND CODE: 124 (7C) (FC)

Command Description

(A) is multiplied by (B). The normalized product (32 bits) is stored in the memory accumulator at locations 320 through 323. The sign of the product is determined algebraically.

If either operand contains an all zero fraction, the product will consist of a true zero (32 zero bits).

Before performing the multiplication, the operands are normalized, if necessary, without changing the configuration of the operands in their original memory locations.

The multiplication is performed by multiplying the fractions and computing the value of the characteristic. The value of the characteristic is computed by adding the A operand characteristic to the B operand characteristic and subtracting a hexadecimal 40. The result of this operation yields an intermediate product. The intermediate product is tested to see if normalization is required before storing it as the final product in the memory accumulator.

If the final product characteristic exceeds hexadecimal 7F, a programming error (PE) results (with bit 8 of memory location 036 turned ON), and memory accumulator locations 320 through 323 contain an unpredictable result. If the intermediate product characteristic exceeds hexadecimal 7F, but is brought within range by normalization, a PE does not occur.

If computing the characteristic value in the intermediate product, or normalizing the fractions to yield the final product produces a characteristic value less than a hexadecimal 00, characteristic underflow occurs and a true zero (32 zero bits) is generated. The true zero is stored in the memory accumulator as the final product, and a programming error (PE) results (with bit 7 of memory location 036 turned ON).

T is not used.

B = 320

FLOATING POINT MULTIPLY DOUBLE (FMULTD)

COMMAND CODE: 125 (7D) (FD)

Command Description

The FLOATING POINT MULTIPLY DOUBLE command functions the same as the FLOATING POINT MULTIPLY SINGLE command, except the fields are double precision and the product is stored in memory accumulator locations 320 through 327 (as opposed to 320 through 323 in the single precision).

FLOATING POINT DIVIDE SINGLE (FDIV)

COMMAND CODE: 126 (7E) (FE)

Command Description

(B), the dividend, is divided by (A), the divisor. The normalized result (quotient) is stored in memory locations 320 through 323. The remainder is not preserved. The sign of the quotient is determined algebraically.

If the divisor contains an all zero fraction, a programming error (PE) results (with bit 6 of memory location 036 turned ON). The accumulator remains undisturbed.

If the dividend contains an all zero fraction, a true zero (32 zero bits) is generated and stored in the accumulator, without causing a PE.

Before performing the division, the operands are normalized, if necessary without changing the configuration of the operands in their original memory locations.

The division is performed by dividing the fractions and computing the value of the characteristic. The value of the characteristic is computed by subtracting the divisor characteristic from the dividend characteristic and adding a hexadecimal 40. The result of this operation yields an intermediate quotient. The intermediate quotient is tested to see if normalization is required before storing it as the final quotient in the memory accumulator.

If the final quotient characteristic exceeds hexadecimal 7F, a programming error (PE) results (with bit 8 of memory location 036 turned ON), and memory accumulator locations 320 through 323 contain an unpredictable result.

If the final quotient characteristic is less than a hexadecimal 00, a true zero (32 zero bits) is generated, and a programming error (PE) results (with bit 7 of memory location 036 turned ON).

T is not used.

B = 320

FLOATING POINT DIVIDE DOUBLE (FDIVD)

COMMAND CODE: 127 (7F) (FF)

Command Description

The FLOATING POINT DIVIDE DOUBLE command functions the same as the FLOATING POINT DIVIDE SINGLE command, except the fields are double precision and the quotient is stored in memory accumulator at locations 320 through 327 (as opposed to 320 through 323 in the single precision).

DECIMAL ARITHMETIC COMMANDS

The decimal arithmetic commands fall into two major categories: Signed decimal commands and unsigned decimal commands. Each is discussed in more detail below.

Signed decimal commands provide for decimally adding, subtracting, multiplying, dividing, and comparing signed, packed BCD data, as well as converting unpacked data (one BCD character with zone bits per byte) to packed data (two BCD characters without zone bits per byte) and vice versa.

The rightmost (least significant) four-bit digit position of the rightmost (least significant) byte of a signed, packed BCD field contains the sign. If the sign is negative (1101), the field is processed as negative; if the configuration of the sign bits is anything other than negative, the field is processed a positive. However, (1011) should be used for the positive sign.

Unsigned decimal commands provide for decimally adding and subtracting unsigned, unpacked BCD data. Only the rightmost (least significant) four bits of each byte are functional. The leftmost (most significant) four bits are nonfunctional. The results of the arithmetic operation, however, contain the proper zone bits in the leftmost (most significant) four bits necessary for ASCII representation.

NOTE

> A negative, unsigned BCD field may be represented in the 10's complement form. This makes the field functionally negative, although processed by the ALP as unsigned and positive.

The command execution time calculations for decimal arithmetic commands are based on operands of at least five digits.

ADD (PADD)

COMMAND CODE: 64 (40) (C0)

Command Description

(A) is decimally added to (B), with the sum replacing (B). Each field is considered to contain signed, packed, decimal data. Addition is performed algebraically. If the absolute values of (A) and (B) are equal, the sign of (B) is unchanged by the operation, making it possible to obtain a negative field with the absolute value zero.

A carry beyond the leftmost character of (B) causes the overflow flag (OF) to be turned ON.

If (A) and (B) overlap, the result may be different from the result obtained if (A) and (B) do not overlap.

Operation on nondecimal information yields a predictable result with no error indication.

The T portion of the command specifies the number of bytes in each field. T may be any number from 0 through 255, with 0 being equivalent to 256.

B = B

SUBTRACT (PSUB)

COMMAND CODE: 65 (41) (C1)

Command Description

(A) is decimally subtracted from (B), with the difference replacing (B). Each field is considered to contain signed, packed, decimal data; subtraction is performed algebraically. If the absolute values of (A) and (B) are equal, the sign of (B) is unchanged by the operation, making it possible to obtain a negative field with the absolute value zero.

A carry beyond the leftmost character of (B) causes the overflow flag (OF) to be turned ON.

If (A) and (B) overlap, the result may be different from the result obtained if (A) and (B) do not overlap.

Operation on nondecimal information yields a predictable result with no error indication.

The T portion of the command specifies the number of bytes in each field. T may be any number from 0 through 255, with 0 being equivalent to 256.

B = B

COMPARE SIGNED (PCOMP)

COMMAND CODE: 69 (45) (C5)

Command Description

(A) is compared to (B). Both operands are considered to be signed, packed, decimal fields. One of the L, E, or G flags is turned ON to indicate the relationship of (A) to (B): less than, equal to, or greater than. The other two flags are turned OFF.

If the E or G flag is turned ON as a result of the comparison, the repeat indicator (RI) is turned OFF in the BCT following the command execution.

When two fields having zeros in all positions, except the sign position, are compared, the result is equal regardless of signs.

Operation on nondecimal information yields a predictable result with no error indication.

The T portion of the command specifies the number of bytes in each field. T may be any number from 0 through 255, with 0 being equal to 256.

B = B

MULTIPLY (PMULT)

COMMAND CODE: 93 (5D) (DD)

Command Description

(A) is decimally multiplied by (B). Each field is considered to contain signed, packed, decimal data. The signed, packed, decimal product is stored, right justified, in the memory accumulator at locations 320 through 335. The length of the accumulator is 16 characters (32 digits). The length of the product is equal to the sum of the lengths of the A and the B operands. Depending upon the length of the product, one, two, or three bytes preceding the most significant byte of the product within the accumulator are set to zero; that is, the processor clears the accumulator to a 0 mod 4 address boundary. The remaining high order bytes of the accumulator, if any, are undisturbed.

The sign of the product is determined algebraically.

If (A) or (B) overlap any part of the product area, the results are indeterminate.

If any digit of (A) or (B), other than the sign digit, contains a value other than 0 through 9, an indeterminate value results in the accumulator.

T is treated as containing two 3-bit values: $T_7 - T_5$ specify the length of (A), and $T_3 - T_1$ specify the length of (B). These lengths are expressed as the number of bytes per field and range in value from 0 through 7, with 0 being equivalent to 8. T_8 and T_4 must be equal to zero or a programming error ($\overline{PE}$) results.

NOTE

T, after command execution, is not equal to the length of the product; therefore, the MULTIPLY command must not be followed by a 1-address command.

$\underline{B} = 335 - T_A$

DIVIDE (PDIV)

COMMAND CODE: 113 (71) (F1)

Command Description

(B), the dividend, is decimally divided by (A), the divisor. Each field is considered to contain signed, packed, decimal data. The signed, packed, decimal quotient and remainder are stored in the memory accumulator at locations 320 through 335. The length of the accumulator is 16 characters (32 digits).

T is treated as containing one 3-bit and one 4-bit value: $T_7 - T_5$ specify the length of (A), and $T_4 - T_1$ specify the length of (B). The maximum length of the divisor is 8 bytes. When $T_A = 0$, a value of 8 bytes (15 digits and a sign) is assumed. The maximum length of the dividend is 16 bytes. When $T_B = 0$, a value of 16 bytes (31 digits and a sign) is assumed. T_8 must equal zero or a programming error (PE) results.

The quotient is stored, right justified, in the memory accumulator. The length of the quotient is equal to $(T_B - T_A)$ and must not be greater than eight. The starting address of the quotient is $336 - (T_B - T_A)$. Nondecimal information in either the dividend or the divisor results in an indeterminate quotient. The sign of the quotient is determined algebraically.

The remainder is stored in the memory accumulator, starting at address 320. The length of the remainder equals T_A. The accumulator locations between the quotient and the remainder are left unchanged. The sign of the remainder is the same as the sign of the dividend.

NOTE

> 1, 2, or 3 bytes preceding the most significant byte of the quotient may be set to zero, depending on the modulus (0 modulo 4) of the quotient length. 1, 2, or 3 bytes following the least significant digit of the remainder may be set to zero, depending on the modulus (0 modulo 4) of the remainder length.

All fields are treated as integers; that is, the decimal point is to the right of the least significant digit. If the divisor equals zero, or if the length of the quotient is greater than the value $(T_B - T_A)$, a programming error (PE) results, division is not attempted, and the memory accumulator remains unchanged.

NOTE

Fields should be defined for most efficient operation. Leading zeroes may be inserted in the dividend to cause the quotient to be the required length. By inserting in the dividend a number of leading zeroes that equals the value T_A insures that quotient overflow does not occur. This technique, however, slows down the execution of the command and causes great inefficiency. A better method is to define T_A and T_B, based on some knowledge of the data being processed. For effective hardware determination of a quotient overflow, a trial subtraction of the complete divisor from an equivalent length of the dividend is performed. The leftmost digit of the divisor aligned with the leftmost-minus-one digit of the dividend must produce a negative difference to avoid quotient overflow.

If the beginning address of (B), the dividend, is not zero modulo four (0 mod 4), a programming error (PE) results.

If any of the following conditions exist, a programming error (PE) results and division is not attempted:

$$T_8 \neq 0$$

$$T_B \leq T_A$$

$$(T_B - T_A) > 8$$

B = 320

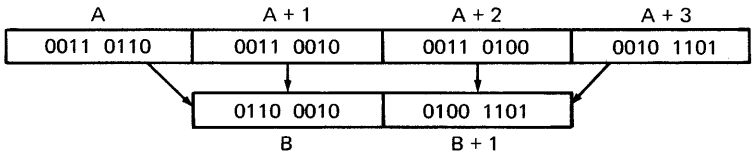
PACK (PACK)

COMMAND CODE: 76 (4C) (CC)

Command Description

(A) is packed into (B). That is, the least significant four bits (b4-b1) of each character in a pair of A characters are combined to form a single 8-bit character which is stored in one B character.

Packing is performed sequentially, from left to right, starting at the low-order A and B addresses. The four least significant bits of the left character of the A pair replace the four most significant bits of the B character; the four least significant bits of the right character of the A pair replace the four least significant bits of the B character:



Packing should be performed only on those characters shown in the shaded area on the following chart:

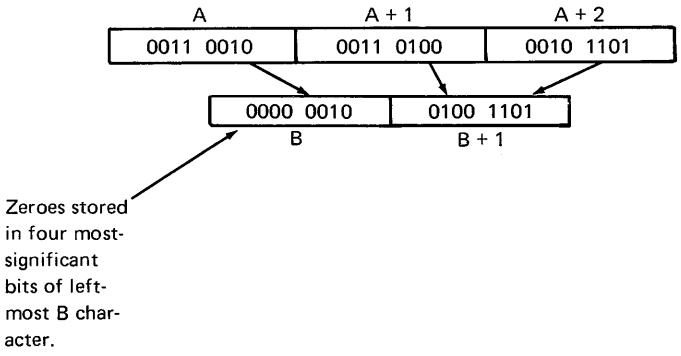
NCR CENTURY CODE CHART

B_8–B_5 \ B_4–B_1		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	NL/ LF	VT	FF	CR	SO	SI
0001	1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
0010	2	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0011	3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0100	4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0101	5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
0110	6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0111	7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

These are the only characters that can be restored to their original values by the UNPACK command. If any other characters are packed, they lose their original values.

The A field retains its original contents, unless the A and B fields overlap. If these fields overlap, the results may be different from the results obtained if these fields do not overlap. The B field contains the packed data from the A field.

The T portion of the command, specifying the number of bytes to be packed (length of the A field), may be any value from 0 through 255, with 0 being equivalent to 256. The number of B characters affected is T/2 if T is even and (T +1)/2 if T is odd. If T is odd, the leftmost byte of the A field is treated as the odd character to be right-justified in the leftmost byte of the B field; that is, b4-b1 of the A-field byte is moved to b4-b1 of the B-field byte, with b8-b5 of the B-field byte set to zeros:



$\underline{B}$ = B + T/2 if T is even and not equal to zero.
$\underline{B}$ = B + (T + 1) /2 if T is odd.
$\underline{B}$ = B + 128 if T = 0.

UNPACK (UNPACK)

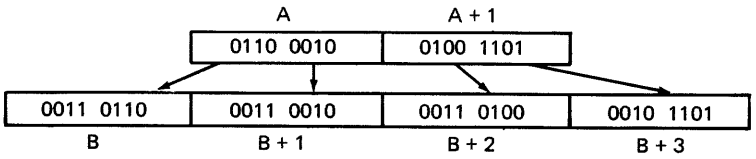
COMMAND CODE: 77 (4D) (CD)

Command Description

(A) is unpacked into (B). That is, each 8-bit A byte is divided into two 4-bit digits. Appropriate zone bits are appended to each 4-bit character forming two NCR Century characters. The two NCR Century characters are then stored in a pair of adjacent locations in the B field.

Unpacking is performed sequentially, from left to right, starting at the low-order A and B addresses. Bits b8 - b5 of each A character replace b4 - b1 of the left B character in each pair, and the appended zone bits are stored in b8 - b5 of the B characters. Bits b4 - b1 of each A character replace b4 - b1 of the right B character in each pair, the zone bits again occupying b8 - b5.

If the binary value of the 4-bit A character is nine or less, zone bits 0011 are appended; if the binary value is greater than nine, zone bits 0010 are appended:



The A field retains its original contents, unless the A and B fields overlap. If these fields overlap, the results may be different from the results obtained if these fields do not overlap. The B field contains the unpacked data from the A field.

The T portion of the command specifies the number of bytes to be unpacked (length of the A field) and may be any number from 0 through 255, with 0 being equivalent to 256. The number of B characters affected is either 2T, if $T \neq 0$, or 512, if $T = 0$.

$\underline{B} = B + 2T$ if T is not equal to 0.
$\underline{B} = B + 512$ if T is equal to 0.

ADD UNSIGNED (UADD)

COMMAND CODE: 98 (62) (E2)

Command Description

(A) is decimally added to (B), with the result replacing (B). The addition is performed from right to left.

Only b4 - b1 of each byte are added. The four most-significant bits of the byte are ignored, but the zone-bit configuration 0011 is stored in b8 - b5 of the sum.

Effectively, each A-field character is added to excess six and then added to the corresponding B-field character. If the addition of the B-field character causes a carry beyond the four bits, the result is stored in b4 - b1 of the B-field character and the carry is added to the next character position to the left. If no carry occurs, logic circuitry corrects the excess six addition and stores the four corrected bits in b4 - b1 of the corresponding B-field character. A carry beyond the leftmost position of the defined field is discarded, but the overflow flag (OF) is turned ON.

The A field retains its initial contents, unless the A and B fields overlap. If these fields overlap, the results may be different from the results obtained if these fields do not overlap.

The T portion of the command specifies the number of bytes in each field. T may be any value from 0 through 255, with 0 being equivalent to 256.

B = B

SUBTRACT UNSIGNED (USUB)

COMMAND CODE: 99 (63) (E3)

Command Description

(A) is decimally subtracted from (B), with the result replacing (B). The subtraction is performed from right to left.

Only b4 - b1 of each byte are subtracted. The four most-significant bits of each byte are ignored, but the zone-bit configuration 0011 is stored in b8 - b5 of the difference.

The 1's complement of the A-bit configuration is formed and added to the B-bit configuration to derive a partial sum. Carries from the first addition and an initial carry (to compensate for the use of the 1's complement) are added to the partial sum to derive the final sum. A carry beyond the leftmost bit position is not included in the result, but is used by the processor to determine whether the result was positive (cotents of B - contents of A) or negative (contents of A - contents of B). If there is a carry beyond the leftmost bit position, the result is positive and is stored in b4 - b1 of the corresponding B character. No carry indicates that the result is negative and the processor makes a decimal correction (in effect, subtracting excess six from the result), stores the 10's complement of (A) - (B), and turns ON the overflow indicator.

The A field retains its initial contents, unless the A and B fields overlap. If these fields overlap, the results may be different from the results obtained if these fields do not overlap.

The T portion of the command specifies the number of bytes in each field. T may be any number from 0 through 255, with 0 being equivalent to 256.

B = B

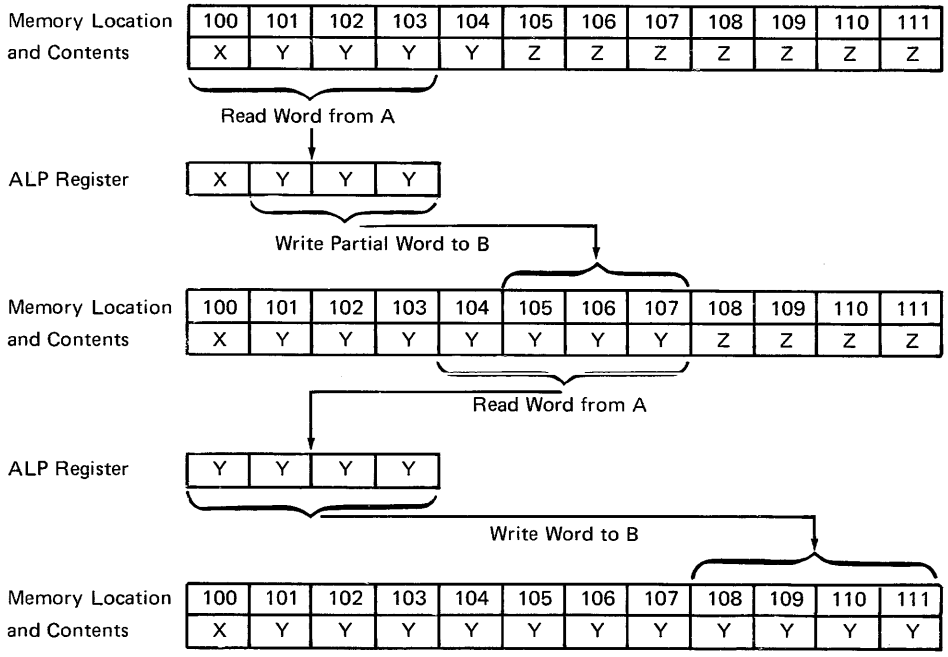
MOVE DATA COMMANDS

Move data commands are used for moving data fields from one location in memory to another. Data is moved four bytes at a time whenever the field length permits. As a general rule, if fields A and B are overlapping, the results may be different from what they would be if the two fields were not overlapping.

Overlapping of the A and B fields may be used for "spread" operations, where four bytes of the character to be spread are established in the "from" field and the "to" field is displaced from the "from" field by four bytes, in the direction of the move.

The following example illustrates how a "spread" operation is accomplished.

Command -- Move A Left to Right A = 101, B = 105, T = 7



MOVE B RIGHT TO LEFT (MVBR)

COMMAND CODE: 68 (44) (C4)

Command Description

Each byte of (B) is moved successively into the corresponding position of (A) starting with the rightmost byte.

If (A) and (B) overlap, the results may be different from the results obtained if these fields do not overlap.

The T portion of the command specifies the number of bytes in each field. T may be any number from 0 through 255, with 0 being equivalent to 256.

B = B

MOVE A LEFT TO RIGHT (MVAL)

COMMAND CODE: 84 (54) (D4)

Command Description

Each byte of (A) is moved into the corresponding position of (B), starting with the leftmost byte.

If (A) and (B) overlap the results may be different from the results obtained if these fields do not overlap.

The T portion of the command specifies the number of bytes in each field. T may be any number from 0 through 255, with 0 being equivalent to 256.

$\underline{B} = B + T$ unless $T = 0$, then $\underline{B} = B + 256$.

MOVE A RIGHT TO LEFT (MVAR)

COMMAND CODE: 100 (64) (E4)

Command Description

Each byte of (A) is moved into the corresponding position of (B), starting with the rightmost byte.

If the A and B fields overlap, the results obtained may be different from the results obtained if these fields do not overlap.

The T portion of the command specifies the number of bytes in each field. T may be any number from 0 through 255, with 0 being equivalent to 256.

$\underline{B} = B$

LOGIC COMMANDS

Logic commands are used for editing, scanning, decoding, and manipulating the data base. The EDIT command, for example, may be used to prepare data for printing according to specified formats derived from programmer-defined masks.

EDIT (EDIT)

COMMAND CODE: 73 (49) (C9)

Command Description

(B) contains the pattern into which (A) is edited. (A) is moved into (B) sequentially, one character at a time, according to the pattern stored in (B). Starting with the leftmost character of each field, the first character of (A) is moved into the first available position of B. A position is available if bit 7 of the character occupying it is equal to 1; otherwise (b7 = 0), the position is not disturbed.

When the rightmost character of (A) is an ASCII "plus," an ASCII "space" is stored in the rightmost position of B.

After execution (B) contains (A) interspersed with those characters of (B) where b7 = 0.

The T portion of the command specifies the number of bytes in the A field. T may be any number from 0 through 255, with 0 being equivalent to 256.

$\underline{B} = B + T + N$, where N is the number of characters with b7 = 0; if T = 0, then $\underline{B} = B + 256 + N$.

Command Execution Example

Before command execution, assume the values to be:

T = 6 (Length of A field)

(A) Binary	0011 0001	0011 0010	0011 0011	0011 0011	0011 0110	0010 1011
(A) ASCII	1	2	3	3	6	+

(B) Binary	0010 0100	0010 1010	0101 1000	0101 1000	0101 1000	0010 1110	0101 1000	0101 1000	0101 1000
(B) ASCII	$	*	X	X	X		X	X	X

Following command execution, the contents of the A operand has been edited according to the pattern stored in the B operand.

(B) Binary	0010 0100	0010 1010	0011 0001	0011 0010	0011 0011	0010 1110	0011 0011	0011 0110	0010 0000
(B) ASCII	$	*	1	2	3		3	6	⌀

NOTE

The last character of the B field after execution is an ASCII space (binary 0010 0000) since the last character to be edited in the A field was a plus sign (+). In the example above, with T = 6 and the number of undisturbed bytes equaling three, the B field length, therefore, is nine.

The length of B field, before execution, must be long enough to accommodate the A field plus the undisturbed bytes of B.

DECODE TO DELIMITER (DCODD)

COMMAND CODE: 78 (4E) (CE)

Command Description

The characters in the B field are replaced by characters from a table of replacement characters whose beginning address is specified by A. To locate the replacement character for the B field, the contents of the first B character is read from memory and added to the eight least significant bits of the A address. The resulting address is the location of the replacement character in the table. The replacement character is read from memory and stored in the B address replacing the original character. The fields are processed from left to right.

The command terminates when a replacement character from the A field has a one in the 8-bit position (b8 = 1), or when the B field is exhausted, whichever condition occurs first. If the command terminates because the b8 delimiter is encountered, that character is not stored in the B field and the "equal to" flag is turned ON. If the B field is exhausted before a character with the b8 delimiter is encountered in the A field table, the command terminates and the "greater than" flag is turned ON.

The address + 1 of the last character of the B field is stored in the next address index register (IR 9).

The T portion of the command specifies the number of bytes in the B field. T may be any number from 0 through 255, with 0 being equivalent to 256.

B = (IR 9)

Command Execution Example

Before command execution, assume the values to be:

T = 5 (Length of B field)

A Binary	0001 1100	0000 0000
A Hexadecimal	1 C	0 0

B Hexadecimal	2400	2401	2402	2403	2404
(B) Binary	0011 1101	0100 0001	0100 0010	0101 0100	0100 1001
(B) Hexadecimal	3 D	4 1	4 2	5 4	4 9

During command execution, the replacement character address for the B field is calculated by adding the contents of the B field to the A operand address, as shown below:

A	1 C 0 0	1 C 0 0	1 C 0 0	1 C 0 0
(B)	3 D	4 1	4 2	5 4
	1 C 3 D	1 C 4 1	1 C 4 2	1 C 5 4

Assume the contents of the replacement character addresses to be:

1C3D = 4B 1C41 = 55 1C42 = 53 1C54 = FF

Following command execution:

B Hexadecimal	2400	2401	2402	2403	2404
(B) Binary	0100 1011	0101 0101	0101 0011	0101 0100	0100 1001
(B) Hexadecimal	4 B	5 5	5 3	5 4	4 9

Since a delimiter was encountered in address 1C54 (b8 = 1), the last 2 characters of the B field are not replaced. The A operand address remains unchanged. The address + 1 of the last B character replaced is stored in IR9, the Next Address Index Register (NAIR). The "E" flag in the flag register is turned ON.

DECODE ALL (DCODA)

COMMAND CODE: 79 (4F) (CF)

Command Description

The DECODE ALL command is similar to the DECODE TO DELIMITER command. The only difference between the two operations is that the DECODE ALL causes the entire B field to be decoded. A 1 bit in b8 of a replacement character will not cause command termination.

The address B + T is stored in the next address index register (IR9), unless T = 0, in which case B + 256 is stored in IR9. The "greater than" flag is turned ON.

The T portion of the command specifies the number of bytes in the B field. T may be any number from 0 through 255, with 0 being equivalent to 256.

B = (IR9)

Command Execution Example

Before command execution, assume the values to be:

T = 6 (Length of B field)

A Binary	0001 1100	0000 0000
A Hexadecimal	1C	00

B Hexadecimal	2400	2401	2402	2403	2404	
(B) Binary	1101 0001	1100 0001	1101 0101	0100 1011	1111 0111	1111 0010
(B) Hexadecimal	D1	C1	D5	4B	F7	F2

For purposes of demonstration, assume that the contents of the B field are data read from a magnetic tape file in the EBCDIC code. By use of the DECODE ALL command, the equivalent ASCII characters from a table replace the EBCDIC characters in the B field.

During command execution, the replacement character addresses are calculated by adding the contents of the B field to the A operand address as shown below:

A Hexadecimal	1C00	1C00	1C00	1C00	1C00	1C00
(B) Hexadecimal	D1	C1	D5	4B	F7	F2
	1CD1	1CC1	1CD5	1C4B	1CF7	1CF2

Assume the contents of the replacement character addresses to be:

1CD1 = 4A 1CD5 = 4E 1CF7 = 37
1CC1 = 41 1C4B = 2E 1CF2 = 32

Following command execution:

B Hexadecimal	2400	2401	2402	2403	2404	2405
(B) Binary	0100 1010	0100 0001	0100 1110	0010 1110	0011 0111	0011 0010
(B) Hexadecimal	4A	41	4E	2E	37	32
(B) ASCII	J	A	N		7	2

SCAN ON KEY LESS THAN (SCANL)

COMMAND CODE: 85 (55) (D5)

Command Description

(A) is a 1-character field called the scan key. (B) is the field that is scanned.

The scan key is compared binarily to each character of the B field, starting with the leftmost character, until the scan key is "less than" a B field character, or until the B field is exhausted.

If as a result of the comparison, the scan key is less than the B character, the address + 1 of the B character satisfying the condition is stored in IR9. The "less than" flag is turned ON, and the repeat indicator (RI) is turned OFF in the BCT flow following the command.

If no B character satisfies the condition, the address B + T is stored in IR9, except when T = 0, in which case B + 256 is stored in IR9. The "greater than" or "equal to" flag is set ON depending on whether the scan key is greater or equal to the rightmost B character. The repeat indicator is undisturbed.

The T portion of the command specifies the number of bytes in the B field. T may be any number from 0 through 255, with 0 being equivalent to 256.

B = (IR9)

Command Execution Example

Before command execution, assume the values to be:

T = 5 (Length of B field)

A Hexadecimal	1800
(A) Binary	0011 0111
(A) Hexadecimal	3 7
(A) ASCII	7

B Hexadecimal	1820	1821	1822	1823	1824
(B) Binary	0011 0111	0011 1000	0011 1001	0011 0010	0011 0011
(B) Hexadecimal	3 7	3 8	3 9	3 2	3 3
(B) ASCII	7	8	9	2	3

Following command execution:

Since the scan key (7) in the A field is less than the contents of address 1821 (8) in the B field, the "less than" condition is satisfied. The "less than" flag is turned ON, the address + 1 (1822) of the B field character satisfying the scan condition is stored in IR 9, the Next Address Index Register (NAIR), and the repeat indicator (RI) is turned OFF following the command.

SCAN ON KEY EQUAL (SCANE)

COMMAND CODE: 86 (56) (D6)

Command Description

(A) is a 1-character field called the scan key. (B) is the field that is scanned.

The scan key is compared binarily to each character of the B field, starting with the leftmost character, until the scan key is "equal to" a B field character, or until the B field is exhausted.

If as a result of the comparison, the scan key is equal to the B character, the address +1 of the B character satisfying the condition is stored in IR 9. The "equal to" flag is turned ON, and the repeat indicator (RI) is turned OFF following the command.

If no B character satisfies the condition, the address B + T is stored in IR9, except when T = 0, in which case B + 256 is stored in IR9. The "greater than" or "less than" flag is turned ON depending on whether the scan key is greater or less than the rightmost B character. The repeat indicator is undisturbed.

The T portion of the command specifies the number of bytes in the B field. T may be any number from 0 through 255, with 0 being equivalent to 256.

B = (IR9)

Command Execution Example

Before command execution, assume the values to be:

T = 5 (Length of B field)

A Hexadecimal	1800
(A) Binary	0011 0111
(A) Hexadecimal	3 7
(A) ASCII	7

B Hexadecimal	1820	1821	1822	1823	1824
(B) Binary	0011 0111	0011 1000	0011 1001	0011 0010	0011 0011
(B) Hexadecimal	3 7	3 8	3 9	3 2	3 3
(B) ASCII	7	8	9	2	3

Following command execution:

Since the scan key (7) in the A field is equal to the contents of address 1820 (7) in the B field, the "equal to" condition is satisfied. The "equal to" flag is turned ON, the address + 1 (1821) of the B field character satisfying the scan condition is stored in IR 9, the Next Address Index Register (NAIR), and the repeat indicator (RI) is turned OFF following the command.

SCAN ON KEY GREATER THAN (SCANG)

COMMAND CODE: 87 (57) (D7)

Command Description

(A) is a 1-character field called the scan key. (B) is the field that is scanned.

The scan key is compared binarily to each character of the B field starting with the leftmost character, until the scan key is greater than the B field character, or until the B field is exhausted.

If as a result of the comparison, the scan key is greater than the B character, the address + 1 of the B character satisfying the condition is stored in IR9. The greater than flag is turned ON, and the repeat indicator (RI) is turned OFF following the command.

If no B character satisfies the condition, the address B + T is stored in IR9, except when T = 0, in chish case B + 256 is stored in IR9. The "equal to" or "less than" flag is set ON depending on whether the scan key is equal to or less than the rightmost B character. The repeat indicator is undisturbed.

The T portion of the command specifies the number of bytes in the B field. T may be any number from 0 through 255, with 0 being equivalent to 256.

B = (IR9)

Command Execution Example

Before command execution, assume the values to be:

T = 5 (Length of B field)

A Hexadecimal	1800
(A) Binary	0011 0111
(A) Hexadecimal	3 7
(A) ASCII	7

B Hexadecimal	1820	1821	1822	1823	1824
(B) Binary	0011 0111	0011 1000	0011 1001	0011 0010	0011 0011
(B) Hexadecimal	3 7	3 8	3 9	3 2	3 3
(B) ASCII	7	8	9	2	3

Following command execution:

Since the scan key (7) in the A field is greater than the contents of address 1823 (2) in the B field, the "greater than" condition is satisfied. The "greater than" flag is turned ON, the address + 1 (1824) of the B field character satisfying the scan condition is stored in IR 9, the Next Address Index Register (NAIR), and the repeat indicator (RI) is turned OFF following the command.

TABLE COMPARE (TCOMP)

COMMAND CODE: 92 (5C) (DC)

Command Description

TABLE COMPARE is a combination of the DECODE ALL and COMPARE BINARY commands.

Each character of the A field and the corresponding character of the B field are decoded according to a table in memory and the decoded characters are then compared. The original characters of A and B are left undisturbed in memory. The operation occurs sequentially from left to right, terminating at the first pair of decoded characters which are unequal or when the fields are exhausted.

The decoding is performed in the following manner:

First the address of the associated table character is computed by adding the contents of the A field to the table address in memory locations 281 through 283. The contents of this newly-formed memory address becomes the decoded A character. The same procedure is repeated for the B character.

The decoded A character is binarily compared to the decoded B character. If the decoded characters are equal, the "equal to" flag is turned ON and the command repeats the decoding and comparing with the next A and B characters. If the characters are unequal, the command terminates with the "less than" or "greater than" flag ON. If the fields are exhausted before inequality between the two characters occurs, the command terminates with the "equal to" flag ON.

If the E or G flag is turned ON, the repeat indicator (RI) is turned OFF following the command.

The T portion of the command specifies the number of bytes in each field. T may be any number from 0 through 255, with 0 being equivalent to 256.

With the E flag ON:

B = B + T except when T = 0, then B = B + 256.

With the L or G flag ON:

B = the address + 1 of the character in B causing termination.

LOGIC (LOGIC)

COMMAND CODE: 94 (5E) (DE)

Command Description

The LOGIC command uses either 1-byte or 4-byte A and B operands to perform the logic function specified by the T character of the command. The result of the operation replaces the B operand.

The logic operations, based on Boolean algebra, are performed on the individual, corresponding bit positions of the A and B operands. For example, if the T character specifies a logical sum (B = A + B), then B (bit 1) = A (bit 1) + B (bit 1), B (bit 2) = A (bit 2) + B (bit 2), etc.

The four least significant bits (b1-b4) of the T character specify one of 16 logic functions to be performed. Bit 8 of the T character specifies whether the operation is performed on one byte or four bytes. If b8 = 0, the specified logic function is performed on 1-byte operands. If b8 = 1, the specified logic function is performed on 4-byte operands. If 4-byte operands are used (b8 = 1), the beginning addresses of both operands must be zero modulo four (0 mod 4), or a programming error (PE) results. Bits 5, 6, and 7 of the T character are not used.

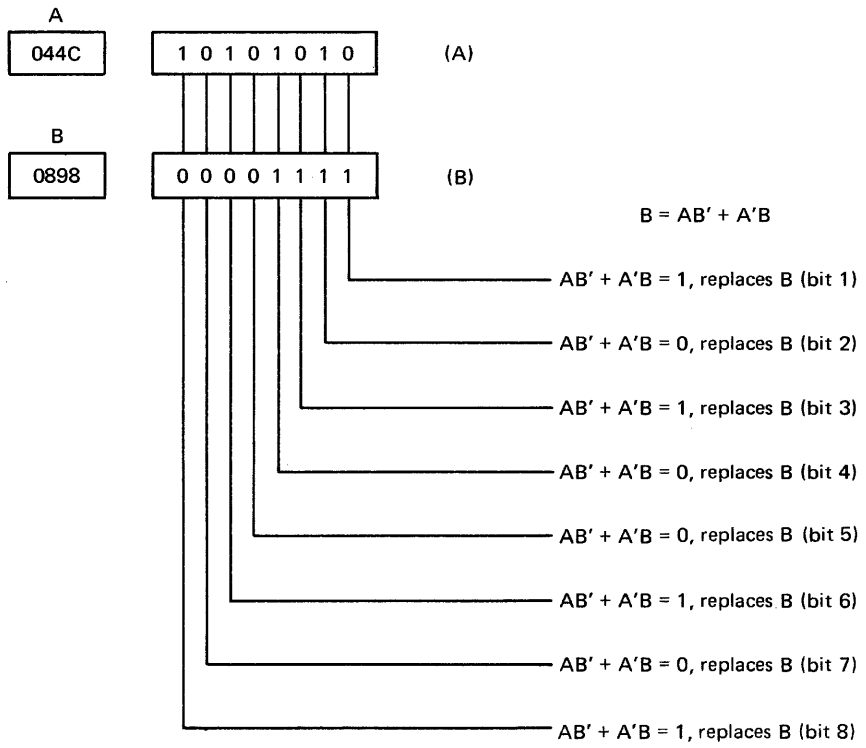
FUNCTIONS OF LOGIC COMMANDS

T (b_4-b_1) HEX VALUE	FUNCTION
0	B = 0 (clear B)
1	B = (A + B)' or A'B'
2	B = A'B
3	B = A'
4	B = AB'
5	B = B'
6	B = AB' + A'B
7	B = (AB)' or A' + B'
8	B = AB
9	B = A'B' + AB
A	B = B (store B)
B	B = A' + B
C	B = A (store A)
D	B = A + B'
E	B = A + B
F	B = 1 (set B)

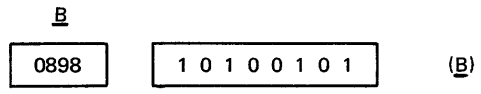
Command Execution Example

Before command execution, assume the transferred values to be:

T = 06 "EXCLUSIVE OR"



After command execution:



TRANSFER COMMANDS

Transfer commands are used for testing, branching, and counting.

During processing, if any of the transfer commands are subject to BAR/LAR addressing, the A address of the command is compared to the proper LAR before it is transferred to the control register. If A is equal to or greater than LAR, an immediate PE occurs and the control register remains unchanged. If A is less than the LAR or if LAR equals zero, the transfer is allowed.

COUNT (COUNT)

COMMAND CODE: 74 (4A) (CA)

Command Description

The 1-character binary field at memory location 64 (called the COUNT counter) is accessed, decremented by 1, and restored to memory. The resulting value is then tested for being equal to 0 (0 decremented by 1 = 255). If the result is not equal to 0, program control is transferred to A. If the result is equal to 0, the next command in sequence is executed. In either case, the repeat indicator (RI) is turned OFF. If A is not a legal command address, a PE occurs, with the control register left undisturbed.

B and T are not used.

JUMP (JUMP)

COMMAND CODE: 75 (4B) (CB)

Command Description

Program control is unconditionally transferred to A; the repeat indicator (RI) and the error indicator (EI) are turned OFF. If A is not a legal command address, a PE occurs, with the control register left undisturbed.

The address of the character following the JUMP command (the link address) is stored in the jump link register (IR8).

B and T are not used.

TEST CHARACTER EQUAL (TESTCE)

COMMAND CODE: 81 (51) (D1)

Command Description

T is compared binarily to the character designated by B.

If the compared characters are identical, program control is transferred to A, and the repeat indicator (RI) is turned OFF following the command.

If A is not a legal command address, a PE occurs, with the control register left undisturbed.

If the compared characters are not identical, the next command in sequence is executed (RI OFF) or the present command is executed (RI ON). RI is undisturbed.

The L, E, and G flags are not used.

B = B

TEST CHARACTER UNEQUAL (TESTCU)

COMMAND CODE: 82 (52) (D2)

Command Description

T is compared binarily to the character designated by B.

If the compared characters are not identical, program control is transferred to A, and the repeat indicator (RI) is turned OFF following the command. If A is not a legal command address, a PE occurs, with the control register left undisturbed.

If the compared characters are identical, the next command in sequence is executed (RI OFF) or the present command is executed (RI ON). RI is undisturbed.

The L, E, and G flags are not used.

B = B

TEST BIT (TESTB)

COMMAND CODE: 83 (53) (D3)

Command Description

B designates the character that is to be tested. The 1-bits of the T character specify which of the corresponding bits are to be tested.

If all the 1-bits of T are matched by 1-bits of the B character, or if there are no 1-bits in T, program control is transferred to A and the repeat indicator is turned OFF following the command. If A is not a legal command address, a PE occurs with the control register left undisturbed.

If any of the 1-bits of T are matched by 0-bits in the B character, the next command in sequence is executed (RI OFF) or the present command is executed (RI ON). RI is undisturbed.

B = B

BRANCH COMMANDS

COMMAND CODES:

BRANCH OVERFLOW	(BROV) 104 (68) (E8) Branch if the OF flag is ON.
BRANCH LESS	(BRL) 105 (69) (E9) Branch if the L flag is ON.
BRANCH EQUAL	(BRE) 106 (6A) (EA) Branch if the E flag is ON.
BRANCH LESS OR EQUAL	(BRLE) 107 (6B) (EB) Branch if the L or E flag is ON.
BRANCH GREATER	(BRG) 108 (6C) (EC) Branch if the G flag is ON.
BRANCH LESS OR GREATER	(BRU) 109 (6D) (ED) Branch if the L or G flag is ON.
BRANCH GREATER OR EQUAL	(BRGE) 110 (6E) (EE) Branch if the G or E flag is ON.
BRANCH UNCONDITIONALLY	(BR) 111 (6F) (EF)

Command Description

Each branch command (except BRANCH UNCONDITIONALLY) tests its appropriate flag (G, L, E, or OF) and turns OFF the repeat indicator (RI). If the tested flag is OFF, the next command in sequence is performed. The BRANCH UNCONDITIONALLY command turns OFF the repeat indicator and takes the branch in all cases.

If the tested flag is ON, or a BRANCH UNCONDITIONALLY command is specified, program control is transferred to the command specified by the effective A address. If this is not a legal command address, a PE results, with the control register left undisturbed.

When the overflow branch is taken, the OF flag is turned OFF.

The T and B portions of the branch commands are not used. If a 2-address command is specified, the T and B values are stored in the appropriate registers where they are available for subsequent implied T and B operations.

SPECIAL COMMANDS

Special commands are used by the ALP to take special action such as: restore processor conditions (control register, address registers, etc.) following a trap routine, set up trace and monitor registers, load the BAR/LAR register in multiprogramming, repeat a command, load the priority register, select a peripheral, or enter the wait state. Included in the special commands are all previleged commands which can be executed only when the processor is in the supervisor state.

SET IP ON (IPON)

COMMAND CODE: 70 (46) (C6)

Command Description

The Supervisor (S) flag is tested. The state (ON or OFF) of the S-flag determines the subsequent steps taken in the command execution flow.

If the S-flag is ON, the interrupt permit (IP) is turned ON, the repeat indicator (RI) is turned OFF, and program control is transferred to A. If A is not a legal command address, a PE results, with the control register left undisturbed.

If the S-flag is OFF, the pseudo interrupt indicator (II), located in bit 8 of memory address 272, is tested. If the pseudo II is ON, it is set OFF and the Command Code Trap is entered, with bit 4 of memory location 036 turned ON. If the pseudo II is OFF, the pseudo IP, located in bit 1 of memory address 272, is turned ON, the RI is turned OFF, and program control is transferred to A. If A is not a legal command address, a PE results, with the control register left undisturbed.

If the S-flag is ON upon termination, the tests for Trace Trapping and Interrupt Trapping are bypassed. If the S-flag is OFF, the tests are performed.

If this command is subject to BAR/LAR addressing, the A address is compared to the proper LAR before it is transferred to the control register. If A is equal to or greater than LAR, a PE results, with the control register left undisturbed. If A is less than LAR or if LAR equals zero, the transfer is allowed.

T and B are not used.

SET IP OFF (IPOFF)

COMMAND CODE: 71 (47) (C7)

Command Description

The Supervisor (S) flag is tested. The state (ON or OFF) of the S-flag determines the subsequent steps taken in the command execution flow.

If the S-flag is ON, the interrupt permit (IP) and the repeat indicator (RI) are turned OFF. Program control is transferred to A. If A is not a legal command address, a PE results, with the control register left undisturbed.

If the S-flag is OFF, the pseudo IP, located in bit 1 of memory address 272, is turned OFF, the RI is turned OFF, and program control is transferred to A. If A is not a legal command address, a PE results with the control register left undisturbed.

If this command is subject to BAR/LAR addressing, the A address is compared to the proper LAR before it is transferred to the control register. If A is equal to or greater than LAR, a PE results, with the control register left undisturbed. If A is less than LAR or if LAR equals zero, the transfer is allowed.

T and B are not used.

RESTORE (RESTOR)

COMMAND CODE: 72 (48) (C8)

Command Description

(A) is an 8-character field, called a status word, which sets up certain values that establish or reestablish a desired program state.

A must be equal to 0 modulo 4 or a PE results.

Correspondence between memory locations and the values affected by (A) is shown below.

USE OF A-FIELD BY RESTORE COMMAND

LOCATION	VALUES
A	T
A + 1, 2, AND 3	LOW ORDER 24 BITS OF B OPERAND ADDRESS
A + 4	FLAGS AND INDICATORS, AS FOLLOWS:
b8	S-FLAG
b7	NOT SPECIFIED AND WILL BE 0.
b6	OF FLAG
b5	REPEAT INDICATOR
b4	TRACE PERMIT
b3	G FLAG
b2	E FLAG
b1	L FLAG
A + 5, 6, AND 7	LOW ORDER 24 BITS OF CONTROL REGISTER

* If the RESTORE command is executed in the user state (S-flag OFF), bit 8 of the (A + 4) character is ignored. All references to memory during the execution of the RESTORE command are according to the original setting of the S-flag.

After the B and T values are established, the contents of A + 5, 6, and 7 are checked to ensure that a legal command address would be loaded into the control register; a violation causes a PE, with the control register left undisturbed.

The RESTORE command bypasses testing the repeat indicator (RI) and the trace permit indicator (TP).

If control is returned to the user state, the address that is to be transferred to the control register is compared with the applicable current LAR. If the address is equal to or greater than LAR, a PE occurs, with the control register left undisturbed. If the address is less than LAR, the command proceeds normally.

T and B are not used but reflect, after command execution, the contents of A and A + 1, 2, and 3.

OPTION SWITCHES INPUT (SWIN)

COMMAND CODE: 80 (50) (D0)

Command Description

The eight bits of the character designated by A are set to reflect the state of the eight option switches. Each bit is set to 1 if the corresponding option switch is ON and 0 if the corresponding option switch is OFF.

OPTION SWITCH/BIT CORRESPONDENCE								
SWITCH	8	7	6	5	4	3	2	1
A BYTE	b8	b7	b6	b5	b4	b3	b2	b1

If the S-flag is OFF, the Command Code Trap is entered.

T and B are not used.

REPEAT (REPEAT)

COMMAND CODE: 102 (66) (E6)

Command Description

The REPEAT command sets up conditions that cause the command immediately following it to be repeated "n" times (repeat number) or skipped (if n = 0). The repeat number is the contents of a 1-character field specified by the effective A address, and may range in value from 0 through 255, with 0 = 0, in this case. Any command may follow a REPEAT command, but certain commands nullify repeat preparation.

The REPEAT command moves the repeat number into the repeat counter at memory location 032. If the repeat number is not equal to zero, the repeat indicator is turned ON and the command terminates. If the repeat number is equal to zero, the repeat indicator is not turned on, the next command in sequence is read out to determine command format (4-byte or 8-byte), and the control register is incremented to access the second command following the REPEAT command.

With a repeat number not equal to zero, the next command in sequence is set up and repeated "n" times. The repeat indicator is tested following the execution of each command, except the REPEAT and RESTORE commands. If the RI is OFF, the next command in sequence is accessed. If the RI is ON, the repeating flow is entered.

In the repeating flow, the repeat number is read from memory and compared to zero. If it equals zero, a programming error (PE) results; however, the repeat number is decremented by one and stored in the repeat counter as FF (hex) before the processor enters the PE trapping flow. If the repeat number does not equal zero, it is decremented by 1 and compared to zero. If the result is equal, the repeat indicator is turned OFF, the contents of the control register is incremented to the address of the next command. If the repeat number, after decrementation, doees not equal zero, the repeat indicator remains ON, the control register is decremented by the size of the command just executed, permitting the same command to be re-executed.

The RI is turned OFF by special conditions occurring during the execution of certain commands. The conditions that cause the RI to be turned OFF are stated in the command descriptions of the following commands:

- BINARY COMPARE
- TEST BIT
- SCAN (3 COMMANDS)
- TABLE COMPARE
- WORD COMPARE BINARY
- SIGNED COMPARE
- TEST CHARACTER EQUAL
- TEST CHARACTER UNEQUAL
- TEST AND SET LOCK

B and T are not used.

WAIT (WAITI)

COMMAND CODE: 103 (67) (E7)

Command Description

The WAIT command causes the program operation to stop functionally. Program interruption as well as input and output operations can occur.

The repeat indicator is turned OFF, and the WAIT indicator on the operator's console is lighted. The 22 bits of A are also displayed.

The compute indicator, which is set on by pressing the COMPUTE switch on the console, is tested. If it is ON, it is turned OFF, the WAIT indicator is turned OFF, and the next command in sequence is accessed. If the S-flag is OFF, the Commond Code Trap is entered.

B and T are not used.

INOUT (INOUT)

COMMAND CODE: 112 (70)

Command Description

(A) is the peripheral address field (PAF).

The INOUT command initiates the specified I/O functions by transmitting the PAF, via the appropriate trunk, to the peripheral, which selects itself according to the PAF contents.

After the processor has transmitted the last PAF character, the appropriate S2 status character is stored and the command terminates. Certain status conditions cause the command to terminate before the PAF is exhausted, and an S2 status character is stored according to the condition that caused the command termination. If the command terminates for any reason other than a selection, the peripheral is not activated nor is it in a selected state at the completion of the command.

The effective B address is the location where the S2 status character is stored. A location different from the status character location in the control word is designated so that the peripheral priority number is preserved. RI is turned OFF. If the S-flag is OFF, the Command Code Trap is entered.

T is not used.

INOUT (INOUT)

COMMAND CODE: 112 (F0)

Command Description

(A) is the Program Priority Number.

The INOUT command in the single-stage version (Q_8 = 1) loads the priority register instead of transmitting a series of PAF characters. The four least significant bits of the character addressed by A are sent to the IOC. If the S-flag is OFF, the Command Code Trap is entered.

II is set OFF.
RI is set OFF.
B is not used.
T is not used.

LOAD T (LOADT)

COMMAND CODE: 49 (31) (B1)

Command Description

The effective A address specifies a 1-byte field which is loaded into the T (Tally) register.

B and T are not used.

NOTE

Normally, this command is performed as a single-stage command. If it is performed as a double-stage command, the T character specified in the command is loaded into the tally register during command setup, but is replaced by the byte specified in the A-operand address during command execution.

LOAD BAR (LDBAR)

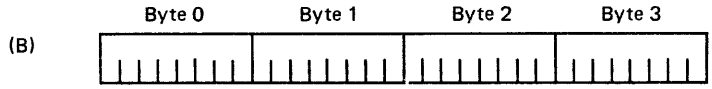
COMMAND CODE: 88 (58) (D8)

Command Description

The A operand address specifies a byte location that points to a specific BAR/LAR register into which the contents of the location specified by the B address is to be loaded. The bit configuration of (A) indicates which BAR/LAR register is to be loaded.

(A)	b_8	b	b_6	b_5	b_4	b_3	b_2	b_1	
	0	0	0	0	0	0	0	0	= BAR/LAR 0
	0	0	0	0	0	0	0	1	= BAR/LAR 1
	0	0	0	0	0	0	1	0	= BAR/LAR 2
	0	0	0	0	0	0	1	1	= BAR/LAR 3
	1	0	0	0	0	0	0	0	= BAR/LAR X

The B operand address specifies a four byte location that contains the values for BAR, LAR, WRITE PROHIBIT flag and SEGMENT UNAVAILABLE flag as follows:



Byte 0 - Bit b_8 = Segment unavailable Flag

Byte 0 - Bit b_7 = Write Prohibit Flag

Byte 0 - Bits b_3 - b_1 =
Byte 1 - Bits b_8 - b_1 = } LAR 0-3, 11 bits; LAR X, 4 bits (byte 1, bits b_{4-1}).

Byte 2 - Bits b_5 - b_1 =
Byte 3 - Bits b_8 - b = } BAR Value (14 bits)

If the B address is not equal to zero modulo four, a PE occurs with the BAR/LAR registers left undisturbed.

If the S-flag is OFF, the Command Code Trap is entered.

T is not used, but should always equal one, for compatibility purposes.

LOAD MONITOR REGISTER (LDMONR)

COMMAND CODE: 89 (59) (D9)

Command Description

The effective A address specifies a 4-character field, of which only the 22 rightmost bits are used. These 22 bits constitute an address which is entered into the Monitor Register.

If the effective A address is not 0 modulo 4, a PE results, with the monitor register left undisturbed. If the S-flag is OFF, the Command Code Trap is entered.

B and T are not used.

LOAD TRACE (LDT)

COMMAND CODE: 90 (5A) (DA)

Command Description

The character specified by the effective A address is tested. If the b4 position of the character contains a 1, the trace permit is set ON. If the b4 position of the character contains a 0, the trace permit is set OFF. In either case, the remaining 7 bits of the character are ignored.

B and T are not used.

STORE TRACE (STTR)

COMMAND CODE: 91 (5B) (DB)

Command Description

The trace permit flag is tested. If it is ON, it is set OFF and a 1 is stored in the b4 position of the character specified by the effective A address. If the trace permit flag is OFF, a 0 is stored in the b4 position of the character specified by the effective A address. In either case the remaining 7 bits of the character are set to zeros.

B and T are not used.

This command cannot be traced and the address into which the b4 indicator is stored cannot be monitored by the Monitor Address Register.

MOVE EFFECTIVE B ADDRESS (MVEB)

COMMAND CODE: 36 (24) (A4)

Command Description

The effective B address is stored in the three bytes of A. A may be 0 mod 4, 1 mod 4, 2 mod 4, or 3 mod 4 without causing a command code trap.

T is not used.

(A) = B

B = B

TEST AND SET LOCK (TESTSL)

COMMAND CODE: 50 (32) (B2)

Command Description

The effective B address designates the word that is to be tested. The word is accessed and the initial value of bit 32 (the most significant bit of the word) is stored in the ALP. Bit 32 of the word is then set ON, and the entire word is restored to memory (the memory read and restore is an uninterruptible sequence). The effective B address must be evenly divisible by four (0 mod 4) or a program error results.

The repeat indicator (RI) is set OFF and the initial value of bit 32 is tested. If bit 32 was ON, program control is transferred to the location specified by the effective A address. The effective A address must be a legal command address or a program error results leaving the original contents of the control register undisturbed. If bit 32 was OFF, the command terminates, and program control is transferred to the next command in sequence; trace and interrupt trapping are bypassed.

If this command is subject to BAR/LAR addressing, the effective A address is compared to the contents of the LAR register when bit 32 is ON. If the effective A address is smaller than the contents of the LAR register or if the LAR register contains zero, transfer of control is completed and the next command is executed. If the effective A address is greater than the contents of the LAR register, a program error results, leaving the original contents of the control register unchanged.

T is not used.

B = B

NOTE

The TEST AND SET LOCK command should not access the word to be tested from a C - 618 - 920 memory storage unit; the test portion of the command is executed, however, the state of bits 31 - 25 of the most significant byte cannot be guaranteed after the word has been restored to memory. The entire most significant byte of the word addressed by B (bits 32-25) is reserved for hardware use.