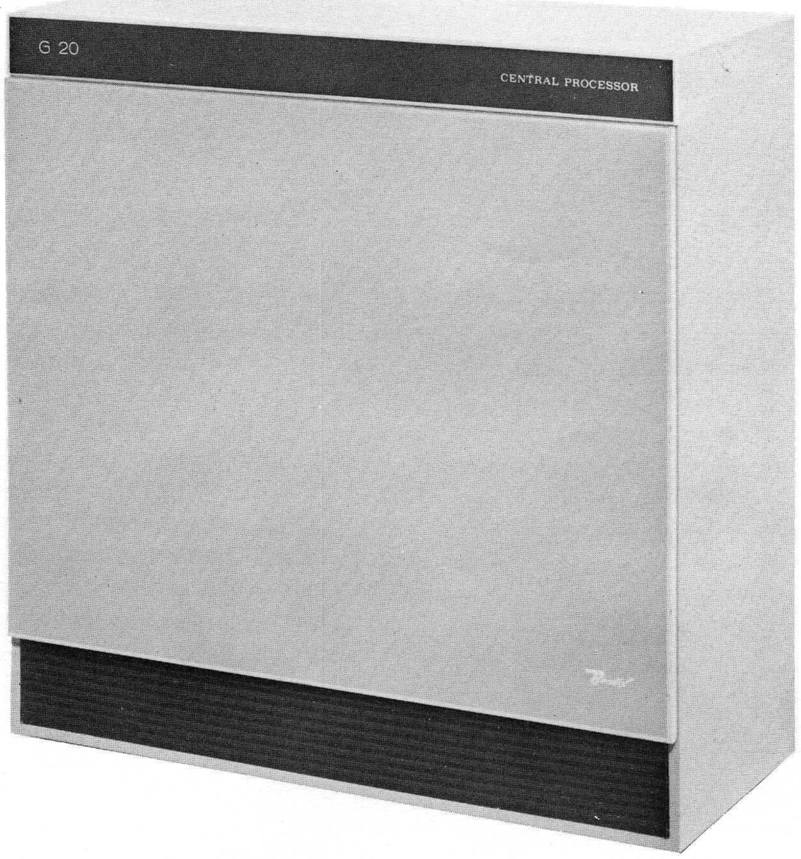
Bendix

G-20

Central Processor Machine Language



THE BENDIX G-20 CENTRAL PROCESSOR MACHINE LANGUAGE

CONTENTS

INTRODUCTION

The Bendix G-20 general purpose computing system is modular in construction. The system can be expanded in size or the nature of its operation changed when desired. The major computing and controlling module in the system is the Central Processor. It is built of solid state components and includes a random access, expandable, magnetic core memory. The system contains peripheral equipment that can operate concurrently with, and independently of, the central processor. The Central Processor may have from 4096 to 32,768 words of magnetic core memory, in modules of 4096 words each. Each word contains 33 bits, including a parity bit. 2 signs
3 flags 7 octals 2 octals

This manual describes the Central Processor's basic machine language. Descriptions of other units in the G-20 System and instructions for the use of automatic programming systems will be found in separate publications.

The bulletin, "The Bendix G-20 System for High-Speed Computing," contains a technical introduction to the G-20 System. It is suggested that this booklet be read before reading the material in this manual.

COMMAND STRUCTURE

The basic machine command specifies a single operand. Either the value of the operand, or its address, may be written in the command. By combining the contents of any number of addresses, using commands provided for this purpose, a very flexible index register operation is provided. Other special commands facilitate use of memory locations 1 through 63 as address modifiers.

NUMBER FORMS

Numerical information is stored in memory in one of three formats: Single Precision, Extended Precision and Pickapoint. A Single Precision quantity is a floating-point number that is contained in a single memory location. An Extended Precision quantity is a floating-point number that is contained in two consecutive memory locations. A Pickapoint quantity is a floating-point number that is stored in a single memory location without the exponent; during storage the quantity is automatically adjusted to an exponent selected by the programmer. All internal computation is performed in Extended Precision floating-point form. If a Single Precision or Pickapoint number is internally copied into an arithmetic register for computation, zeros are automatically appended so the number appears at Extended Precision length. Therefore, Single Precision, Pickapoint and Extended Precision numbers may be mixed during computation. The numeric result may then be stored in memory in any of the forms.

REGISTERS

There are five registers which are used to control information flow (see page 29). Special commands are used to set and reset these registers

The **Next Command** register contains the address of the next command to be executed.

The **Interrupt Request** register contains switches which are set by the computer when a request for interrupt is encountered.

The **Enable** register contains switches which are set by the programmer to permit computation to be interrupted when specific requests for interrupt occur. The first switch in the Enable register controls all of the interrupt operations and is called "Control."

The **Pickapoint Exponent** register contains the Pickapoint exponent.

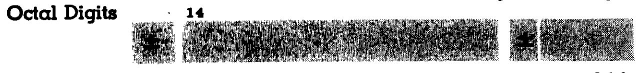
The **Line Response** register is concerned with Input-Output and its operation is discussed in the section on Input-Output.

INDEX REGISTERS

Locations 1 through 63 of the memory are special locations which can be used as Index registers. Special commands are available which increment, decrement, and test these locations (see page 28). These locations can also be used for normal storage.

ACCUMULATOR

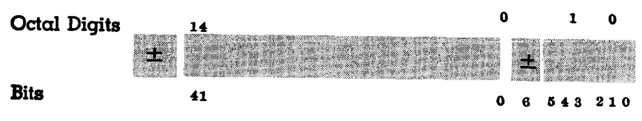
The accumulator consists of three registers. The first register will contain the 14-octal digit mantissa of any operand brought to the accumulator. The second register will contain the sign of that operand. The third register will contain the 2-octal digit exponent and the sign of the exponent.



ACCUMULATOR

OPERAND ASSEMBLY REGISTER

The Operand Assembly register is a register like the Accumulator containing a 14-octal digit mantissa with its sign and a 2-octal digit exponent with its sign.



OPERAND ASSEMBLY REGISTER

COMMAND PRESENTATION

The commands which are developed in later sections are presented in the following form:

Operation Code	Octal Designation	Function	Operation

The explanation of the command will immediately follow the heading.

Example

The Addition command is given as:

AD	045	Add	(Acc) + B → (Acc)

Notes

In internal computation, numeric information is handled in octal form. An octal number is equivalent to a binary number in which each group of these bits, from right to left, is represented by a digit from 0 to 7.

"Operand" is the name given any quantity being operated on by the "operation code" phase of a command. "Operand B" is the name given the result of the "operand select" phase of a command (see page 3).

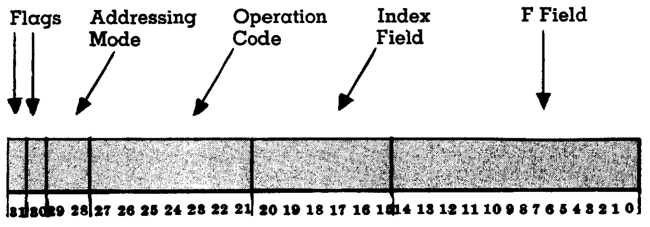
In addition to the normal use of parentheses, they are also used to denote "contents of." For example, the (Accumulator) is read "contents of the accumulator."

Brackets "[ ]" are used, instead of parentheses, where context demands, to denote separations.

Numeric subscripts are used to denote the portion of the contents of a register which are operated on by a command operation. For example, $_{31}$(Accumulator)$_0$ means that the least significant thirty-two bits of the accumulator will be operated on.

ADDRESSING FACILITIES

The G-20 Computer uses a single-operand command structure. Each command operates in two phases. The first is the "operand select" phase and the second is the "operation code" phase such as add or multiply. During the operand select phase, the addressing mode is inspected and the operand B is assembled in the Operand Assembly register, which contains a 14-octal digit mantissa with its sign and a 2-octal digit exponent with its sign.



Command Word

A command word consists of two flag bits, a 2-bit addressing mode code, an operation code, and two number fields designated "F" and "I." The F field covers the least significant fifteen bits and the I field covers the 6 bit positions 15 through 20. These designate two octal integers of 5 digits and 2 digits, respectively. The 5-octal digit F field is used as a part of the address of the operand B. The I field is the Index field. The 2-octal digits of the I field are used to address any of the Index locations, 01 through 63. Therefore, the programmer may use an Index register with all Standard Addressing (see exceptions, page 4). An Index field of zero is ignored by the addressing assembly. The contents of these two fields, taken as integers, or the contents of the locations they individually address are summed in one of four distinct ways with the contents of the Operand Assembly register to form the operand B of any command with Standard Addressing. The form of each of these four sums is controlled by the addressing mode.

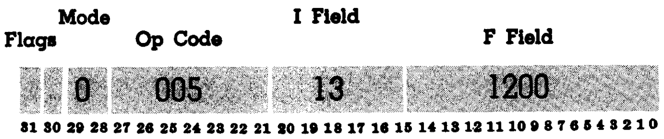
STANDARD ADDRESSING

Addressing Mode	Action
0	(OA) + F + (I) = B

In addressing mode 0, the action of the "operand select" phase of the command is to sum the F field, the contents of the location specified by the I field and the contents of the Operand Assembly register. The addition is performed in floating-point. In all modes, the I field designates an address in the range 01 through 63 and any of these locations may contain numbers in any format. Therefore, the operand B can be in any number format (see page 6).

Example

Consider a Clear and Add command (octal code 005). Using addressing mode 0, let the F field contain 1200 and the I field contain 13:



Let the contents of the Operand Assembly register be 0 and the contents of location 13 be 5.0. The operand B will be:

B = (OA) + F + (I) = 0 + 1200 + (13) = 0 + 1200 + 5.0 = 1205

Therefore, the number 1205 will be placed in the accumulator.

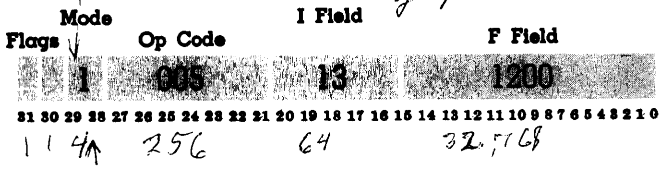
Addressing Mode	Action
1	(OA) + (F) + (I) = B

In addressing mode 1, the action of the command specifies that the F field will be taken as an integer and used as an address. Therefore, the operand B will be composed of the sum of the contents of the location specified by the F address, the contents of the location specified by the I field and the contents of the Operand Assembly register.

Example

Consider a Clear and Add command (octal code 005). Using addressing mode 1, let the F field contain 1200 and the I field contain 13:

"The result of the operations in mode "28" are an address of operand B"



"if not 29 then ", "F field is an address"

Let the contents of the Operand Assembly register be 0, the contents of location 13 be 5.0 and the contents of location 1200 be 652.0. The operand B will be:

B= (OA) + (F) + (I) = 0 + (1200) + (13) = 0 + 652.0 + 5.0 = 657.0

Therefore, the number 657.0 will be placed in the accumulator.

Addressing Mode	Action
2	((OA) + F + (I)) = B

In addressing mode 2, the "operand select" phase, using an action identical with addressing mode 0, first prepares a number which may be in floating-point. If the number first brought to the Operand Assembly register is not an integer, it is shifted to zero exponent and bits 15 through 41 are made zero. The least significant 15 bits are not disturbed. This number is then used as an address and the operand B is the contents of that address.

Example

Consider a Clear and Add command (octal code 005). Using addressing mode 2, let the F field contain 1200 and the I field contain 13:

Flags	Mode	Op Code	I Field	F Field
	2	005	13	1200

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Let the contents of the Operand Assembly register be 0, the contents of location 13 be 5.0 and the contents of location 1205 be 55. The operand B will be:

B= ((OA) + F + (I)) = (0 + 1200 + (13)) = (0 + 1200 + 5.0) = (1205) = 55.

Therefore, the number 55 will be placed in the accumulator.

Addressing Mode	Action
3	((OA) + (F) + (I)) = B

In addressing mode 3, the "operand select" phase prepares a number in a manner identical with mode 1 but then, this number, shifted to zero exponent and truncated, is used as the address of a memory location and the contents of that location are used as the operand B.

Example

Consider a Clear and Add command (octal code 005). Using addressing mode 3, let the F field contain 1200 and the I field contain 13:

Flags	Mode	Op Code	I Field	F Field
	3	005	13	1200

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Let the contents of the Operand Assembly register be 0, the contents of location 13 be 5.0, the contents of location 1200 be 652.0 and the contents of location 657 be 43.0. The operand B will be:

B = ((OA) + (F) + (I)) = (0 + (1200) + (13)) = (0 + 652.0 + 5.0) = (657) = 43.0

Therefore, the number 43.0 will be placed in the accumulator.

INDEX ADDRESSING

Reference has been made to Standard Addressing. Commands with Standard Addressing are the Numeric, Non-numeric and Test commands. There are two groups of commands with non-standard addressing. One such group is the set of Index commands (page **28**) and the set of Register commands (page **29**). Index Addressing is used with this group of commands. The "operand select" phase of the command is altered in form. The 6-bit I field is used as an identification tag, specifying which of the 63 Index register addresses is to be operated on, or on which of the five registers will the operation be performed. Therefore, the "operand select" phase of these commands operates without the "(I)" portion of the operand format.

OPERAND SELECT FOR THE INDEX AND REGISTER COMMANDS

Mode	Action
0	(OA) + F = B
1	(OA) + (F) = B
2	((OA) + F) = B
3	((OA) + (F)) = B

bit 29 still does what it seems
bit 28 still does what it seams -
I becomes part of the op code

PUTAWAY ADDRESSING

The other group of commands with non-standard addressing, is the set of Putaway or store commands. The action of these commands is to store, in a specified location, a number which is currently in the accumulator. Putaway Addressing is used with these commands. In Putaway Addressing, though the "operand select" phase still functions like the Standard Addressing operation, the final B is not the operand but specifies the location for storage of the contents of the accumulator.

the operand B address is calculated as in standard addressing but it is the store into address

Example

Consider again the example from the Standard Command explanation:

Flags	Mode	Op Code	I Field	F Field
	1	113	13	1200

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

In memory locations 13 and 1200, respectively, 5.0 and 652.0 are stored, the Operand Assembly register contains a zero and 300 is in the accumulator. Use mode 1 and a Putaway command to store the contents of the accumulator in B. Therefore:

$B = (OA) + (F) + (I) = 0 + (1200) + (13)$
$= 0 + 652.0 + 5.0 = 657.00$

The Putaway command would store 300 in memory location 00657.

FORMAT DESCRIPTION

In the G-20 a memory word consists of 32 bits of information plus a parity bit. All internal arithmetic is performed in floating-point form with the result having a signed mantissa and a signed exponent. Internally, numerical data can be of three forms: Extended Precision, Single Precision and Pickapoint.

A Single Precision quantity consists of a 7-octal digit number in the form ± NNNNNNN × $8^{\pm NN}$. An Extended Precision quantity consists of a 14-octal digit number in the form ± NNNNNNNNNNNNNN × $8^{\pm NN}$. A Pickapoint quantity consists of a 9-octal digit number in the form ± NNNNNNNNN × $8^{\pm PE}$ where $^{\pm PE}$ indicates the contents of the Pickapoint Exponent register.

The following table lists the range of values for the different number formats.

RANGE OF VALUES

	Octal Form	Decimal Equivalent
Single Precision	± 7777777 × 8^{+63} (Note 1)	± 2,000,000. × 10^{+56}
Extended Precision	± 77777777777777 × 8^{+63}	± 4,000,000,000,000. × 10^{+56}
Pickapoint	± 777777777 (Note 2)	± 130,000,000. (Note 3)
Pickapoint Integer	± 777777777	± 130,000,000.

Note 1: The exponent range is $8^{\pm 63}$ octal and $10^{\pm 56}$ decimal.

Note 2: Times the Pickapoint exponent whose range is $8^{\pm 63}$.

Note 3: Times the Pickapoint exponent whose range is $10^{\pm 56}$.

TABLE 1

The form in which information is held in memory is explicitly explained on the following pages.

When a piece of data is putaway in memory, the central processor automatically places the correct indicators in the specified location along with the data:

The precision of the number is indicated in bit number 29.

A 0 in bit number 29 indicates Single Precision or Pickapoint.

A 1 in bit number 29 indicates Extended Precision.

The sign of the mantissa, and the sign of the exponent (except Pickapoint), is placed in bit numbers 27 and 28.

A 0 indicates a positive sign.

A 1 indicates a negative sign.

When the Pickapoint switch is ON and a number is putaway:

A 1 in bit number 27 indicates a Pickapoint number.

A 0 in bit number 27 indicates a Pickapoint integer.

The way in which numerical data is brought from memory is explained on page 9, following the description of the number formats.

SINGLE PRECISION FLOATING POINT NUMBER

exponent sign

0 ± ± Exponent Mantissa

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

This number form has a 7-octal digit mantissa with a 2-octal digit exponent, and is stored in memory in the following format:

Bit positions 0 through 20 contain the mantissa

Bit positions 21 through 26 contain the exponent

Bit position 27 contains the sign of the exponent

Bit position 28 contains the sign of the mantissa

Bit position 29 contains a 0 to indicate a Single Precision number

Bit positions 30 and 31 contain interrupt flag bits

EXTENDED PRECISION NUMBER

1 ± ± Exponent Mantissa

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

least significant half

not used Mantissa

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

most significant half

This number form has a 14-octal digit mantissa with a 2-octal digit exponent. The Extended Precision number is stored in adjacent locations automatically and any "clear and add'" type command or "operand select" phase of a command operating on such a number will bring all 14 octal digits from memory. This number form is stored in memory in the following format:

Bit positions 0 through 20 of the first word contain the least significant 7 octal digits of the mantissa

Bit positions 21 through 26 of the first word contain the exponent

Bit position 27 of the first word contains the sign of the exponent

Bit position 28 of the first word contains the sign of the mantissa

Bit position 29 of the first word contains a 1 to indicate an Extended Precision number

Bit positions 30 and 31 of the first word contain interrupt flag bits

Bit positions 0 through 20 of the second word contain the most significant 7 octal digits of the mantissa

Bit positions 21 through 31 of the second word are ignored

PICKAPOINT NUMBER

0 ± 1 Mantissa

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

This number form has a 9-octal digit mantissa with a 2-octal digit exponent which is taken from the Pickapoint Exponent register. It is stored in memory in the following way:

Bit positions 0 through 26 contain the mantissa

Bit position 27 contains a 1 to indicate that the value of the exponent is taken from the Pickapoint Exponent register

Bit position 28 contains the sign of the mantissa

Bit position 29 contains a 0 to indicate a Single Precision number

Bit positions 30 and 31 contain interrupt flag bits

The signed exponent in the PE register will be brought from memory to the accumulator or Operand Assembly register with the number, if and only if, the Pickapoint Switch is ON.

PICKAPOINT INTEGER

0 ± 0 Integer

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

The Pickapoint number can be stored in memory in integer format where:

Bit positions 0 through 26 contain the integer

Bit position 27 contains a 0 to indicate that the number is an integer (The contents of the Pickapoint Exponent register is disregarded when this form is brought from memory.)

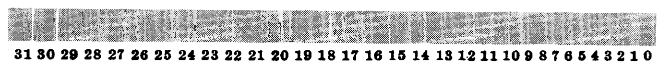
Bit position 28 contains the sign of the integer

Bit position 29 contains a 0 to indicate a Single Precision number

Bit positions 30 and 31 contain interrupt flag bits

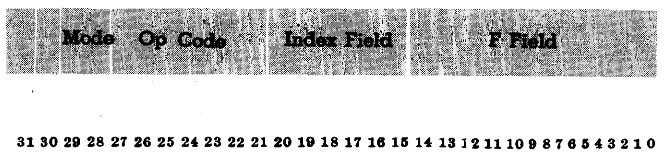
A positive zero exponent will be provided when the integer is brought from memory, if and only if, the Pickapoint Switch is ON (see Putaway, page 25).

LOGIC WORD



The Logic word is 32 bits long. Bit positions 30 and 31 are interrupt flag bits but are also part of the Logic word.

COMMAND WORD



The command word is stored in memory in the following form:

Bit positions 0 through 14 contain the F field

Bit positions 14 through 20 contain the Index field

Bit positions 21 through 27 contain the octal Operation Code

Bit positions 28 and 29 contain the Addressing Mode code

Bit positions 30 and 31 contain interrupt flag bits

ROUNDOFF RULES

A number produced by a machine operation such as multiplication, whose magnitude exceeds the physical limitations of the arithmetic unit, is rounded to avoid loss of precision due to truncation. (Division is the exception.) Roundoff will occur only when a number has been shifted right as a result of some machine operation or when a number of more than 14 octal digits is generated in the arithmetic unit.

ROUNDOFF RULES

When the magnitude of the digits to be discarded is greater than one half of the value of the first digit position to be retained, the magnitude of the digit in that position is increased by one.

When the magnitude of the digits to be discarded is exactly equal to one half of the value of the first digit position to be retained, and if that last digit is even, the magnitude of the digit is increased by one. In all other cases the digits are simply discarded.

ADD-SUBTRACT

If the exponents of the two operands in an Add or Subtract command are equal or if the exponents can be made equal by a left shift of the number with the larger exponent, no digit will be shifted out of the register, and no roundoff will be necessary.

If the exponents are not equalized after left shifting the number with the larger exponent, right shifts of the number with the smaller exponent will be executed until the exponents are equal or until the number has been shifted out of the register, at which time a true zero will replace the number. The right-shifted number will be rounded according to the Roundoff Rules.

If an Add or Subtract operation produces an overflow out of the most significant end of the arithmetic unit, a right shift will occur and the number will be rounded according to the Roundoff Rules.

MULTIPLICATION

The product generated by the multiplication command is octally normalized. If the normalized product exceeds the 14-octal digit capacity of the arithmetic unit, the remaining most significant 14 digits of the number are rounded according to the Roundoff Rules.

DIVISION

In the division process the quotient is truncated at 14 octal digits and the remainder is lost.

SINGLE PRECISION STORAGE PREPARATION

In the Single Precision, floating-point, Putaway commands, the contents of the accumulator are prepared for storage in the least significant 21 bits of the arithmetic unit. If the number in the accumulator contains more than 7 octal digits of significance, it is shifted right until only 7 octals remain. The lost digits are examined and the remaining number is rounded according to the Roundoff Rules. If the roundoff produces an overflow into the eighth least significant digit, one more shift to 7 octal digits occurs (see page 25).

In the Pickapoint Putaway command, the contents of the accumulator are prepared for storage in the least significant 27 bits of the arithmetic unit. The number in the accumulator is shifted (if necessary) until its exponent is equal to the Pickapoint exponent and then 9 octal digits are putaway. If the exponent of the contents of the accumulator cannot be made equal to the Pickapoint exponent without generating more than 9 octal digits, an interrupt request is generated

INTERRUPT FACILITIES

Internal computation can be interrupted under program control. An interrupt may be due to: exponent overflow, an illegal operation code, an illegal address, an elapse of a selected period of time, a signal from an accessory unit, a flagged command, or a flagged word of data. Interrupt requests, how to enable them and how to request them, are discussed in this section. Input-output interrupt requests are discussed in the input-output section on page **44**.

Two 15-bit registers, the Enable register and the Interrupt Request register, are used in preparing and executing interrupt requests (see Table 2 below). The commands which are used to address these registers are explained on page **29**.

INTERRUPT REGISTERS

ENABLE REGISTER	BIT POSITION	INTERRUPT REQUEST REGISTER
Command Flag 31	14	Command Flag 31
Command Flag 30	13	Command Flag 30
Logic Flag 31	12	Logic Flag 31
Logic Flag 30	11	Logic Flag 30
Data Flag 31	10	Data Flag 31
Data Flag 30	9	Data Flag 30
Pickapoint Switch	8	Overflow
Real Time Enable	7	Real Time Interrupt
	6	
Sound Tone Enable	5	
Transmit-Interrupt	4	Receive Interrupt
Transmit-Interrupt	3	Receive Interrupt
	2	Receive Interrupt
	1	Receive Interrupt
Control	0	Illegal Address

TABLE 2

The first bit in the Enable register controls the entire interrupt process. It will be referred to as "Control". It must be in the ON position (contain a 1) before any interrupt request will interrupt computer operation.

The interrupt enable switches are in bits 9 through 14 of the Enable register as follows:

The enable switch for the flag in bit position 31 of the command word is in bit 14.

The enable switch for the flag in bit position 30 of the command word is in bit 13.

The enable switch for the flag in bit position 31 of the logic word is in bit 12.

The enable switch for the flag in bit position 30 of the logic word is in bit 11.

The enable switch for the flag in bit position 31 of the data word is in bit 10.

The enable switch for the flag in bit position 30 of the data word is in bit 9.

The Pickapoint switch is in bit 8. A real time interrupt enable switch is in bit 7. Bit positions 3 and 4 in the Enable register are for input-output (see page **45**). Bit positions 1, 2 and 6 are not assigned.

The Interrupt Request register contains the indicators which are turned on by the computer when an interrupt condition is encountered as follows:

The interrupt request indicator for the flag in bit position 31 of the command word is in bit 14.

The interrupt request indicator for the flag in bit position 30 of the command word is in bit 13.

The interrupt request indicator for the flag in bit position 31 of the logic word is in bit 12.

The interrupt request indicator for the flag in bit position 30 of the logic word is in bit 11.

The interrupt request indicator for the flag in bit position 31 of the data word is in bit 10.

The interrupt request indicator for the flag in bit position 30 of the data word is in bit 9.

The real time interrupt request indicator is in bit 7.

The overflow interrupt request indicator is in bit 8.

The illegal address indicator is in bit 0. Since the occurrence of the exponent overflow and the illegal address interrupts is abortive in nature, no corresponding enable switches exist in the Enable register. Bit positions 1, 2, 3 and 4 in the Interrupt Request register are for input-output (see page **45**). Bit positions 5 and 6 are not assigned.

To cause an interrupt of computation by the interrupt flags in word positions 30 and/or 31 the program must:

1. Turn Control ON,
2. Set the desired flag, or flags, in the Enable register to a one (see page 29 for the commands), then
3. Process an operand which has a "one" in the corresponding bit position 30 and/or 31. This will set the flag bit or bits to one in the Interrupt Request register.

The previous three operations will cause the central processor to:

1. Mark the place of return to the program by placing the contents of the Next Command register in the first 15 bit positions of location 64 (a zero is loaded into bits 15 through 31).
2. Turn Control OFF and
3. Transfer to location 65 to the first command of the Interrupt Service Routine, which will inspect the Interrupt Request register to determine the interruption, report as the programmer wishes and turn the Control ON again.

If a one is encountered in a flag bit, in a command, data or logic word, but the flag is not enabled (i.e. the corresponding bit is not a one in the Enable register), the one is ignored.

If a one in an enabled flag bit is encountered in a command, data or logic word and Control is OFF, the corresponding flag bit is set to a one in the Interrupt Request register, but there is no transfer to location 65. The Next Command in sequence is executed. When Control is turned ON again, the interrupt and transfer will occur.

If Control is ON and an exponent overflow occurs in either the accumulator, the Operand Assembly register, the arithmetic unit, or, if a number in the Pickapoint mode is too large for the Pickapoint Exponent, the exponent overflow bit will be turned on in the Interrupt Request register, the contents of the Next Command register will be marked in location 64 and the Interrupt Service Routine will be entered at location 65.

If Control is OFF and an exponent overflow occurs in either the accumulator, the Operand Assembly register, the arithmetic unit, or, if a number in the Pickapoint mode is too large for the Pickapoint Exponent, the exponent overflow bit is made a one in the Interrupt Request register and the Next Command in sequence is executed.

Note

Exponent underflows (i.e. attempts to generate a number with an exponent less than -63) do not result in interrupt requests. The number is shifted right, increasing the exponent back within range, if possible. If the number is shifted entirely out of the register, it is replaced by a true zero, which has positive zero mantissa and positive zero exponent.

If Control is ON and an illegal code is processed, the illegal code bit will be made a one in the Interrupt Request register, the contents of the Next Command register will be marked in location 64 as before and the Interrupt Service Routine will be entered at location 65.

If Control is OFF and an illegal address is encountered, the results of the command being processed are not predictable. The illegal code indicator will be made a one in the Interrupt Request register and the Next Command in sequence is executed.

The real time interrupt (Bit 7 of the Interrupt Request register) is turned on once per second, if Control is ON and the clock bit (number 7) of the Enable register is a one. This interrupt is turned on and off by command.

An audio tone is turned on, if Control is ON and the fifth bit in the Enable register is made a one. The audio tone is turned off when the fifth bit of the Enable register is made a zero.

Interrupt requests are processed after the Operand Assembly register is cleared during a command operation. The set of Address Preparation commands (page 13) and the Selectively Read register to OA command (page 29) do not clear the Operand Assembly register during their operation. Therefore, no interrupt request can occur while using these commands, but if an interrupt condition is encountered, the appropriate bit in the Interrupt Request register is set. This interrupt request will be processed as soon as a command is used which clears the Operand Assembly register. A memory overflow and an exponent overflow are exceptions to the above rules about the Operand Assembly register and the commands noted.

ADDRESS PREPARATION COMMANDS

These commands are used to assemble addresses and operands in the Operand Assembly (OA) register. The Standard Addressing Modes (page 3) are used with these commands. All operands B are brought from memory in a numeric form. The Operand Assembly register is not cleared upon completion of any of these commands.

OCA 000 Clear and Add to OA $B \rightarrow (OA)$

The contents of the Operand Assembly register are replaced by the operand B. The Operand Assembly register is not cleared at the completion of this command. The contents of the accumulator are undisturbed.

This command can be used as a no-operation command if the contents of the Operand Assembly register are zero, the Index address I is zero and addressing mode 0 is used with the F field equal to zero.

OCS 020 Clear and Subtract from OA $-B \rightarrow (OA)$

The contents of the Operand Assembly register are replaced by the operand B with its sign changed. The Operand Assembly register is not cleared at the completion of this command. The contents of the accumulator are undisturbed.

OAD 040 Add in OA $(Acc) + B \rightarrow (OA)$

The operand B is added to the contents of the accumulator. The sum replaces the previous contents of the Operand Assembly register. Unnormalized floating-point addition is performed. (See page **14**). The Operand Assembly register is not cleared at the completion of this command. The contents of the accumulator are undisturbed.

OSU 060 Subtract in OA $(Acc) - B \rightarrow (OA)$

The operand B is subtracted from the contents of the accumulator. The difference replaces the previous contents of the Operand Assembly register. Unnormalized floating point-subtraction is performed. The Operand Assembly register is not cleared at the completion of this command. The contents of the accumulator are undisturbed.

OSS 100 Add and Reverse Sign in OA $-[(Acc) + B] \rightarrow (OA)$

The operand B is added to the contents of the accumulator. The sum with its sign reversed then replaces the previous contents of the Operand Assembly register. Unnormalized floating-point addition is performed. The Operand Assembly register is not cleared at the completion of this command. The contents of the accumulator are undisturbed.

ORS 120 Reverse Subtract in OA $B - (Acc) \rightarrow (OA)$

The contents of the Accumulator are subtracted from the operand B. The difference replaces the previous contents of the Operand Assembly register. Unnormalized floating-point subtraction is performed. The Operand Assembly register is not cleared at the completion of this command. The contents of the accumulator are undisturbed.

OAV 140 Add and Take Absolute Value in OA $|(Acc) + B| \rightarrow (OA)$

The operand B is added to the contents of the accumulator. The absolute value is taken and the result replaces the previous contents of the Operand Assembly register. Unnormalized floating-point addition is performed. The Operand Assembly register is not cleared at the completion of this command. The contents of the accumulator are undisturbed.

OSV 160 Subtract and Take Absolute Value in OA $|(Acc) - B| \rightarrow (OA)$

The operand B is subtracted from the contents of the accumulator. The absolute value of the difference is taken and replaces the previous contents of the Operand Assembly register. Unnormalized floating-point addition is performed. The Operand Assembly register is not cleared at the completion of this command. The contents of the accumulator are undisturbed.

ARITHMETIC PROCESSES

ADDITION

Addition is performed in an arithmetic unit of 14-octal digit capacity. The two operands involved in the addition are not normalized before the addition process begins. The exponents of the numbers are made equal and the resulting mantissas are summed, with the result having the equalized exponent. This method is called unnormalized floating-point addition.

The exponents of the two numbers are compared. If the exponents are equal, the mantissas are algebraically added together and the result is placed in the accumulator. The exponent of the operand B replaces the exponent of the accumulator. If the result of the addition contains more than 14 significant octal digits, the least significant digit is inspected, discarded and the sum is rounded according to the Roundoff Rules to 14 octal digits. These 14 digits are placed in the accumulator and the exponent of the accumulator is increased by one.

If the exponents of the two numbers are unequal, an attempt to equalize them is made by left shifting the mantissa of the number with the larger exponent. The exponent of the number is decreased by one for each left shift of one-octal digit position. The left shifts will cease when a non-zero octal digit is shifted into the most significant octal digit position (14th) of the arithmetic unit.

Example

To add 25×8^3, in the accumulator, to an operand B of 13×8^6, the operand B is left-shifted three digits before the addition. $25 \times 8^3 + 13 \times 8^6 = 25 \times 8^3 + 13000 \times 8^3 = 13025 \times 8^3$. After the addition, the accumulator will contain 13025×8^3.

If the exponents can be made equal by left shifting, the algebraic addition of the mantissas occurs when they are equal and the result is placed in the accumulator. The smaller exponent is taken as that of the result. If the result of the addition contains more than 14 significant octal digits, the least significant digit is inspected, discarded and the sum is rounded according to the Roundoff Rules to 14 octal digits. These 14 digits are placed in the accumulator and the smaller exponent, increased by one, is taken as the exponent of the result.

Example

The addition command AD is used to add the number in location 100 to the contents of the accumulator. Location 100 contains 36×8^{20} and the accumulator contains 36×8^{25}. Using mode 1 with a 0 Index field, an F field of 100, and a 0 in the Operand Assembly register, operand B will be 25×8^{20}. The contents of the accumulator are shifted to 3600000×8^{20} and then added to 25×8^{20}. The result, 360025×8^{20}, is placed in the accumulator. The contents of location 100 are undisturbed.

If the exponents cannot be equalized by left shifts, an attempt to equalize them is made by right shifting the mantissa of the number with the smaller exponent. After the left shifts of the number with the larger exponent have been accomplished, right shifting of the other number begins. The smaller exponent is increased by one for each right shift of one octal digit. The digits which are shifted out of the right end of the arithmetic unit are remembered and when the exponents are equalized these digits are inspected and the number is rounded by the Roundoff Rules.

The resulting mantissas are algebraically added together, the sum is placed in the accumulator and the number equal to the final exponents is taken as the exponent of the accumulator. If the result contains more than 14 significant octal digits, the least significant digit is inspected, discarded and the sum is rounded according to the Roundoff Rules to 14 octal digits and the exponent is increased by one.

If all of the non-zero digits are right-shifted out of the register before the exponents are equalized, the number is replaced by a true zero with a positive zero exponent and positive sign and the addition is performed without further shifting.

Example

The addition command AD (octal 045) is used to add the number in location 100 to the contents of the accumulator. Location 100 contains 257×8^{02} and the accumulator contains 234,434,-433,540 $\times 8^{05}$. Using mode 1 with a 0 Index field and a 0 in the Operand Assembly register, operand B is 257×8^{02}.

Flags	Mode	Op Code	I Field	F Field
0 0	1	045	0	100
31 30	29	28 27 26 25 24 23 22 21	20 19 18 17 16 15	14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

The contents of the accumulator are shifted to 23,443,443,354,000 $\times 8^{03}$. The exponents are not equal so the operand B is shifted right until they are equal and B is rounded to 26. $\times 8^{03}$. The addition is performed,

$$\begin{array}{r} 23{,}443{,}443{,}354{,}000. \times 8^{03} \\ \underline{00{,}000{,}000{,}000{,}026. \times 8^{03}} \\ 23{,}443{,}443{,}354{,}026. \times 8^{03}, \end{array}$$

and 23,443,443,354,026. $\times 8^{03}$
is placed in the accumulator. The contents of location 100 are undisturbed.

SUBTRACTION

Subtraction is performed exactly like addition except that the sign of the subtrahend is reversed before computation.

MULTIPLICATION

In multiplication, the multiplicand, the number in the accumulator, is octally normalized before the arithmetic operation begins. The multiplier, operand B, is not normalized or modified in any way before the multiplication begins. The multiplication begins with the least significant octal digit of the multiplier and ends when the last non-zero octal digit has been processed. An octally normalized product is formed and is placed in the accumulator. If the product contains more than 14 octal digits, the excess digits from the least significant end of the number are inspected, discarded and the product is rounded to 14 digits.

DIVISION

Upon the completion of a division operation, the accumulator contains an octally normalized, 14-digit, truncated quotient with the appropriate exponent.

ARITHMETIC COMMANDS

Operation Code	Octal Designation	Function	Operation
CA	005	Clear and Add	B→(Acc)

The contents of the accumulator are replaced by the operand B.

CS	025	Clear and Subtract	—B→(Acc

The contents of the accumulator are replaced by the operand B with its sign reversed.

AD	045	Add	(Acc)+B→(Acc)

The operand B is added to the contents of the accumulator. The sum replaces the previous contents of the accumulator. Unnormalized floating-point addition is performed (see Addition, page **14**).

SU	065	Subtract	(Acc)—B→(Acc)

The operand B is subtracted from the contents of the accumulator. The difference replaces the previous contents of the accumulator. Unnormalized floating-point subtraction is performed. (See Subtraction, page**15**).

SS 105 Add and Change Sign —[(Acc)+B]→(Acc)

The operand B is added to the contents of the accumulator. The sum with its sign reversed replaces the previous contents of the accumulator. Unnormalized floating-point addition is performed.

RS 125 Reverse Subtract B — (Acc)→(Acc)

The contents of the accumulator are subtracted from the operand B. The difference replaces the previous contents of the accumulator. Unnormalized floating-point subtraction is performed.

AV 145 Add and Take Absolute Value |(Acc)+B|→(Acc)

The operand B is added to the contents of the accumulator. The absolute value is taken and this result replaces the previous contents of the accumulator. Unnormalized floating-point addition is performed.

SV 165 Subtract and Take Absolute Value |(Acc) — B|→(Acc)

The operand B is subtracted from the contents of the accumulator. The absolute value is taken and this result replaces the previous contents of the accumulator. Unnormalized floating-point subtraction is performed.

ML 077 Multiply B*(Acc)→(Acc)

The contents of the accumulator are normalized. The normalized contents of the accumulator are multiplied by operand B. The multiplication process begins with the least significant octal digit of operand B and ends when the least non-zero digit has been processed. Floating-point multiplication is performed. The octally normalized product replaces the previous contents of the accumulator. Only the most significant 14 octal digits are retained. If the product has more than 14 significant digits, it is rounded by the Roundoff Rules.

DI 053 Divide (Acc)/B→(Acc)

The contents of the accumulator and the operand B are normalized. The normalized contents of the accumulator are divided by the normalized operand B. The quotient is a 14-octal digit number and it replaces the previous contents of the accumulator.

RD 057 Reverse Divide B/(Acc)→(Acc)

The contents of the accumulator and the operand B are normalized. The normalized operand B is divided by the normalized contents of the accumulator. The quotient is a 14-octal digit number and replaces the previous contents of the accumulator.

Shifting

The contents of the accumulator can be shifted by either multiplication or division. Multiplication is preferred because it is faster. The most common shifting situation is that of shifting a 32-bit logic word left or right by multiples of 8 bits. For example, to shift the mantissa of a number in the accumulator 8 bits to the left, multiply the number by $2^8=400_8$. The result of the multiplication will be octally normalized and will have the appropriate exponent. Any logic command (see page **18**) operating on the product causes automatic shift to zero exponent and thus returns the mantissa to the desired form.

ADD-SUBTRACT TEST COMMANDS

TP 001 Transfer on Positive — Go to (NC) if $B > 0$, Resume at $(NC)+1$ if $B \leq 0$

The operand B is tested. If B is greater than zero, the Next Command in sequence is executed. If B is less than or equal to zero, the next command in sequence is skipped and the following command is executed. The contents of the accumulator remain unchanged.

TN 021 Transfer on Negative — Go to (NC) if $B < 0$, Resume at $(NC)+1$ if $B \geq 0$

The operand B is tested. If B is less than zero, the Next Command in sequence is executed. If B is greater than or equal to zero, the Next Command in sequence is skipped and the following command is executed. The contents of the accumulator remain unchanged.

T2 041 Transfer on Sum Positive — Go to (NC) if $(Acc)+B > 0$, Resume $(NC)+1$ if $(Acc)+B \leq 0$

The operand B is added to the contents of the accumulator and the sum is tested. If (Acc)+B is greater than zero, the Next Command in sequence is executed. If (Acc)+B is less than, or equal to zero, the Next Command in sequence is skipped and the following command is executed. The contents of the accumulator remain unchanged.

TG 061 Transfer on Contents of Accumulator Greater than Operand B — $(Acc) > B$, If Yes, Go to (NC), If No, Resume at $(NC)+1$

The operand B is tested against the contents of the accumulator. If the contents of the accumulator are greater than the operand B, the Next Command in sequence is executed. If the contents of the accumulator are less than or equal to the operand B, the Next Command in sequence is skipped and the following command is executed. The contents of the accumulator remain unchanged.

T1 101 Transfer on Sum Negative — Go to (NC) if $(Acc)+B < 0$ Resume at $(NC)+1$ if $(Acc)+B \geq 0$

The contents of the accumulator are added to the operand B and the sum is tested. If (Acc)+B is less than zero, the Next Command in sequence is executed. If (Acc)+B is greater than or equal to zero, the Next Command in sequence is skipped and the following command is executed. The contents of the accumulator remain unchanged.

TL 121 Transfer on Contents of Accumulator That are Less than Operand B — $(Acc) < B$ If Yes, Go to (NC), If No, Resume at $(NC)+1$

The operand B is tested against the contents of the accumulator. If the contents of the accumulator are less than the operand B, the Next Command in sequence is executed. If the contents of the accumulator are greater than or equal to the operand B, the Next Command in sequence is skipped and the following command is executed. The contents of the accumulator remain unchanged.

T3 141 Transfer on Sum Not Equal to Zero — Go to (NC) if $|(Acc)+B| > 0$, Resume at $(NC)+1$ if $|(Acc)+B| = 0$

The absolute value of the sum of the contents of the accumulator and the operand B is tested. If $|(Acc)+B|$ is greater than zero, the Next Command in sequence is executed. If $|(Acc)+B|$ is equal to zero, the Next Command in sequence is skipped and the following command is executed. A less than zero case is not possible. The contents of the accumulator remain unchanged.

TU 161 Transfer on Contents of Accumulator not equal to Operand B — $(Acc) \neq B$ if Yes, Go to (NC), If No, Resume at $(NC)+1$

The operand B is tested against the contents of the accumulator. If the contents of the accumulator are not equal to the operand B, the Next Command in sequence is executed. If the contents of the accumulator are equal to the operand B, the Next Command in sequence is skipped and the following command is executed. The contents of the accumulator remain unchanged.

NON-NUMERIC COMMANDS

These commands can handle a word of information as a code of ones and zeros instead of a numeric quantity. They can be used to handle alphanumeric characters or to set up and use bit patterns as switches or logic words.

The result of all of these commands is a 32-bit configuration of ones and zeros placed in the least significant 32 bit positions of the accumulator. The addressing modes, which are used in preparing the operand B, perform a very important function in the operation of these commands. In addressing modes 0 and 1, all operands are prepared exactly like the operand for any numeric command, including numbers stored in Extended Precision form. In addressing modes 2 and 3, the 32 bits in the specified location are placed into the least significant 32 bit positions of the Operand Assembly register. (See Logic format, page 38). Bit positions 32 through 41 in the Operand Assembly register are made zero, the exponent is cleared to zero and the sign of the Operand Assembly register is made positive. This number in the Operand Assembly register is used as the operand B.

In the following figures, the numeric subscripts denote which part of the number is operated on. In a "Unite" operation, indicated V , two operands are combined in this manner:

```
 00000000000000000000000000101101  Value in
31                               0 Accumulator

 00000000000000000000000000011001  Value of
31                               0 Cperand B
_____________________________________________

 00000000000000000000000000111101  Result in
31                               0 Accumulator
```

A "one'" appears in the result in each position that a "one" appears in either operand.

In an "Extract" operation, indicated Λ , two operands are combined in this manner:

```
 00000000000000000000000000101101  Value in
31                               0 Accumulator

 00000000000000000000000000011001  Value of
31                               0 Operand B
_____________________________________________

 00000000000000000000000000001001  Result in
31                               0 Accumulator
```

A "one" appears in the result in each position that a "one" appears in both operands.

The complement of an operand B, indicated B, is a value in which ones and zeroes have been reversed. For example:

Operand B

```
 00000000000000000000000000001001
31                               0
```

The Complement
of Operand B

```
 11111111111111111111111111110110
31                               0
```

BB 015 Logical Bring $_{31}B_0 \rightarrow {}_{31}(Acc)_0 , 0 \rightarrow {}_{41}(Acc)_{32}$

In addressing modes 0 and 1, the mantissa of the operand B is shifted until the exponent of the operand is zero. Any digits shifted out of the register are lost and the operand is truncated. The least significant 32 bits of the result replace the least significant 32 bits of the accumulator. Bits 32 through 41 of the accumulator are made zero. The sign of the operand B is taken as the sign of the accumulator.

In addressing modes 2 and 3, the 32 bits in the specified location are placed in the least significant 32 bit positions of the accumulator. Bit positions 32 through 41 are made zero, the exponent is cleared to zero and the sign of the accumulator is made positive.

Example

Consider the octal number 202 $\times 8^{+3}$ to be stored, Single Precision floating-point with both flags zero, in location 142 in memory. The number would have the following form in memory:

```
00  000  000011  00000000000000010000010
```

Let the contents of the I field and the contents of the Operand Assembly register be zero, and let the contents of the F field be 142. The command format would be:

```
      Mode                    I Field
Flags        Op Code                          F Field
 0 0  1      015              0              142
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
```

The results of the BB command operating in addressing mode number 1 would be:

$(OA) + (F) + (I) = 0 + (142) + 0 = 202 \times 8^{+3} = B$

The operand B, shifted to zero exponent or 202000 $\times 8^{+0}$, would be loaded into the accumulator with positive sign. Bit positions 32 through 41 would be made zero. The bit pattern in the least significant 32 positions of the accumulator would be:

```
00000000000000010000010000000000
31                             0
```

If the same operation code were used in mode number 2 (Logic format) the same location would be specified, but since the Logic access brings all 32 bits from memory, supplies a positive zero exponent and positive mantissa sign, the same 32 bit positions of the accumulator would contain:

```
00000000011000000000000010000010
31                             0
```

BC 035 Bring Complement $\quad {}_{31}\overline{B}_0 \rightarrow {}_{31}(Acc)_0,\ 0 \rightarrow {}_{41}(Acc)_{32}$

In addressing modes 0 and 1, the mantissa of the operand B is shifted, if necessary, until the exponent of the operand is zero. Any digits shifted out of the register are lost and the operand is truncated. The complement of the least significant 32 bits of the result replace the least significant 32 bits of the accumulator. Bits 32 through 41 of the accumulator are made zero. The sign of the uncomplemented operand B is taken as the sign of the accumulator.

In addressing modes 2 and 3, the complement of the 32 bits in the specified location is placed in the least significant 32 bit positions of the accumulator. Bit positions 32 through 41 are made zero, the exponent is cleared to zero and the sign of the accumulator is made positive.

AL 055 Add Logical $\quad {}_{31}(Acc) + B_0 \rightarrow {}_{31}(Acc)_0,\ 0 \rightarrow {}_{41}(Acc)_{32}$

In addressing modes 0 and 1, the operand B in normal numeric form is added to the contents of the accumulator using unnormalized floating-point addition. This sum is shifted until the exponent is zero. Any digits shifted out of the register are lost and the sum is truncated. The least significant 32 bits of the sum replace the least significant 32 bits of the accumulator. Bits 32 through 41 of the accumulator are made zero. The sign of the sum is taken as the sign of the accumulator.

In addressing modes 2 and 3, the 32 bits in the specified location are placed in the least significant 32 bit positions of the Operand Assembly register, the exponent is cleared to zero (no shifting), the sign is made positive and bits 32 through 41 of the Operand Assembly register are made zero. This number in the Operand Assembly register is then added to the contents of the accumulator, the sum is shifted to zero exponent as in modes 0 and 1 and the least significant 32 bits of the sum replace the least significant 32 bits of the accumulator. Bits 32 through 41 of the accumulator are made zero. The sign of the sum is taken as the sign of the accumulator.

SL 075 Subtract Logical $\quad {}_{31}(Acc) - B_0 \rightarrow {}_{31}(Acc)_0,\ 0 \rightarrow {}_{41}(Acc)_{32}$

In addressing modes 0 and 1, the operand B in normal numeric form is subtracted from the contents of the accumulator using unnormalized floating-point subtraction. This difference is shifted until the exponent is zero. Any digits shifted out of the register are lost and the difference is truncated. The least significant 32 bits of the difference replace the least significant 32 bits of the accumulator. Bits 32 through 41 of the accumulator are made zero. The sign of the difference is taken as the sign of the accumulator.

In addressing modes 2 and 3, the 32 bits in the specified location are placed in the least significant 32 bit positions of the Operand Assembly register, the exponent is cleared to zero, the sign is made positive and bits 32 through 41 of the Operand Assembly register are made zero. This number in the Operand Assembly register is then subtracted from the contents of the accumulator, the difference is shifted to zero exponent as in modes 0 and 1 and the least significant 32 bits of the sum replace the least significant 32 bits of the accumulator. Bits 32 through 41 of the accumulator are made zero. The sign of the difference is taken as the sign of the accumulator.

XX 115 Extract $_{31}(\text{Acc}) \wedge B_0 \rightarrow {_{31}(\text{Acc})_0}, 0 \rightarrow {_{41}(\text{Acc})_{32}}$

In addressing modes 0 and 1, the operand B is assembled in normal numeric form in the Operand Assembly register. Both the operand B and the contents of the accumulator are shifted to zero exponent and any digits shifted out of either are lost and the operands truncated. Bits 32 through 41 of both are set to zero. Then the least significant 32 bits of the operand B are extracted from the least significant 32 bits of the accumulator and the result is placed in the least significant 32 bits of the accumulator. Bit positions 32 through 41 of the accumulator are made zero. The previous sign and the zero exponent of the accumulator are retained. (see Extract, page **18**).

In addressing modes 2 and 3, the 32 bits in the specified location are placed in the least significant 32 bit positions of the Operand Assembly register, the exponent is cleared to zero, the sign is made positive and bits 32 through 41 of the Operand Assembly register are made zero. The contents of the Operand Assembly register are then operand B. The contents of the accumulator are shifted to zero exponent, any digits shifted out of the accumulator are lost (no roundoff) and bits 32 through 41 are made zero. The least significant 32 bits of operand B are extracted from the least significant 32 bits of the accumulator and the result is placed in the least significant 32 bits of the accumulator. Bit positions 32 through 41 of the accumulator are made zero. The previous sign and the zero exponent of the accumulator are retained.

XC 135 Extract Complement $_{31}(\text{Acc}) \wedge \overline{B}_0 \rightarrow {_{31}(\text{Acc})_0}, 0 \rightarrow {_{41}(\text{Acc})_{32}}$

In addressing modes 0 and 1, the operand B is assembled in normal numeric form in the Operand Assembly register. Both the operand B and the con tents of the accumulator are shifted to zero exponent and any digits shifted out of either are lost and the operands are truncated. Bits 32 through 41 of both are set to zero. Then, the complement of the least significant 32 bits of the operand B are extracted from the least significant 32 bits of the accumulator and the result is placed in the least significant 32 bits of the accumulator. Bit positions 32 through 41 of the accumulator are made zero. The previous sign and the zero exponent of the accumulator are retained.

In addressing modes 2 and 3, the 32 bits in the specified location are placed in the least significant 32 bit positions of the Operand Assembly register, the exponent is cleared to zero, the sign is made positive and bits 32 through 41 of the Operand Assembly register are made zero. The contents of the Operand Assembly register are then the operand B. The contents of the accumulator are shifted to zero exponent, any digits shifted out of the accumulator are lost (no roundoff) and bits 32 through 41 are made zero. The complement of the least significant 32 bits of the operand B are extracted from the least significant 32 bits of the accumulator and the result is placed in the least significant 32 bits of the accumulator. Bit positions 32 through 41 of the accumulator are made zero. The previous sign and the zero exponent of the accumulator are retained.

UU 155 Unite $_{31}(\text{Acc}) \vee B_0 \rightarrow {_{31}(\text{Acc})_0}, 0 \rightarrow {_{41}(\text{Acc})_{32}}$

In addressing modes 0 and 1, the operand B is assembled in normal numeric form in the Operand Assembly register. Both the operand B and the contents of the accumulator are shifted to zero exponent and any digits shifted out of either are lost and the operands are truncated. Bits 32 through 41 of both are set to zero. Then, the least significant 32 bits of the operand B are united with the least significant 32 bits of the accumulator and the result is placed into the least significant 32 bits of the accumulator. Bit positions 32 through 41 of the accumulator are made zero. The previous sign and the zero exponent of the accumulator are retained.

In addressing modes 2 and 3, the 32 bits in the specified location are placed in the least significant 32 bit positions of the Operand Assembly register, the exponent is cleared to zero, the sign is made positive and bits 32 through 41 of the Operand Assembly are made zero. The contents of the Operand Assembly register are then used as the operand B. The contents of the accumulator are shifted to zero exponent, any digits shifted out of the accumulator are lost (no roundoff) and bits 32 through 41 are made zero. The least significant 32 bits of the operand B are united with the least significant 32 bits of the accumulator, and the result is placed in the least significant 32 bits of the accumulator. Bit positions 32 through 41 of the accumulator are made zero. The previous sign and the zero exponent of the accumulator are retained.

UC 175 Unite Complement $\,_{31}(\mathrm{Acc}) \vee \overline{B}_0 \rightarrow \,_{31}(\mathrm{Acc})_0\,,\ 0 \rightarrow \,_{41}(\mathrm{Acc})_{32}$

In addressing modes 0 and 1, the operand B is assembled in normal numeric form in the Operand Assembly register. Both the operand B and the contents of the accumulator are shifted to zero exponent and any digits shifted out of either are lost and the operands are truncated. Bits 32 through 41 of both are set to zero. Then the complement of the least significant 32 bits of the operand B are united with the least significant 32 bits of the accumulator and the result is placed in the least significant 32 bits of the accumulator. Bit positions 32 through 41 of the accumulator are made zero. The previous sign and the zero exponent of the accumulator are retained.

In addressing modes 2 and 3, the 32 bits in the specified location are placed in the least significant 32 bit positions of the Operand Assembly register, the exponent is cleared to zero, the sign is made positive and bits 32 through 41 of the Operand Assembly register are made zero. The contents of the Operand Assembly register are then used as the operand B. The contents of the accumulator are shifted to zero exponent, any digits shifted out of the accumulator are lost (no roundoff) and bits 32 through 41 are made zero. The complement of the least significant 32 bits of the operand B are united with the least significant 32 bits of the accumulator and the result is placed in the least significant 32 bits of the accumulator. Bit positions 32 through 41 of the accumulator are made zero. The previous sign and the zero exponent of the accumulator are retained.

NON-NUMERIC TEST COMMANDS

The preparation of all B operands and accumulator contents prior to the test is identical with that of the Non-numeric Commands (page 18). The previous contents of the accumulator are left undisturbed upon completion of any test.

J0	011	Jump on Zeros	Go to (NC) if ${}_{31}B_0 = 0$ Resume at (NC)+1 if ${}_{31}B_0 \neq 0$

In addressing modes 0 and 1, the operand B is placed in the Operand Assembly register, the mantissa of the operand is shifted until the exponent of the operand is zero, any digits shifted out of the register are lost (no roundoff) and bits 32 through 41 are made zero. The contents of the Operand Assembly register are then tested. If the contents are equal to zero, the Next Command in sequence is executed. If the contents are not equal to zero, the Next Command in sequence is skipped and the following command is executed. The operand B is not retained upon completion of this command. The previous contents of the accumulator remain unchanged.

In addressing modes 2 and 3, the 32 bits in the specified location are placed in the least significant 32 bit positions of the Operand Assembly register. Bit positions 32 through 41 are made zero, the exponent is cleared to zero and the sign of the Operand Assembly register is made positive. The operation of the test with this operand B is identical with that for addressing modes 0 and 1.

J1	031	Jump on Ones	Go to (NC) if ${}_{31}\overline{B}_0 = 0$, Resume at (NC)+1 if ${}_{31}B_0 \neq 0$

In addressing modes 0 and 1, the operand B is placed in the Operand Assembly register, the mantissa of the operand is shifted until the exponent is zero, any digits shifted out of the register are lost (no roundoff) and bits 32 through 41 are made zero. The complement of the contents of the Operand Assembly register are then tested. If the complement of the contents is equal to zero, the Next Command in sequence is executed. If the complement of the contents is not equal to zero, the Next Command in sequence is skipped and the following command is executed. The operand B is not retained upon completion of this command. The previous contents of the accumulator remain unchanged.

In addressing modes 2 and 3, the 32 bits in the specified location are placed in the least significant 32 bit positions of the Operand Assembly register. Bit positions 32 through 41 are made zero, the exponent is cleared to zero and the sign of the Operand Assembly register is made positive. The operation of the test with this operand B is identical with that for addressing modes 0 and 1.

JS	051	Jump on Sum Non-zero	Go to (NC) if ${}_{31}(Acc)+B_0 \neq 0$ Resume at (NC)+1 if ${}_{31}(Acc)+B_0 = 0$

In addressing modes 0 and 1, the operand B in normal numeric form is added to the contents of the accumulator using unnormalized floating-point addition. The sum is shifted to zero exponent, any digits shifted out of the register are lost (no roundoff) and bit positions 32 through 41 are made zero. Then the modified sum is tested. If the sum is not equal to zero, the Next Command in sequence is executed. If the sum is equal to zero, the Next Command in sequence is skipped and the following command is executed. The operand B is not retained upon completion of this command. The previous contents of the accumulator remain unchanged.

In addressing modes 2 and 3, the 32 bits of the specified location are placed in the least significant 32 bit positions of the Operand Assembly register. Bit positions 32 through 41 are made zero, the exponent is cleared to zero and the sign of the Operand Assembly register is made positive. The operation of the test with this operand B is identical with that for addressing modes 0 and 1.

JG	071	Jump on Difference Non-zero	Go to (NC) if ${}_{31}(Acc) - B_0 \neq 0$, Resume at (NC)+1 if ${}_{31}(Acc) - B_0 = 0$

In addressing modes 0 and 1, the operand B in normal numeric form is subtracted from the contents of the accumulator using unnormalized floating-point subtraction. The difference is shifted to zero exponent, any digits shifted out of the register are lost (no roundoff) and bit positions 32 through 41 are made zero. Then the modified difference is tested. If the difference is not equal to zero, the Next Command in sequence is executed. If the difference is equal to zero, the Next Command in sequence is skipped and the following command is executed. The Operand B is not retained upon completion of this command. The previous contents of the accumulator remain undisturbed.

In addressing modes 2 and 3, the 32 bits of the specified location are placed in the least significant 32 bit positions of the Operand Assembly register. Bit positions 32 through 41 are made zero, the exponent is cleared to zero and the sign of the Operand Assembly register is made positive. The operation of the test with this operand B is identical with that for addressing modes 0 and 1.

JX	111	Jump on Extract	Go to (NC) if $_{31}(\mathrm{Acc})\wedge B_0 = 0$ Resume at (NC)+1 if $_{31}(\mathrm{Acc})\wedge B_0 \neq 0$

In addressing modes 0 and 1, the operand B is assembled in normal numeric form in the Operand Assembly register. Both the operand B and the contents of the accumulator are shifted to zero exponent, any digits shifted out of either are lost (no roundoff) and bits 32 through 41 of both are set to zero. The least significant 32 bits of the operand B are extracted from the least significant 32 bits of the contents of the accumulator (see Extract, page **18**). The result of the extraction is tested. If the result is zero, the Next Command in sequence is executed. If the result is not zero, the Next Command in sequence is skipped and the following command is executed. The operand B is not retained upon completion of this command. The previous contents of the accumulator remain undisturbed.

In addressing modes 2 and 3, the 32 bits of the specified location are placed in the least significant 32 bit positions of the Operand Assembly register. Bit positions 32 through 41 are made zero, the exponent is cleared to zero and the sign of the Operand Assembly register is made positive. The operation of the test with this operand B is identical with that for addressing modes 0 and 1.

XJ	131	Jump on Extract Complement	Go to (NC) if $_{31}(\mathrm{Acc})\wedge \overline{B_0} = 0$ Resume at (NC)+1 if $_{31}(\mathrm{Acc})\wedge \overline{B_0} \neq 0$

In addressing modes 0 and 1, the operand B is assembled in normal numeric form in the Operand Assembly register. Both the operand B and the contents of the accumulator are shifted to zero exponent, any digits shifted out of either are lost (no roundoff) and bits 32 through 41 of both are set to zero. The complement of the least significant 32 bits of the operand B is extracted from the least significant 32 bits of the contents of the accumulator. The result of the extraction is tested. If the result is zero, the Next Command in sequence is executed. If the result is not zero, the Next Command in sequence is skipped and the following command is executed. The operand B is not retained upon completion of this command. The previous contents of the accumulator remain undisturbed.

In addressing modes 2 and 3, the 32 bits of the specified location are placed in the least significant 32 bit positions of the Operand Assembly register. Bit positions 32 through 41 are made zero, the exponent is cleared to zero and the sign of the Operand Assembly register is made positive. The operation of the test with this operand B is identical with that for addressing modes 0 and 1.

JU	151	Jump on Unite	Go to (NC) if $_{31}(\mathrm{Acc})\vee B_0 = 0$ Resume at (NC)+1 if $_{31}(\mathrm{Acc})\vee B_0 \neq 0$

In addressing modes 0 and 1, the operand B is assembled in normal numeric form in the Operand Assembly register. Both the operand B and the contents of the accumulator are shifted to zero exponent, any digits shifted out of either are lost (no roundoff) and bits 32 through 41 of both are set to zero. The least significant 32 bits of the operand B are united with the least significant 32 bits of the contents of the accumulator. The result of the union is tested. If the result is zero, the Next Command in sequence is executed. If the result is not zero, the Next Command in sequence is skipped and the following command is executed. The operand B is not retained upon completion of this command. The previous contents of the accumulator remain undisturbed.

In addressing modes 2 and 3, the 32 bits of the specified location are placed in the least significant bit positions of the Operand Assembly register. Bit positions 32 through 41 are made zero, the exponent is cleared to zero and the sign of the Operand Assembly register is made positive. The operation of the test with this operand B is identical with that for addressing modes 0 and 1.

UJ	171	Jump on Unite Complement	Go to (NC) if ${}_{31}(\text{Acc}) \vee \overline{B}_0 = 0$ Resume at (NC)$+1$ if ${}_{31}(\text{Acc}) \vee \overline{B}_0 \neq 0$

In addressing modes 0 and 1, the operand B is assembled in normal numeric form in the Operand Assembly register. Both the operand B and the contents of the accumulator are shifted to zero exponent, any digits shifted out of either are lost (no roundoff) and bits 32 through 41 of both are set to zero. The complement of the least significant 32 bits of the operand B is united with the least significant 32 bits of the contents of the accumulator. The result of the union is tested. If the result is zero, the Next Command in sequence is executed. If the result is not zero, the Next Command in sequence is skipped and the following command is executed. The operand B is not retained upon completion of this command. The previous contents of the accumulator remain undisturbed.

In addressing modes 2 and 3, the 32 bits of the specified location are placed in the least significant bit positions of the Operand Assembly register. Bit positions 32 through 41 are made zero, the exponent is cleared to zero and the sign of the Operand Assembly register is made positive. The operation of the test with this operand B is identical with that for addressing modes 0 and 1.

PUTAWAY COMMANDS

Putaway Addressing, as explained on page 5, is used for Putaway commands. B is the location to be filled by the number in the accumulator when using Putaway commands. A number format is used in preparing the Putaway address B. Therefore, the operand select operation is the same as in the Standard command operand assembly but the final operand B is always an address. The contents of the accumulator are left undisturbed at the completion of the command. A zero stored by any numeric Putaway command will be a true zero. The storage formats are shown on page 6.

Interrupt flag bits 30 and 31 of location B are cleared to zero during the execution of all of the Putaway commands except the Putaway Logic command. To set either or both of the interrupt flags to a "one" or "zero":

The word to be placed in location B is brought to the accumulator in Logic form (see page 18 and 38),

The desired bit configuration is put in bit positions 30 and 31 of the accumulator using, for instance, a Unite command,

And the word in the accumulator is stored in location B using the Putaway Logic command which will not clear the flag bits.

PF 153 Putaway Extended Precision $(Acc) \rightarrow [(Location\ B), {}_{20}(Location\ B+1)_0]$

In the Extended Precision Putaway, the full 14-octal digit floating-point number is stored, the right half in location B, bits 0 through 20, and the left half in location B+1, bits 0 through 20. Flag bits 30 and 31 in location B are cleared. An Extended Precision tag is set in bit 29 of location B. The sign and the exponent from the accumulator are stored in location B. Bits 21 through 31 of location B+1 are left undisturbed. The previous contents of the accumulator are left undisturbed.

PH 113 Putaway Single Precision $(Acc) \rightarrow (Location\ B)$

In the Single Precision Putaway, the action depends on the state of the Pickapoint switch. When the Pickapoint switch (page 11) is OFF, the PH command stores the accumulator as a Single Precision floating-point number. The contents of the accumulator are shifted to the right (if necessary) and rounded (see Roundoff Rules, page 10) to 7 octal digits, then stored with sign, adjusted exponent and a Single Precision flag (bit 29) in location B. Flag bits 30 and 31 are made zero. If the contents of the accumulator are too large to be held as a Single Precision number, nothing is putaway and an interrupt request is generated. The contents of the accumulator are left undisturbed.

When the Pickapoint Switch is ON, the number in the accumulator is shifted (if necessary) until its exponent is equal to the contents of the PE register, then rounded to 9 octal digits. These 9 digits are stored in location B with the sign of the accumulator (except + if the magnitude is zero) and with a Single Precision flag in bit 29. Flag bits 30 and 31 are made zero. Pickapoint does not carry an exponent or exponent sign to storage but does store a negative exponent sign to differentiate between Pickapoint numbers (negative exponent sign, 1 bit in the exponent sign bit, number 27), and Pickapoint integers (positive exponent sign, 0 bit in the exponent sign bit, number 27). If the exponent of the contents of the accumulator cannot be made equal to the Pickapoint exponent without generating more than 9 octal digits, nothing is putaway and an interrupt request is generated. The contents of the accumulator are left undisturbed.

PL 173 Putaway Logic ${}_{31}(Acc)_0 \rightarrow {}_{31}(Location\ B)_0$

The contents of the accumulator are shifted to zero exponent and truncated (if necessary). Then the least significant 32 bits of the accumulator are stored in location B. The remaining bits of the accumulator are ignored. This command is the only Putaway command which does not set flag bits 30 and 31 to zero. The previous contents of the accumulator are left undisturbed upon completion of this command.

PI 133 Putaway Integer (Acc)$\rightarrow$(Location B)

In the Integer Putaway, the action depends on the state of the Pickapoint Switch. When the Pickapoint Switch is OFF, the contents of the accumulator are shifted to zero exponent and truncated (if necessary). The least significant 7 octal digits are stored in location B with a Single Precision flag in bit position 29, the same sign as the accumulator (+ if magnitude zero) and a positive zero exponent. Flag bits 30 and 31 are made zero. If there are any non-zero digits in positions of significance greater than 7 after the exponent has been adjusted to zero, they are ignored.

When the Pickapoint Switch is ON, the contents of the accumulator are shifted (if necessary) to zero exponent and truncated. The least significant 9 octal digits are stored in location B with a Single Precision flag in bit position 29, the same sign as the accumulator (+ if magnitude is zero) and a positive zero exponent. Flag bits 30 and 31 are made zero. If there are any non-zero digits in positions of significance greater than 9 after the exponent has been adjusted to zero, they are ignored.

PZ 073 Putaway Zeros $0\rightarrow{}_{31}(B)_0$

The PZ command clears all 32 bits of location B to zero.

CONTROL COMMANDS

These commands are used to transfer control from one sequence of commands to another.

The destination of the transfer is specified by the operand B, assembled in the normal way. In all four modes the "operand select" phase of the command assembles the operand B as usual. After assembly, B is shifted to zero exponent and any digits shifted out of the Operand Assembly register are lost. The least significant five octal digits are then used as the transfer location B and all other bits in the Operand Assembly register are ignored. If the location specified is negative, or if it exceeds the memory capacity, an interrupt request will occur. (See Interrupts, page 11).

Example

Consider an unconditional transfer which has an I field equal to zero, 01400 in the F field and is operating in addressing mode 1. Let 1332 x 8^{-1} be stored in location 01400. The contents of the Operand Assembly register are zero.

The destination of transfer with these conditions is prepared by the operand select in two steps. B is found as below:

$B = (OA) + (F) + (I) = 0 + (1400) + (0) = 0 + 1332 \times 8^{-1} + 0,$

Then 1332 x 8^{-1} is shifted to zero exponent and truncated to equal 133. x 8^{0}. The destination of transfer location B is then 133.

GO 017 Go to Location B $\quad {}_{14}B_0 \rightarrow (NC)$

The next command executed is the command in location B. The contents of the accumulator are not disturbed.

GE 037 Go and Enable $\quad {}_{14}B_0 \rightarrow (NC)$

The next command executed is the command in location B. Control is turned ON (see Control, page 11). The contents of the accumulator are not disturbed.

SK 137 Go to Location (NC)+B $\quad {}_{14}(NC) + B_0 \rightarrow (NC)$

The operand B is prepared in the Operand Assembly register as in a Standard command. Before shifting to zero exponent, operand B is added to the contents of the Next Command address register using floating-point addition and placed back in the Next Command address register. Operand B may be negative, but if the sum is negative, an interrupt request is generated in the illegal address indicator. Also, if the sum exceeds the memory capacity, an interrupt occurs. (i.e. bit number zero of the Interrupt Request register is set to a "one"). If the result of the addition is rounded due to an exponent shift, an interrupt request is generated in the illegal address indicator.

The Next Command executed is the command in the address specified by the sum placed in the Next Command address register. If the SK command is in location C, the next command executed is the command in location C+1+B. The contents of the accumulator are undisturbed.

MT	177	Mark Place in	$(NC) \rightarrow {}_{14}(B)_0$
		Location B	$0 \rightarrow {}_{31}(B)_{15}$
		Go to Location B+1	${}_{14}(B+1)_0 \rightarrow (NC)$

The Next Command address is placed in the least significant 15 bit positions of location B. Bit positions 15 through 31 of B are made zero. The Next Command executed is the command in location B+1.

INDEX COMMANDS

The Index commands set, increment, decrement, and test the Index registers. These commands cannot be indexed because the field of the command structure is used to specify the number of the Index register. Locations 0 through 63 can be addressed by any of the Index commands. During the "operand select" phase of the Index commands, Index Addressing is used instead of Standard Addressing (page **4**). A numeric format is used in the preparation of operand B in all four addressing modes. This format is the same as the one described for the Single Precision Integer Putaway command (page **25**).

LP 012 Load Index Positive $B \rightarrow (I)$

The operand B replaces the contents of the Index address I. The contents of the accumulator are not disturbed.

LN 032 Load Index Negative $-B \rightarrow (I)$

The operand B with its sign reversed replaces the contents of the Index address I. The contents of the accumulator are not disturbed.

IN 002 Increment Index $B + (I) \rightarrow (I)$

The Operand B is added to the contents of location I using unnormalized floating-point addition. The sum is placed back into location I as an integer. The contents of location I are brought to the arithmetic unit in numeric form. The contents of the accumulator are not disturbed.

DE 022 Decrement Index $(I) - B \rightarrow (I)$

The operand B is subtracted from the contents of location I using unnormalized floating-point subtraction. The difference is placed back into location I, as an integer. The contents of location I are brought to the arithmetic unit in numeric form. The contents of the accumulator are not disturbed.

PT 016 Load Positive and Test $B \rightarrow (I)$, If $(I) = 0$ then $NC + 1 \rightarrow (NC)$

The operand B replaces the contents of location I. If the new contents of location I are zero, the Next Command in sequence is skipped and the following command is executed. If the new contents of location I are not zero, the Next Command in sequence is executed. The contents of the accumulator are not disturbed.

NT 036 Load Negative and Test $-B \rightarrow (I)$, If $(I) = 0$ then $NC + 1 \rightarrow (NC)$

The operand B with its sign reversed replaces the contents of location I. If the new contents of location I are zero, the Next Command in sequence is skipped and the following command is executed. If the new contents of location I are not zero, the Next Command in sequence is executed. The contents of the accumulator are not disturbed.

IT 006 Increment Index and Test $(I) + B \rightarrow (I)$

The operand B is added to the contents of location I using unnormalized floating-point addition. The sum is placed back into location I as an integer. If the new contents of location I are zero, the Next Command in sequence is skipped and the following command is executed. If the new contents of location I are not zero, the Next Command in sequence is executed. The contents of the accumulator are not disturbed.

DT 026 Decrement Index and Test $(I) - B \rightarrow (I)$

The operand B is subtracted from the contents of location I using unnormalized floating-point subtraction. The difference is placed back into location I as an integer. If the new contents of location I are zero, the Next Command in sequence is skipped and the following command executed. If the new contents of location I are not zero, the Next Command in sequence is executed. The contents of the accumulator are not disturbed.

REGISTER COMMANDS

These commands put information into the various control registers of the G-20. These commands cannot be indexed because the I field of the command is used to specify which register is being addressed. During the "operand select" phase of the register commands, Index Addressing is used instead of Standard Addressing (page **4**).

The operand B is always brought to the Operand Assembly register in number format, and floating-point arithmetic is used in assembling it. The operand B is automatically shifted to a zero exponent and is truncated before the action of the command begins. In the commands with two operands, the number brought from the register is automatically supplied with a zero exponent and positive sign. The names, the alphabetic codes and the numeric designations of these registers are given below:

REGISTER NAME	ALPHA CODE	NUMERIC DESIGNATION IN OCTAL
Next Command	NC	00
Enable	CE	01
Line Response	LR	02
Interrupt Request	IR	03
Pickapoint Exponent	PE	04

The Next Command register contains the address of the next command to be executed. It is automatically incremented by one as soon as the operand for the current command is brought to the Operand Assembly register and before the operation specified by that command has begun. When the operation specified is a transfer, the transfer address is placed in the Next Command register and the program is continued from the transfer address.

The Enable register is a 15-bit register which contains the enable indicators for the interrupt flags (bits 30 and 31 in the three formats, command, Logic and data), the Pickapoint switch bit, a one-second real time interrupt bit, a sound tone enable bit and various other interrupt and input-output switches. The Control switch which controls the interrupt hardware is in the 0 bit of the Enable register (see page **11** for further explanations).

The Line Response register is concerned with Input-Output and its operation is discussed on page **52**.

The Interrupt Request (IR) register is a 15-bit register which contains the interrupt request bits corresponding to the enable bits of the CE register (see Table 2, page **11**). The exceptions are the exponent overflow interrupt request indicator, the illegal address request and four receive interrupt requests.

The Pickapoint Exponent (PE) register holds the Pickapoint exponent in sign and magnitude form. Bit number six is a 1 if the exponent is negative, a 0 if it is positive. The mantissa of the exponent is contained in the first six bit positions. Do not load a negative zero into the Pickapoint Exponent register since the result of this action is unpredictable.

LR 056 Load Register I $\quad {}_{14}B_0 \rightarrow (\text{Reg. I})$

The contents of the specified register are replaced by the first fifteen bits of the operand B. The Next Command register cannot be addressed with this command.

XR 076 Selectively Reset Register I $\quad (\text{Reg. I}) \wedge {}_{14}B_0 \rightarrow (\text{Reg. I})$

The indicators in the Enable, Interrupt Request and Line Response registers are selectively turned OFF. Enter zeros into the operand B at those positions which are to be turned OFF.

XO 052 Selectively Read Register I to OA $\quad (\text{Reg. I}) \wedge {}_{14}B_0 \rightarrow {}_{14}(\text{OA})_0$, $0 \rightarrow {}_{41}(\text{OA})_{15}$

Specific bit positions in a specified register I can be read by this command. The answer is placed in the least significant 15 bit positions of the Operand Assembly register.

Enter "ones" into the operand B, assembled as described on page **29**, in positions corresponding to the bit positions of register I to be read. Enter "zeros" into those positions of B not to be read. Those bit positions and bit positions 15 through 41 of the Operand Assembly register are cleared to zero by the command. Only the Pickapoint Exponent register cannot be read. No interrupts can occur other than exponent overflow or illegal address when using this command.

XA	072	Selectively Read Register I to Accumulator	$(\text{Reg. I})\Lambda {}_{14}B_0 \rightarrow {}_{14}(\text{Acc})_0$, $0 \rightarrow {}_{41}(\text{Acc})_{15}$

Specific bit positions in a specified register I can be read by this command. The answer is placed in the least significant 15 bit positions of the accumulator. Enter "ones" into the operand B, assembled as described on page **29**, in positions corresponding to the bit positions of register I to be read. Enter "zeros" into those positions of B not to be read. Those bit positions and bit positions 15 through 41 of the accumulator are cleared to zero by the command. Only the Pickapoint Exponent register cannot be read. The Operand Assembly register is cleared to zero.

REPEAT COMMANDS

Two command words are required to initiate and to set a limit on the repeat operation. The first word consists of an operation code, M0, and the initial address of the block. When using the multiple-access commands, the B in all four addressing modes is the location at which the operation will begin. In preparing B, a number format is used, and floating-point arithmetic is performed. The final operand B will be shifted to a zero exponent and bit positions 15 through 41 will be made zero before the operations are performed. The remaining 15 bits, 0 through 14, are used as the location B. This is identical with the Putaway command addressing modes. For example, in mode 3, the addressing assembly would proceed like this:

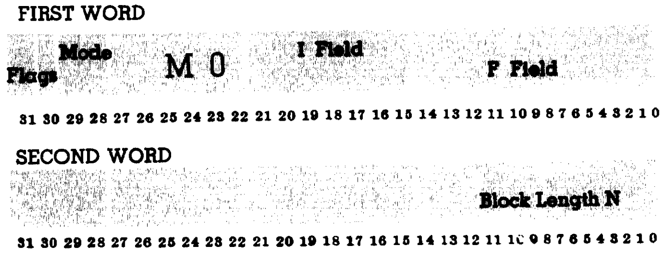
If	40 is stored in location 00030 = F
	35 is stored in location 00040,
	the contents of the Operand Assembly register is 0
and	the contents of the Index field is 0
then	

$$B = ((AO) + (F) + (I)) = (0 + (00030) + 0) = (+00040) = 35$$

The Repeat command will begin operation at location 00035. The contents of any location B in the Repeat commands will be referred to as operand B.

The second word designates the operation to be repeated and gives the upper limit N on the block length desired. N is held in integer form in the F number field of the second word. The numbers operated on by the Repeat commands can be Single or Extended precision. Therefore, the block length N is a count of the numbers, Single and Extended, which are brought from memory, not a count of addresses in the block.

REPEAT COMMAND FORMAT



M0 013 Initiate Repeat Operations, Begin at Location B

The operation to be repeated is completely specified in the most significant five bits (bits 23 through 27) in the operation code field of the second command word. Therefore, bit positions 21 and 22 are ignored in the operation of any Repeat command. Any one of the Non-numeric, Non-numeric Tests, Add-Subtract, and Add-Subtract Test commands can be repeated. Any other operation code which has a bit pattern identical with any of the above in the last five bit positions but different in the first two bit positions (21, 22) will be taken as one of the repeatable commands and the computer will not be aware of this error. Also, bits 28 through 31 and the I field (bits 15 through 20) of the second word will be ignored by the repeat operation. Before the execution of the M0 command, the following indicators in the Interrupt Request register should be in the OFF state:

Operand flag indicators (bit positions 12 through 9)
Overflow indicator IR8 (bit position 8)
Illegal address indicator IR0 (bit position 0)
(see Interrupt Request, page 11)

The M0 command and the next command are then read. If, as the operation proceeds, any of the above indicators are turned on, the operation is terminated and an interrogation of "Control" occurs.

If Control is OFF (contains a zero bit), computation continues at the location contained in the Next Command (NC) register.

If Control is ON (contains a one bit), the contents of the Next Command register are marked in location 64 and the program continues at location 65 at the start of the Interrupt Service Routine.

If the operation is terminated by the set limit or by the function of a test as in a Repeat Test command, the interrupt request indicators are interrogated as outlined on page 12.

The order of priority of repeat command terminations is as follows:

Repeat Add-Subtract and Non-numeric Commands

1. Any program fault interrupt
2. An enabled flag or the completion of block length

Repeat Add-Subtract Test and Non-numeric Test Commands

1. Any program fault interrupt
2. A result of the test
3. An enabled flag or the completion of the block length

When using the Repeat Non-numeric commands and Repeat Non-numeric Test commands, the word in location B will be used in the Logic format with all four addressing modes. When using the Repeat Add-Subtract commands and Repeat Add-Subtract Test commands, the word in location B will be used in a number format with all four addressing modes.

The Repeat commands function the same as those not repeated except where explicitly stated.

In this section the Repeat commands will be referred to by prefixing the command with an R. This will replace both the M0 and the second command. For example, a repeated clear and add command CA will be RCA.

Example

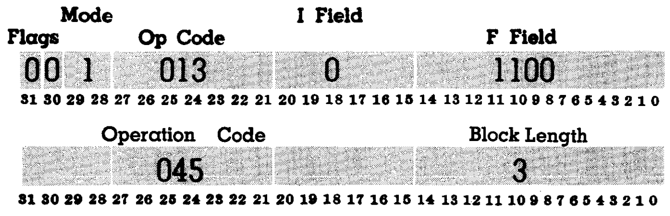
Consider:

Repeat Add command, RAD
Use a block length of N = 3
Use addressing mode 1
Use data flags in all words in the block of 0
Use command flags equal to zero
Use the following:

Contents of the accumulator equal to 25×8^3
Contents of the Operand Assembly register equal to 0
Contents of the Index field equal to 0
Starting address equal to 1100
Contents of location 1100 equal to 3100.0
Contents of location 3100 equal to 5.0
Contents of location 3101 equal to 0
Contents of location 3102 equal to 1000

The octal code for the M0 command is 013
The octal code for the AD command is 045

Therefore, the commands will have the following form:



These commands will:

assemble the starting location B which will be

$$B = (AO) + (F) + (I) = 0 + (1100) + 0 = 3100;$$

add the contents of the accumulator to the sum of the contents of the locations in the block;

place the final results in the accumulator, namely,

$$(Acc) = 25 \times 8^3 + 5.0 + 0 + 1000.0 = 26005.0$$

Note

The Control switch controls the interrupt transfers and is the zero bit of the Enable register (see Interrupt, page 11).

REPEAT ADD-SUBTRACT COMMANDS

RCA Search for the First Flagged Operand $B_f \rightarrow (Acc)$

The first operand B with an enabled flag is placed in the accumulator. If the Control switch is ON, the Next Command address is marked in location 64. The corresponding data flag bit in the Interrupt Request register is made a one, Control is turned OFF and the command in location 65 is executed.

If no enabled flag is encountered, the last operand B in the block is placed in the accumulator and the Next Command in sequence is executed.

If any other interrupt request is encountered, the search is terminated as shown in Table 3, page **34**.

RCS Search for the First Flagged Operand and Bring with the Sign Reversed $-B_f \rightarrow (Acc)$

The first operand B with an enabled flag is placed in the accumulator with its sign reversed. If the Control switch is ON, the Next Command address is marked in location 64. The corresponding data flag bit in the Interrupt Request register is made a one, Control is turned OFF and the command in location 65 is executed.

If no enabled flag is encountered, the last operand B in the block is placed in the accumulator with its sign reversed and the Next Command in sequence is executed.

If any other interrupt request is encountered, the search is terminated as shown in Table 3, page **34**.

RAD Add the Contents of the Accumulator to the Σ B $(Acc) + \Sigma B \rightarrow (Acc)$

All of the operands B in the block are added to the contents of the accumulator and the sum is placed in the accumulator. The Next Command in sequence is executed.

If an enabled flag or any other interrupt request is encountered, see Table 3, page **34**, for the operation of the command.

RSU Subtract the ΣB from the Contents of the Accumulator $(Acc) - \Sigma B \rightarrow (Acc)$

All of the operands B in the block are subtracted from the contents of the accumulator and the difference is placed in the accumulator. The Next Command in sequence is executed.

If an enabled flag or any other interrupt request is encountered, see Table 3, page **34**, for the operation of the command.

RRS $(-1)^i[(Acc) - B_1 + B_2 - \dots] \rightarrow (Acc)$
or $(-1)^i[(Acc) + \Sigma(-1)^i(B_i)] \rightarrow (Acc)$

The contents of the accumulator are subtracted from the first operand B. This difference is placed in the accumulator. The new contents of the accumulator are subtracted from the second operand B. This difference is placed in the accumulator. This procedure continues through the last operand B in the block. The Next Command in sequence is executed.

If an enabled flag or any other interrupt request is encountered, see Table 3, page **34**, for the operation of the command.

RSS $(-1)^i[(Acc) + B_1 - B_2 \dots] \rightarrow (Acc)$
or $(-1)^i[(Acc) - \Sigma(-1)^i B_i] \rightarrow (Acc)$

The contents of the accumulator are added to the first operand B. This sum with its sign reversed is placed in the accumulator. The new contents of the accumulator are added to the second operand B. This sum with its sign reversed is placed in the accumulator. This procedure continues through the last operand B in the block. The Next Command in sequence is executed.

If an enabled flag or any other interrupt request is encountered, see Table 3, page **34**, for the operation of the command.

RAV $|\dots||(Acc) + B_1| + B_2| \dots + B_n| \rightarrow (Acc)$

The first operand B is added to the contents of the accumulator. The absolute value of this sum is placed in the accumulator. The new contents of the accumulator are added to the second operand B. The absolute value of this sum is placed in the accumulator. This procedure continues through the last operand B in the block. The Next Command in sequence is executed.

If an enabled flag or any other interrupt request is encountered, see Table 3, page **34**, for the operation of the command.

RSV $|\dots|||(Acc) - B_1| - B_2| - B_3| \dots - B_n| \rightarrow (Acc)$

The first operand B is subtracted from the contents of the accumulator. The absolute value of this difference is placed in the accumulator. The second operand B is subtracted from the new contents of the accumulator. This procedure continues through the last operand B in the block. The Next Command in sequence is executed.

If an enabled flag or any other interrupt request is encountered, see Table 3, page 34, for the operation of the command.

REPEAT ADD-SUBTRACT INTERRUPTS

Condition Encountered	Control OFF	Control ON
Flagged Operand (not enabled)	Ignored	Ignored
Flagged Operand (enabled)	1→IR9 or IR10 f (TL)→(Acc) Resume at (NC)	1→IR9 or IR10 f (TL)→(Acc) Mark (NC) in location 64 Go to location 65 Control turned OFF
Illegal Address	1→IR0 f (TL−1)→(Acc) Resume at (NC)	1→IR0 f (TL−1)→(Acc) Mark (NC in location 64 Go to location 65 Control turned OFF
Exponent Overflow	1→IR8 f (TL)→(Acc) Resume at (NC)	1→IR8 f (TL)→(Acc) Mark (NC) in location 64 Go to location 65 Control turned OFF

TL: the Terminating Location which is the address of the operand B that terminated the Repeat command.

f(TL): the result of the function of the Repeat command operating on all of the operands B up through the operand at Location TL.

(NC): contents of the Next Command address register.

Control: the first bit position of the Enable register.

IRn: bit number *n* of the Interrupt Request register.

Condition Encountered: this condition encountered during processing of data block.

TABLE 3

EXPLANATION OF TABLE 3

If a flagged operand B is encountered, the flag is ignored if the flag is not enabled in the Enable register.

If a flagged operand B (enabled) is encountered when Control is OFF, a one is placed in the data flag bit in the Interrupt Request register which corresponds to the enabling bit in the Enable register. The result of the command function operating through location TL is placed in the accumulator and the Next Command in sequence is executed. The enabling flag must be a data flag.

If a flagged operand B (enabled) is encountered when Control is ON, a one is placed in the data flag bit in the Interrupt Request register which corresponds to the enabling bit in the Enable register. The result of the command function operating through location TL is placed in the accumulator, the Next Command address is marked in location 64, Control is turned OFF and the command in location 65 is executed.

If an illegal address is encountered when Control is OFF, a one is placed in bit number zero of the Interrupt Request register, the result of the command function operating through location TL−1 is placed in the accumulator and the Next Command in sequence is executed.

If an illegal address is encountered when Control is ON, a one is placed in bit number zero of the Interrupt Request register, the result of the command function operating through location TL−1 is placed in the accumulator, the Next Command address is marked in location 64, Control is turned OFF and the command in location 65 is executed.

If an exponent overflow is encountered when Control is OFF, a one is placed in bit number eight of the Interrupt Request register, the result of the command function operating on TL is placed in the accumulator and the Next Command in sequence is executed.

If an exponent overflow is encountered when Control is ON, a one is placed in bit number eight of the Interrupt Request register, the result of the command function operating on TL is placed in the accumulator, the Next Command address is marked in location 64, Control is turned OFF and the command in location 65 is executed.

REPEAT ADD-SUBTRACT TEST COMMANDS

RTP	Find the Address of the First Non-Positive Operand B	If $B \leq 0$, TL $+ 1 \rightarrow$ (Acc)

A search is made for the first non-positive operand B. If a non-positive operand B is encountered, an address one greater than the address of B is placed in the accumulator. The Next Command in sequence is skipped and the following command is executed.

If a non-positive operand B is not encountered, an address one greater than the address of the last operand in the block is placed in the accumulator. The Next Command in sequence is executed.

If any flagged operand or other interrupt request is encountered, the search is terminated as shown in Table 4, page 36.

RTN	Find the address of the First Non-negative Operand B	If $B \geq 0$, TL $+ 1 \rightarrow$ (Acc)

A search is made for the first non-negative operand B.

If a non-negative operand B is encountered, an address one greater than the address of B is placed in the accumulator. The Next Command in sequence is skipped and the following command is executed.

If a non-negative operand B is not encountered, an address one greater than the address of the last operand in the block is placed in the accumulator. The Next Command in sequence is executed.

If any flagged operand or other interrupt request is encountered, the search is terminated as shown in Table 4, page 36.

RT2	Find the address of the first Operand B such that (Acc) $+ B \leq 0$	If (Acc) $+ B \leq 0$, TL $+ 1 \rightarrow$ (Acc)

A search is made for the first operand B, which, when added to the contents of the accumulator, will produce a sum that is less than or equal to zero.

If such an operand B is encountered, an address one greater than the address of B is placed in the accumulator. The Next Command in sequence is skipped and the following command is executed.

If no such operand B is encountered, an address one greater than the address of the last operand in the block is placed in the accumulator. The Next Command in sequence is executed.

If any flagged operand or other interrupt request is encountered, the search is terminated as shown in Table 4, page 36.

RTG	Find the address of the first operand B which is greater than or equal to the contents of the accumulator	If $B \geq$ (Acc), TL$+1\rightarrow$(Acc)

A search is made for the first operand B which is greater than or equal to the contents of the accumulator.

If such an operand B is encountered, an address one greater than the address of B is placed in the accumulator. The Next Command in sequence is skipped and the following command is executed.

If no such operand B is encountered, an address one greater than the address of the last operand in the block is placed in the accumulator. The Next Command in sequence is executed.

If any flagged operand or other interrupt request is encountered, the search is terminated as shown in Table 4, page 36.

RTL	Find the address of the first operand B which is less than or equal to the contents of the accumulator	If $B \leq$ (Acc), TL $+ 1 \rightarrow$ (Acc)

A search is made for the first operand B which is less than or equal to the contents of the accumulator.

If such an operand B is encountered, an address one greater than the address of B is placed in the accumulator. The Next Command in sequence is skipped and the following command is executed.

If no such operand B is encountered, an address one greater than the address of the last operand in the block is placed in the accumulator. The Next Command in sequence is executed.

If any flagged operand or other interrupt request is encountered, the search is terminated as shown in Table 4, page 36.

RT1	Find the address of the first operand B such that $-[B+(Acc)]<0$	If $-[(Acc)+B] \leq 0$, $TL+1 \rightarrow (Acc)$

A search is made for the first operand B which, when added to the contents of the accumulator and the sign of the sum is reversed, will produce a result that is less than or equal to zero.

If such an operand B is encountered, an address one greater than the address of B is placed in the accumulator. The Next Command in sequence is skipped and the following command is executed.

If no such operand B is encountered, an address one greater than the address of the last operand in the block is placed in the accumulator. The Next Command in sequence is executed.

If any flagged operand or other interrupt request is encountered, the search is terminated as shown in Table 4.

RT3	Find the address of the first operand B which is equal to the contents of the accumulator with its sign reversed	If $-B=(Acc)$, $TL+1 \rightarrow (Acc)$

A search is made for the address of the first operand B which is equal to the contents of the accumulator with its sign reversed. If such an operand B is encountered, an address one greater than the address of B is placed in the accumulator. The Next Command in sequence is skipped and the following command is executed.

If no such operand B is encountered, an address one greater than the address of the last operand in the block is placed in the accumulator. The Next Command in sequence is executed in the accumulator.

If any flagged operand or other interrupt request is encountered, the search is terminated as shown in Table 4.

RTU	Find the address of the first operand B which is equal to the contents of the accumulator	If $B=(Acc)$, $TL+1 \rightarrow (Acc)$

A search is made for the first operand B which is equal to the contents of the accumulator.

If such an operand B is encountered, an address one greater than the address of B is placed in the accumulator. The Next Command in sequence is skipped and the following command is executed.

If no such operand B is encountered, an address one greater than the address of the last operand in the block is placed in the accumulator. The Next Command in sequence is executed.

If any flagged operand or other interrupt request is encountered, the search is terminated as shown in Table 4.

REPEAT ADD-SUBTRACT TEST INTERRUPTS

Condition Encountered	Control OFF	Control ON
Flagged Operand B (not enabled)	Ignored	Ignored
Flagged Operand B (enabled)	$1 \rightarrow$ IR9 or IR10 $TL + 1 \rightarrow (Acc)$ Go to (NC)	$1 \rightarrow$ IR9 or IR10 $TL + 1 \rightarrow (Acc)$ Mark (NC) in Location 64 Go to location 65 Control turned OFF
Illegal Address	$1 \rightarrow$ IR0 $TL \rightarrow (Acc)$ Go to (NC)	$1 \rightarrow$ IR0 $TL \rightarrow (Acc)$ Mark (NC) in Location 64 Go to location 65 Control turned OFF
Exponent Overflow	$1 \rightarrow$ IR8 $TL \rightarrow (Acc)$ Go to (NC)	$1 \rightarrow$ IR8 $TL \rightarrow (Acc)$ Mark (NC) in Location 64 Go to location 65 Control turned OFF

TL: the Terminating Location which is the address of operand B that terminates the Repeat command

(NC): contents of the Next Command address register

Control: the first bit position of the Enable register

IRn: bit number *n* of the Interrupt Request register

Condition Encountered: this condition encountered during processing of data block

TABLE 4

EXPLANATION OF TABLE 4

If a flagged operand B is encountered, the flag is ignored if the flag is not enabled in the Enable register.

If a flagged operand B (enabled) is encountered when Control is OFF, a one is placed in the data flag bit in the Interrupt Request register which corresponds to the enabling bit in the Enable register. An address one greater than the address of the flagged operand is placed in the accumulator and the Next Command in sequence is executed. The enabling flag must be a data flag.

If a flagged operand B (enabled) is encountered when Control is ON, a one bit is placed in the data flag bit in the Interrupt Request register which corresponds to the enabling bit in the Enable register. An address one greater than the address of the flagged operand is placed in the accumulator, the Next Command address is marked in location 64, Control is turned OFF and the command in location 65 is executed.

If an illegal address is encountered when Control is OFF, a one is placed in bit number zero of the Interrupt Request register, the address is placed in the accumulator and the Next Command in sequence is executed.

If an illegal address is encountered when Control is ON, a one is placed in bit number zero of the Interrupt Request register, the address is placed in the accumulator, the Next Command address is marked in location 64, Control is turned OFF and the command in location 65 is executed.

If an exponent overflow is encountered when Control is OFF, a one is placed in bit number eight of the Interrupt Request register, the address of the interrupting operand is placed in the accumulator and the Next Command in sequence is executed.

If an exponent overflow is encountered when Control is ON, a one is placed in bit number eight of the Interrupt Request register, the address of the interrupting operand is placed in the accumulator, the Next Command address is marked in location 64, Control is turned OFF and the command in location 65 is executed.

REPEAT NON-NUMERIC COMMANDS

Each operand B in this section will be assembled in Logic form as follows:

The entire 32 bits in location B are placed in the least significant 32 bit positions of the Operand Assembly register (Accumulator).

Bit positions 32 through 41 of the Operand Assembly register (Accumulator) are made zero.

The sign in the Operand Assembly register (Accumulator) is made positive.

The exponent in the Operand Assembly register (Accumulator) is made a positive zero.

(See page 18 for more explanation of this form.)

RBB Search for the First Logic Flagged Operand $_{31}(B)_0 \rightarrow {}_{31}(Acc)_0$, $0 \rightarrow {}_{41}(Acc)_0$

The contents of the first location B which contains an enabled Logic flag are placed in the accumulator in Logic form. If the Control switch is ON, the Next Command address is marked in location 64, the corresponding Logic flag bit in the Interrupt Request register is made a one, Control is turned OFF and the command in location 65 is executed.

If no enabled flag is encountered, the contents of the last location in the block are placed in the accumulator in Logic form (page 18). The Next Command in sequence is executed.

If any other interrupt request is encountered, the search is terminated as shown in Table 5, page 39.

RBC Search for the First Flagged Operand and Place the Complement of It $_{31}(\overline{B})_0 \rightarrow {}_{31}(Acc)_0$, $0 \rightarrow {}_{41}(Acc)_0$

The complement of the contents of the first location B which contains an enabled Logic flag is placed in the accumulator in Logic form. If the Control switch is ON, the Next Command address is marked in location 64, the corresponding Logic flag bit in the Interrupt Request register is made a one, Control is turned OFF and the command in location 65 is executed.

If no enabled flag is encountered, the complement of the contents of the last location in the block is placed in the accumulator in Logic form. The Next Command in sequence is executed.

If any other interrupt request is encountered, the search is terminated as shown in Table 5, page 39).

RAL Repeat Add Logical $_{31}(Acc) + \Sigma(B+n)_0 \rightarrow {}_{31}(Acc)_0$, $0 \rightarrow {}_{41}(Acc)_{32}$

As each operand B in the block is encountered, it is assembled in Logic form. All of the operands B in the block are added to the contents of the accumulator. The first 32 bits of the sum are placed in the accumulator and bits 32 through 41 are made zero. The Next Command in sequence is executed.

If an enabled flag or any other interrupt request is encountered see Table 5, page 39, for the operation of the commands.

RSL Repeat Subtract Logical $_{31}(Acc) - \Sigma(B+n) \rightarrow {}_{31}(Acc)_0$, $0 \rightarrow {}_{41}(Acc)_{32}$

As each operand B in the block is encountered, it is assembled in Logic form (page 18). All of the operands B in the block are subtracted from the contents of the accumulator. The first 32 bits of the difference are placed in the accumulator and bits 32 through 41 are made zero. The Next Command in sequence is executed.

If an enabled flag or any other interrupt request is encountered see Table 5, page 39 for the operation of the command.

RXX Repeat Extract $_{31}[(Acc)\Lambda B_1 \Lambda B_2 \Lambda \ldots \Lambda B_n]_0 \rightarrow {}_{31}(Acc)_0$, $0 \rightarrow {}_{41}(Acc)_{32}$

As each operand B in the block is encountered, it is assembled in Logic form (page). The contents of the accumulator are shifted to zero exponent and any digits shifted out of the accumulator are truncated. The first operand B is extracted from the contents of the accumulator. The result is placed in the accumulator. The second operand B is extracted from the new contents of the accumulator. The result is placed in the accumulator. This procedure continues through the last operand B in the block. The sign of the original contents of the accumulator is retained. The Next Command in sequence is executed.

If an enabled flag or any other interrupt request is encountered, see Table 5, page 39, for the operation of the command.

RXC Repeat Extract Complement $_{31}[(Acc)\Lambda \overline{B}_1 \Lambda \overline{B}_2 \Lambda \ldots \Lambda \overline{B}_n]_0 \rightarrow {}_{31}(Acc)_0$, $0 \rightarrow {}_{41}(Acc)_{32}$

As each operand B in the block is encountered, it is assembled in Logic form. The contents of the accumulator are shifted to zero exponent and any digits shifted out of the accumulator are truncated.

The complement of the first operand B is extracted from the contents of the accumulator. The result is placed in the accumulator. The complement of the second operand B is extracted from the new contents of the accumulator. The result is placed in the accumulator. This procedure continues through the last operand B in the block. The sign of the original contents of the accumulator is retained. The Next Command in sequence is executed.

If an enabled flag or any other interrupt request is encountered, see Table 5, page **39**, for the operation of the command.

RUU Repeat Unite $_{31}[(Acc)\vee B_1\vee B_2\vee \ldots \vee B_n]_0 \rightarrow {}_{31}(Acc)_0$, $0 \rightarrow {}_{41}(Acc)_{32}$

As each operand B in the block is encountered, it is assembled in Logic form (page **18**). The contents of the accumulator are shifted to zero exponent and any digits shifted out of the accumulator are truncated. The first operand B is united with the contents of the accumulator. The result is placed in the accumulator. The second operand B is united with the new contents of the accumulator. The result is placed in the accumulator. This procedure continues through the last operand B in the block. The sign of the original contents of the accumulator is retained. The Next Command in sequence is executed.

If an enabled flag or any other interrupt request is encountered, see Table 5, page **39**, for the operation of the command.

RUC Repeat Unite Complement $_{31}[(Acc)\vee \overline{B}_1\vee \overline{B}_2\vee \ldots \vee \overline{B}_n]_0 \rightarrow {}_{31}(Acc)_0$, $0 \rightarrow {}_{41}(Acc)_{32}$

As each operand B in the block is encountered, it is assembled in Logic form. The contents of the accumulator are shifted to zero exponent and any digits shifted out of the accumulator are truncated. The complement of the first operand B is united with the contents of the accumulator. The result is placed in the accumulator. The complement of the second operand B is united with the new contents of the accumulator. The complement of the second operand B is united with the new contents of the accumulator. The result is placed in the accumulator. This procedure continues through the last operand B in the block. The sign of the original contents of the accumulator is retained. The Next Command in sequence is executed.

If an enabled flag or any other interrupt request is encountered, see Table 5, page **39**, for the operation of the command.

REPEAT NON-NUMERIC INTERRUPTS

Condition Encountered	Control OFF	Control ON
Flagged Operand B (Not enabled)	Ignored	Ignored
Flagged Operand B (Enabled)	$1\rightarrow IR11$ or $IR12$ $_{31}f(TL)_0\rightarrow {}_{31}(Acc)_0$ Resume at (NC)	$1\rightarrow IR11$ or $IR12$ $_{31}f(TL)_0\rightarrow {}_{31}(Acc)_0$ Mark (NC) in location 64 Go to location 65 Control turned OFF
Illegal Address	$1\rightarrow IR0$ $_{31}f(TL-1)_0\rightarrow {}_{31}(Acc)_0$ Resume at (NC)	$1\rightarrow IR0$ $_{31}f(TL-1)_0\rightarrow {}_{31}(Acc)_0$ Mark (NC) in location 64 Go to location 65 Control turned OFF
Exponent Overflow	Not Possible	

TL: The Terminating Location which is the address of operand B that terminates the Repeat command.

$_{31}f(TL)_0$: the least significant 32 bits of the result of the function of the Repeat command operating on all of the operands B up through the operand in location TL.

(NC): contents of the Next Command address register

Control: the first bit position of the Enable register

IRn: bit number *n* of the Interrupt Request register

Condition Encountered: this condition encountered during processing of data block.

TABLE 5

EXPLANATION OF TABLE 5

If a flagged operand B is encountered, the flag is ignored if the flag is not enabled in the Enable register.

If a flagged operand B (enabled) is encountered when Control is OFF, a one is placed in the Logic flag bit in the Interrupt Request register which corresponds to the enabling bit in the Enable register. The result of the command function operating on TL is placed in the accumulator and the Next Command in sequence is executed. The enabling flag must be a Logic flag.

If a flagged operand B (enabled) is encountered when Control is ON, a one is placed in the Logic flag bit in the Interrupt Request register which corresponds to the enabling bit in the Enable register. The result of the command function operating on TL is placed in the accumulator, the Next Command address is marked in location 64, Control is turned OFF and the command in location 65 is executed.

If an illegal address is encountered when control is OFF, a one is placed in bit number zero of the Interrupt Request register, the result of the command function operating on TL − 1 is placed in the accumulator and the Next Command in sequence is executed.

If an illegal address is encountered when Control is ON, a one is placed in bit number zero of the Interrupt Request register, the result of the command function operating on TL − 1 is placed in the accumulator, the Next Command address is marked in location 64, Control is turned OFF and the command in location 65 is executed.

An Exponent Overflow is not possible with a Repeat Non-numeric command.

REPEAT NON-NUMERIC TEST COMMANDS

RJ0	Find the Address of the First Non-zero Operand B	If ${}_{31}(B)_0 \neq 0$, then $TL + 1 \rightarrow (Acc)$

A search is made for the first non-zero operand B.

As each operand B in the block is encountered, it is assembled in Logic form (page 38).

If a non-zero operand B is encountered, an address one greater than the address of B is placed in the accumulator. The Next Command in sequence is skipped and the following command is executed.

If a non-zero operand B is not encountered, an address one greater than the address of the last operand in the block is placed in the accumulator. The Next Command in sequence is executed.

If any Logic flagged operand or other interrupt request is encountered, the search is terminated as shown in Table 6, page 43.

RJ1	Find the Address of the First Operand B which is not all ones	If ${}_{31}\overline{B}_0 \neq 0$, then $TL + 1 \rightarrow (Acc)$

A search is made for the first operand B which is not all ones.

As each operand B in the block is encountered, it is assembled in Logic form.

If an operand B is encountered which is not all ones, an address one greater than the address of B is placed in the accumulator. The Next Command in sequence is skipped and the following command is executed.

If every operand B encountered is all ones, an address one greater than the address of last operand B in the block is placed in the accumulator. The Next Command in sequence is executed.

If any Logic flagged operand or other interrupt request is encountered, the search is terminated as shown in Table 6, page 43.

RJS	Find the Address of the First Operand B, which when added to the Contents of the Accumulator is equal to zero.	If ${}_{31}B + (Acc)_0 = 0$, then $TL + 1 \rightarrow (Acc)$

A search is made for the first operand B which, when added to the contents of the accumulator, is equal to zero.

As each operand B in the block is encountered, it is assembled in Logic form.

If an operand B is encountered which, when added to the contents of the accumulator, is equal to zero, an address one greater than the address of B is placed in the accumulator. The Next Command in sequence is skipped and the following command is executed.

If such an operand B is not encountered, an address one greater than the address of the last operand B in the block is placed in the accumulator. The Next Command in sequence is executed.

If any Logic flagged operand or other interrupt request is encountered, the search is terminated as shown in Table 6, page 43.

RJG	Find the Address of the First Operand B which, when subtracted from the contents of the Accumulator, is equal to zero.	If ${}_{31}(Acc) - B_0 = 0$, then $TL + 1 \rightarrow (Acc)$

A search is made for the first operand B which, when subtracted from the contents of the accumulator, is equal to zero.

As each operand B in the block is encountered, it is assembled in Logic form (page 38).

If an operand B is encountered which, when subtracted from the contents of the accumulator, is equal to zero, an address one greater than the address of B is placed in the accumulator. The Next Command in sequence is skipped and the following command is executed.

If such an operand B is not encountered, an address one greater than the address of the last operand B in the block is placed in the accumulator. The Next Command in sequence is executed.

If any Logic flagged operand or other interrupt request is encountered, the search is terminated as shown in Table 6, page 43.

RJX	Find the Address of the First Operand B which has a "one" in any of the bit positions indicated by a "one" in the Accumulator	If ${}_{31}B \wedge (Acc)_0 \neq 0$, then $TL + 1 \rightarrow (Acc)$

A search is made for the first operand B which has a "one" in any of the bit positions indicated by a "one" in the accumulator.

As each operand B in the block is encountered, it is assembled in Logic form. The contents of the accumulator are shifted to zero exponent and any digits shifted out of the accumulator are truncated. Bit positions 32 through 41 of the accumulator are then set to zero.

If an operand B is encountered which has a "one" in any of the bit positions indicated by a "one" in the accumulator, an address one greater than the address of B is placed in the accumulator. The Next Command in sequence is skipped and the following command is executed.

If such an operand B is not encountered, an address one greater than the address of the last operand B in the block is placed in the accumulator. The Next Command in sequence is executed.

If any Logic flagged operand or other interrupt request is encountered, the search is terminated as shown in Table 6, page **43**.

RXJ	Find the Address of the First Operand B which has a "zero" in any of the bit positions indicated by a "one" in the Accumulator	If ${}_{31}\bar{B}\wedge(\mathrm{Acc})_0\neq 0$, then $TL+1\rightarrow(\mathrm{Acc})$

A search is made for the first operand B which has a "zero" in any of the bit positions indicated by a "one" in the accumulator.

As each operand B in the block is encountered, it is assembled in Logic form (page **38**). The contents of the accumulator are shifted to zero exponent and any digits shifted out of the accumulator are truncated. Bit positions 32 through 41 of the accumulator are then set to zero.

If an operand B is encountered which has a "zero" in any of the bit positions indicated by a "one" in the accumulator, an address one greater than the address of B is placed in the accumulator. The Next Command in sequence is skipped and the following command is executed.

If such an operand B is not encountered, an address one greater than the address of the last operand B in the block is placed in the accumulator. The Next Command in sequence is executed.

If a Logic flagged operand or other interrupt request is encountered, the search is terminated as shown in Table 6, page **43**.

RJU	Find the address of the first operand B which, when united with the contents of the Accumulator, has a "one" in any bit position	If ${}_{31}B\vee(\mathrm{Acc})_0\neq 0$, then $TL+1\rightarrow(\mathrm{Acc})$

This command unites the contents of the accumulator with each operand B and tests this union for a "one" in any position.

As each operand B in the block is encountered, it is assembled in Logic form. The contents of the accumulator are shifted to zero exponent and any digits shifted out of the accumulator are truncated. Bit positions 32 through 41 of the accumulator are set to zero.

If an operand B is encountered, which, when united with the contents of the accumulator, has a "one" in any bit position, an address one greater than the address of B is placed in the accumulator. The Next Command in sequence is skipped and the following command is executed.

If such an operand B is not encountered, an address one greater than the address of the last operand B in the block is placed in the accumulator. The Next Command in sequence is executed.

If a Logic flagged operand or other interrupt request is encountered, the search is terminated as shown in Table 6, page **43**.

RUJ	Find the address of the first operand B which when united with the contents of the accumulator has a "zero" in any of the 32 bit positions	If ${}_{31}\bar{B}\vee(\mathrm{Acc})_0\neq 0$, then $TL+1\rightarrow(\mathrm{Acc})$

This command unites the contents of the accumulator with the complement of each operand B and tests this union for a one in any position.

As each operand B in the block is encountered, it is assembled in Logic form (page **38**). The contents of the accumulator are shifted to zero exponent and any digits shifted out of the accumulator are truncated. Bit positions 32 through 41 of the accumulator are set to zero.

If the contents of the accumulator contain a "one" in any bit position, an address one greater than the address of the first operand B is placed in the accumulator. The Next Command is skipped and the following command is executed.

If an operand B is encountered, which has a "zero" in any of the 32 bit positions, an address one greater than the address of B is placed in the accumulator. The Next Command in sequence is skipped and the following command is executed.

If such an operand B is not encountered, an address one greater than the address of the last operand B in the block is placed in the accumulator. The Next Command in sequence is executed.

If any Logic flagged operand or other interrupt request is encountered, the search is terminated as shown in Table 6, (page 43).

REPEAT NON-NUMERIC TEST INTERRUPTS

Condition Encountered	Control OFF	Control ON
Flagged Operand B (not enabled)	Ignored	Ignored
Flagged Operand B (enabled)	1 → IR11 or IR12 TL + 1 → (Acc) Go to (NC)	1 → IR11 or IR12 TL + 1 → (Acc) Mark NC in location 64 Go to location 65 Control turned OFF
Illegal Address	1 → IRO TL → (Acc) Go to (NC)	1 → IRO TL → (Acc) Mark NC in location 64 Control turned OFF
Exponent Overflow	Not Possible	

TL: the Terminating Location which is the address of the operand B that terminates the Repeat

(NC): contents of Next Command address register

Control: the first bit position of the Enable register

IRn: bit number n of the Interrupt Request register

Condition Encountered: this condition encountered during processing of data block

TABLE 6

EXPLANATION OF TABLE 6

If a flagged operand B is encountered, the flag is ignored if the flag is not enabled in the Enable register.

If a flagged operand B (enabled) is encountered when Control is OFF, a one is placed in the Logic flag bit in the Interrupt Request register which corresponds to the enabling bit in the Enable register. An address one greater than the address of the flagged operand is placed in the accumulator and the Next Command in sequence is executed. The enabling flag must be a Logic flag.

If a flagged operand B (enabled) is encountered when Control is ON, a one bit is placed in the Logic flag bit in the Interrupt Request register which corresponds to the enabling bit in the Enable register. An address one greater than the address of the flagged operand is placed in the accumulator, the Next Command address is marked in location 64, Control is turned OFF and the command in location 65 is executed.

If an illegal address is encountered when Control is OFF, a one is placed in bit number zero of the Interrupt Request register, the address is placed in the accumulator and the Next Command in sequence is executed.

If an illegal address is encountered when Control is ON, a one is placed in bit number zero of the Interrupt Request register, the address is placed in the accumulator, the Next Command address is marked in location 64, Control is turned OFF and the command in location 65 is executed.

An exponent overflow is not possible with a Repeat Non-numeric Test Command.

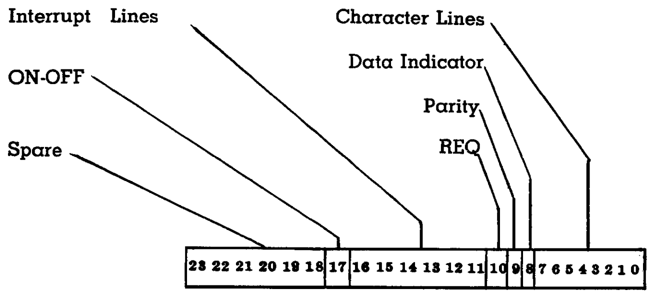
CENTRAL PROCESSOR INPUT-OUTPUT

The Bendix Communication System is a transmit—answer system. Conversations between units take place via Communication Lines which physically connect main terminal units to each other in the system. Every communication on a Communication Line is a series of individual character transmissions, such that one character is sent from one unit and a reply is given to it from the other unit in the communication, for every character in a transmission.

The Bendix Communication System is a three-element system, consisting of a controller, transmitter and receiver for every transmission. The controller notifies the unit which is to be the transmitter and sets it for transmission. The controller notifies the unit which is to be the receiver and sets it for reception. The controller then initiates the transmission, which is carried out between transmitter and receiver. The controller may also be one of the transmitting or receiving units. For example, the central processor, as a controller, can initiate a communication between a magnetic tape unit and a control buffer; or the control buffer, as both receiver and controller, can initiate and continue a magnetic tape to control buffer communication.

COMMUNICATION LINE

A Communication Line contains 24 conducting lines distributed as follows:



Character Lines (8 numeric, 1 data indicator, 1 parity)	10
REQ (Request Data) signal line	1
Interrupt lines	6
System on/off control	1
Spare	6

CHARACTER LINES

10-bit data and instruction characters are transmitted via the character lines. 8 bits of information will be sent over lines 0 through 7 each time a character is transmitted over a Communication Line. A data indicator, sent on line 8, and a parity bit, sent on line 9, will complete every character sent over the Communication Line.

The first 8 bits can be:

Instructions to the external equipment, or

Data to be stored by the external equipment, or

Output data to the external equipment.

With these 8 bits, 255 instructions can be sent over the Communication Line. Line number 8 (ninth least significant line) is used as a data character indicator. It will be made a "one" by the communication system when data is being transmitted from the central processor to the external equipment. It is "zero" when an instruction is being transmitted. These 8 lines can be represented by a three-octal digit code. The octal codes for the input-output instructions range from octal 001 to 377. The octal codes for data characters range in magnitude from 400 to 777. The general designations are listed below.

LINES	OCTAL DESIGNATIONS (without parity)	TOTAL NUMBER ALLOWED (Decimal)	ASSIGNMENT
9 876 543 210			
P 000 0XX XXX	000-037	48	Commands
P 000 10X XXX	040-057		
P 000 11X XXX	060-077	16	Queries
P 001 XXX XXX	100-177	64	Instruction Numerics 0-63 (Decimal)
P 01X XXX XXX	200-377	128	Call Instructions
P 1XX XXX XXX	400-777	256	Data Characters

Line number 9 contains an "even" parity bit. A "one" bit is put on line 9 by the communication system when necessary to make the numeric sum of the information on the first 10 lines even. Each character transmitted is checked for even parity by the receiver and, if there is a parity error, an alternate branch condition will be created (page 52).

REQUEST LINE, REQ

A "one" is transmitted on the REQ line by a receiving device (including the central processor) to request the next data character from the transmitting device. The REQ line is used only when data characters are being transmitted. When instructions are being transmitted, reply is made by a character sent on the character lines.

INPUT-OUTPUT INTERRUPTS AND INTERRUPT LINES

Lines 11 through 14 are "receive interrupt" lines. Interrupt requests are transmitted to the Interrupt Request register from external equipment via lines 11 through 14.

> Line number 11 turns bit number 1 of the Interrupt Request register ON
>
> Line number 12 turns bit number 2 of the Interrupt Request register ON
>
> Line number 13 turns bit number 3 of the Interrupt Request register ON
>
> Line number 14 turns bit number 4 of the Interrupt Request register ON

These interrupts operate like the exponent overflow interrupt and do not need corresponding enabling bits in the Enable register.

If Control is ON and one of the interrupt request bits is turned ON by one of the interrupt lines, the contents of the Next Command register will be marked in location 64 and the Interrupt Service Routine will be entered at location 65.

If Control is OFF and one of the interrupt request bits is turned ON by one of the interrupt lines, the Next Command in sequence is executed. When Control is again turned ON, an interrupt request will then be generated.

Any unit which is to be allowed to interrupt the central processor should be connected to one of the interrupt lines 11 through 14. An interrupt signal coming from any one of the external units connected to the interrupt line 2, for example, will set bit number 2 in the Interrupt Request register.

Lines 15 and 16 are transmit interrupt lines. Interrupt requests are transmitted to external equipment from the central processor via these lines. A "one" bit placed in bit position 3 in the Enable register (page 11) will transmit an interrupt to any equipment on interrupt line 15. A "one" bit loaded into bit position 4 in the Enable register will transmit an interrupt to any equipment on interrupt line 16. These two lines will normally be used to interrupt control units or other G-20's in a complex system. These bits are turned off automatically after the interrupt. The Control switch has no effect over these two indicators.

ON-OFF LINE

Line 17, ON-OFF Line, is a power line which can turn on all of the equipment in a system when the power is turned on at the master switch.

COMMUNICATION INSTRUCTIONS

An explanation of the following instruction characters is included to clarify discussion of external device operating states.

A GRN character is the response the central processor will receive to tell it to continue in the expected sequence. Its octal code is 002.

A RED character is the response the central processor will receive to tell it to branch to an alternate action. Its octal code is 003.

Example

> A search for the device which requested an interrupt is expected to continue until the device is found; any device questioned which did not send the interrupt will answer GRN. The device which sent the interrupt will answer RED to halt the search and cause the program to branch to the next section.

An END character is sent in a communication to signify the completion of a correct data transmission. The next line signal can come only from a controlling device. The octal code is 004.

An ERR character is sent in a communication to signify that the data transmission ended erroneously. The next line signal can come only from a controlling device. The octal code is 005.

An SDT character is used by the central processor to initiate a series of data transmissions. It is usually the 8-bit code in the second word of a block transmission command (page 48). The octal code is 010.

The OUT character switches a piece of external equipment to the "out of service" or off-line state. The octal code is 011.

OPERATING STATES

When **Out of Service**, a unit is effectively disconnected, and can only be put on-line by manual operation of the ON-LINE button. This will place the unit in the "Standby" state.

When in the **Standby** state, a unit examines and rejects all call signals until it receives its own unique call (with correct parity). When it hears this call, it answers GRN and enters the "Called" state.

When in the **Called** state, a unit can answer all queries and execute all instructions meaningful to it. Undefined characters or characters with incorrect parity are ignored.

When in the **Briefing** (partly instructed) state, a unit has received and acknowledged one or more meaningful instructions but is not completely instructed. In this state a unit can receive more instructions or answer queries. When it receives its own call in this state, it answers GRN and returns to the "Called" state. If it receives a call directed to another unit, it returns to Standby without issuing any signal.

When in the **Instructed** state, a unit has been given the complete set of instructions required to establish a data transfer but has not yet been told by the control unit to start data transfer.

A unit which has been set up to receive will, upon receiving an SDT instruction, enter the **Message** state and send a REQ signal.

A unit which has been instructed to transmit will enter the **Message** state when it receives SDT. No response is sent to the SDT, but an REQ character is transmitted by the receiver. A unit in the Instructed state, upon receiving its own call, will answer GRN and return to the Called state. No instructions except SDT and its own call are meaningful to a unit in the Instructed state. (Magnetic tape is an exception to these conditions, it goes directly to the Message state when completely instructed without waiting for SDT).

When in the **Busy** state, a unit is executing an operation other than a data transfer or it is interlocked. The Busy state has two substates—Busy-quiet and Busy-alert. In Busy-quiet, a unit can hear only its own call, which it will answer and go to Busy-alert. In Busy-alert, the unit will answer all queries and some instructions. Upon hearing a call directed to another unit, it will return to Busy-quiet without answering. Upon completing an operation, a Busy unit goes to Standby and sends an interrupt. (Magnetic tape is an exception; it goes to Standby without issuing any signal).

A unit engaged in a block communication is said to be in the **Message** state. All other units are forbidden to use the data character lines during this time. When the block transmission is terminated, the transmitter sends END or ERR and returns to Standby. The END or ERR signal returns the receiver to Standby. If the central processor is the receiver, it may terminate the block by sending END or ERR.

LC 157 Line Command $_7(B)_0 \rightarrow (CL)_1$

In all addressing modes, the mantissa of the operand B assembled in numeric form is shifted, if necessary, until the exponent of B is zero. Any digits shifted out of the register are lost and the operand is truncated. Then the least significant 8 bits of the operand B are transmitted over the character lines of the Communication Line. A "0" bit is automatically sent on line number 8 and a parity bit is sent on line number 9 to complete the 10-bit instruction character.

If the central processor then receives a GRN response from the specified input-output device, the Next Command in sequence is skipped and the following command is executed.

If any response other than GRN is received, that response is placed in the least significant 9 bit positions of the Line Response register and the Next Command in sequence is executed. The programmer must determine what the response is and take appropriate action.

If the response is REQ, bit positions 0 through 8 of the Line Response register are cleared, but bit number 10 is made a "1". The Next Command in sequence is executed. The programmer must determine what the response is and take appropriate action.

If the response contains a parity error, it is placed in the least significant 9 bit positions—0 through 8—of the Line Response register, bit number 9 is made "1" and the Next Command in sequence is executed. The programmer must determine what the response is and take appropriate action.

If no response occurs within one second, bit number 11 of the Line Response register is made "1" and the Next Command in sequence is executed. The programmer must determine what the response is and take appropriate action.

DC 117 Data Character $_7(B)_0 \rightarrow (CL),$

In all addressing modes, the mantissa of the operand B in normal numeric form is shifted, if necessary, until the exponent of B is zero. Any digits shifted out of the register are lost and the operand is truncated. Then the least significant 8 bits of operand B are transmitted over the character lines of the Communication Line. A "1" data character indicator is automatically sent on line number 8 and a parity bit is sent on line number 9 to complete the 10-bit data character.

The GRN, REQ, parity error, no response and all other responses are handled exactly like those in the Line Command.

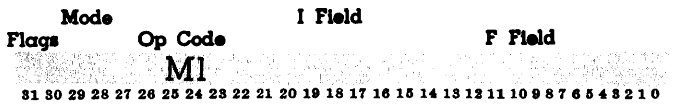
The Block Input-Output Command

Three basic block operations can be performed:

> "Receive" data characters and place them in memory
>
> "Transmit" a block of computer words from memory to an output device as "Data" characters
>
> "Transmit" a block of computer words from memory to an output device as "Instructions"

Two command words are required to initiate, set a limit on, and specify the repeat input-output operation. The first word consists of an operation code—M1—and the initial address of the block. When using these commands, the B in all four addressing modes is the location at which the operation will begin. In preparing B, a number format is used, and floating point arithmetic is performed. The final operand B will be shifted to a zero exponent in the Operand Assembly and bit positions 15 through 41 will be made zero before the operations are performed. The remaining 15 bits—0 through 14—are used as the address of location B. This is identical with the Putaway Addressing modes. The octal code for the M1 command is 033.

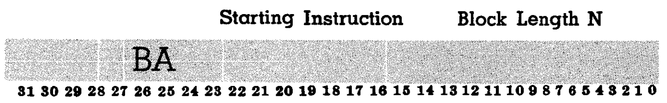
FIRST WORD



The second word designates the operation to be repeated, the upper limit N on the block length desired, and the starting instruction to the communication system. N is held in integer form in the F number field of the second word and is the number of machine words in the block. A block length N = 0 will cause one word to be transmitted.

The information specified in the second word is:

SECOND WORD



Bits 0 through 14 contain the block length N

Bits 15 through 22 contain the Starting Instruction such as Start Data Transmission (SDT).

Bits 23 and 24 are ignored

Bits 25 and 26, AB, specify which type of operation is to be performed:

B	A	
0	0	Transmit Data
0	1	Receive
1	0	Transmit Instructions

Bit 27 must be a "1" to specify the 8-bit character format and a "0" to specify the 6-bit character format.

Bits 28 through 31 are ignored.

CHARACTER FORMAT

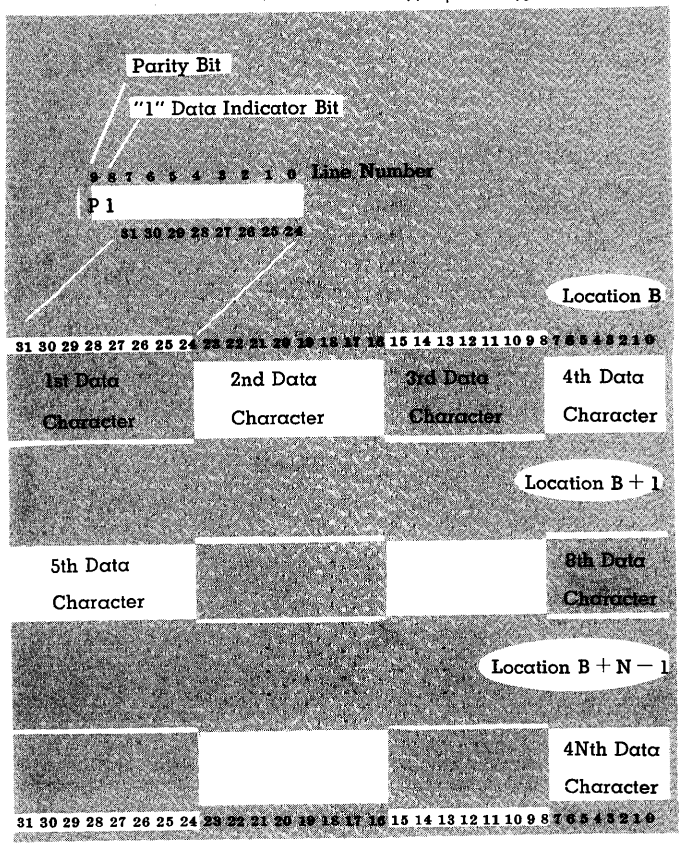
There are two character formats used with the block input-output commands. These are four 8-bit characters per computer word or one 8-bit character followed by four 6-bit characters from one computer word.

It is not necessary for terminal equipment to transmit the full 8 or 6 bits in a character. Zeros will be inserted in the most significant positions to fill the format (i.e. if some piece of equipment sends 4 bit characters to the computer, the most significant four spaces will be filled with zeros). In the 6-bit transmission, the bit positions 6 and 7 are filled with zeros to fill the 8 character lines.

COMMANDS

In this section the block input-output commands are discussed by format as well as by command. Both the M1 command word and the second command word are represented by a single, three-character code. The first character is an 8 or a 6 to denote the transmission format. For example, a block Transmit Data command, TD, in the 8-bit format is denoted by 8TD.

8TD Block Transmit Data, 8-bit Characters — Send ${}_{31}(B)_{24}$, ${}_{23}(B)_{16}$, ${}_{15}(B)_{8}$, ${}_{7}(B)_{0}$, ${}_{31}(B+1)_{24}$, $\ldots {}_{7}(B+N-1)_{0}$



Upon receipt of REQ the response from the instruction in bits 15 througn 22 of the second word of the Block Transmit Data Command, 8-bit data characters are transmitted to that piece of equipment starting with location B and continuing through location B + N − 1, where N is the block length. An REQ is received from the external equipment before the transmission of each data character.

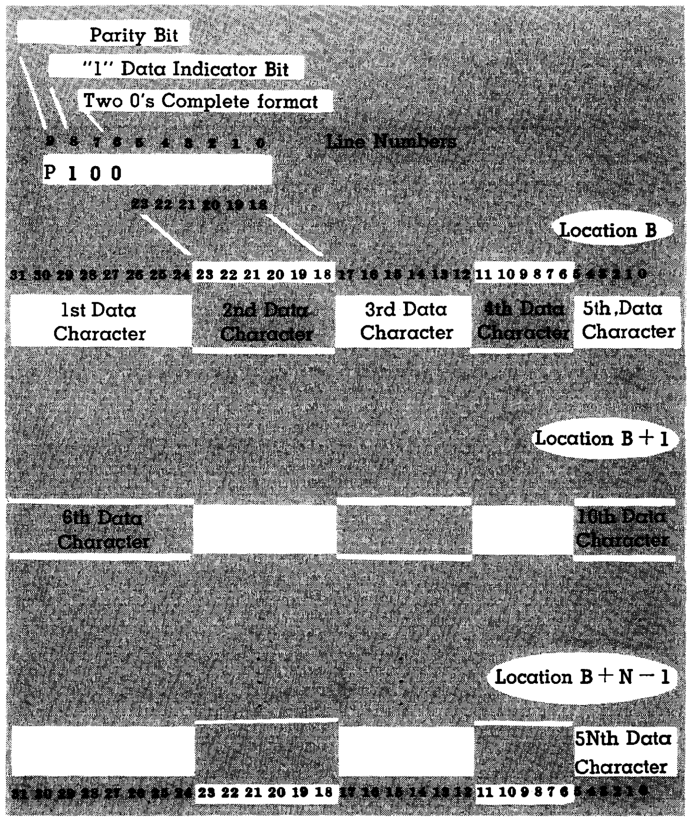
The first data character sent contains the 8 bits, 31 through 24, of location B. A "1" bit is automatically sent on line number 8 (ninth least significant bit of the data character) along with a parity bit on line number 9 to complete the 10-bit transmission. The "1"'s purpose is to indicate to the external equipment that the character is data, not an instruction. The second data character sent contains the 8 bits, 23 through 16, of location B. A "1" bit is automatically sent on line 8 along with a parity bit on line 9 to complete the 10-bit transmission. The fifth data character sent contains the 8 bits, 31 through 24, of location B + 1. A "1" bit is sent on line 8 along with a parity bit on line 9. This transmission procedure continues until the last data character is sent. This character contains the 8 bits, 7 through 0, of location B + N − 1. The "1" data bit and the parity bit are sent as before. Upon receipt of the next REQ an END character is transmitted. An address two greater than the address of the last location in the block is placed in the accumulator. The Next Command in sequence is skipped and the following command is executed.

If any response other than REQ is received, that response is placed in bit positions 0 through 8 of the Line Response register. The transmit operation terminates and an error, ERR, is transmitted. An address two larger than the address of the last location operated on is placed in the accumulator. The Next Command in sequence is executed.

If a parity error, time delay error, or memory overflow is encountered, the operation is terminated as shown in Table 8, page 52 . In these cases an ERR will be transmitted.

When the operation is terminated in the middle of a word, a code representing the character count of that word is placed in bits 14 through 12 of the Line Response register so that the programmer may determine the number of characters transmitted (see Table 9, page 53).

6TD Block Transmit Data — Send ${}_{31}(B)_{24}$, ${}_{23}(B)_{18}$, ${}_{17}(B)_{12}$, ${}_{11}(B)_{6}$, ${}_{5}(B)_{0}$, ${}_{31}(B+1)_{24}$, ${}_{23}(B+1)_{18}$, $\ldots {}_{5}(B+N-1)_{0}$.



Upon receipt of REQ, the response from the SDT instruction in bits 15 through 22 of the second word of the Block Transmit Data command, one 8-bit and four 6-bit data characters per word are transmitted to that piece of equipment starting with location B and continuing through location B + N − 1 where N is the block length. An REQ is received from the external equipment before each transmission.

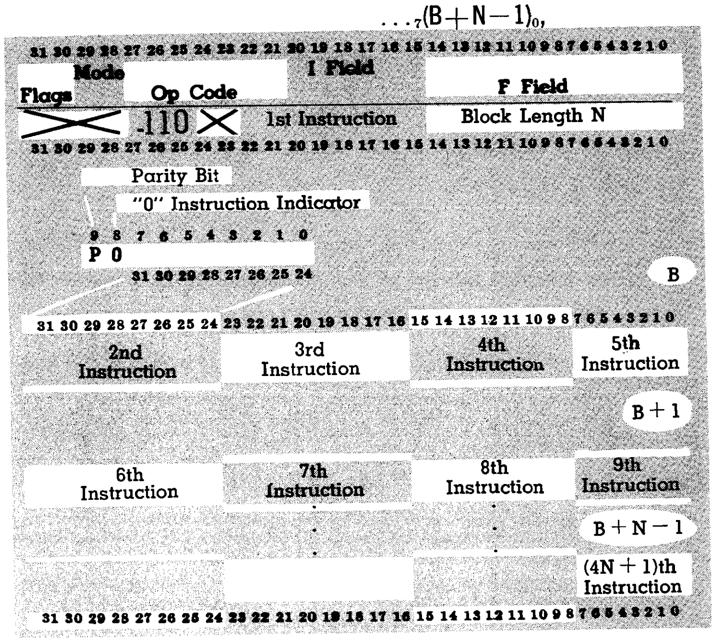
The first character sent contains the 8 bits, 31 through 24, of location B. A "1" bit is automatically sent on line 8 along with a parity bit on line 9 to complete the 10-bit transmission. The second data character sent contains the 6 bits, 23 through 18, of location B. Zeros are transmitted on lines 6 and 7 of the character. A "1" bit is automatically sent on line 8 along with a parity bit on line 9 to complete the transmission. The third data character sent contains the 6 bits, 17 through 12, of location B. Zeros are transmitted on lines 6 and 7. A "1" bit is automatically sent on line 8 along with a parity bit on line 9 to complete the transmission. The sixth data character sent contains the 8 bits, 31 through 24, of location B + 1, transmitted as before with a "1" on line 8 and a parity bit on line 9 of the transmission. This transmission procedure continues until the last data character is sent. This character contains the 6 bits, 5 through 0, of location B + N − 1. Zeros are transmitted on lines 6 and 7. The "1" data bit and the parity bit are sent as before. Upon receipt of the next REQ, an END character is transmitted. An address two greater than the address of the last location in the block is placed in the accumulator. The Next Command in sequence is skipped and the following command is executed.

If any response other than REQ is received, that response is placed in bit positions 0 through 8 of the Line Response register. The transmit operation terminates and an error, ERR, is transmitted. An address two larger than the address of the last location operated on is placed in the accumulator. The Next Command in sequence is executed.

If a parity error, time delay error, or memory overflow is encountered, the operation is terminated as shown in Table 8, page 52. In these cases an ERR will be transmitted.

When the operation is terminated in the middle of a word, a code representing the character count of that word is placed in bits 14 through 12 of the Line Response register so that the programmer may determine the number of characters transmitted (see Table 9, page 53).

8TI Block Transmit Instructions 8-bit Characters Send ${}_{31}(B)_{24}$, ${}_{23}(B)_{16}$, ${}_{15}(B)_{8}$, ${}_{7}(B)_{0}$, ${}_{31}(B+1)_{24}$ $\ldots{}_{7}(B+N-1)_{0}$,



When a Block Transmit Instruction is processed, 8-bit instructions are transmitted to the specific piece of equipment starting with the 8-bit instruction in the second word of the Block Operator command, taking the next from location B, and continuing through location B + N − 1 where N is the block length. A GRN is received from the external equipment acknowledging receipt of a proper instruction.

The first instruction sent is the 8-bit instruction in bits 15 through 22 of the second word of the Block Transmit Instruction command. A "0" bit is sent on line 8 along with a parity bit on line 9 to complete the 10-bit transmission.

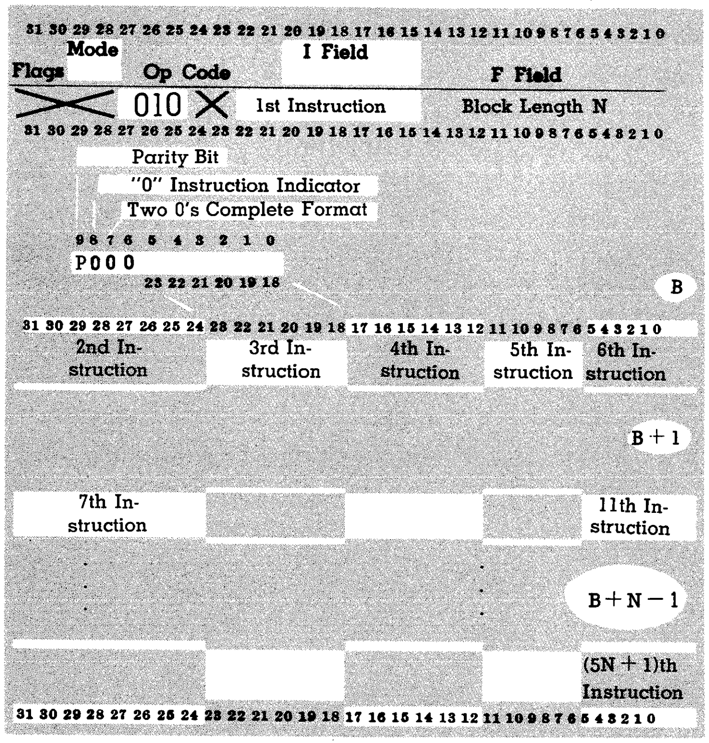
The second instruction sent contains the 8 bits, 31 through 24, of location B. A "0" bit is automatically sent on line 8 along with a parity bit on line 9 to complete the 10-bit transmission. The "0" bit indicates to the external equipment that this is an instruction to be executed. The third instruction sent contains the 8 bits, 23 through 16, of location B. A "0" bit is automatically sent on line 8 along with a parity bit on line 9 to complete the transmission. The sixth instruction sent contains the 8 bits, 31 through 24, of location B + 1. A "0" bit is sent on line 8 along with a parity bit on line 9. This transmission procedure continues until the last instruction is sent. This instruction contains the 8 bits, 7 through 0, of location B + N-1. The "0" instruction bit and the parity bit are sent as before. Upon receipt of the next GRN, no character is sent to indicate the end of transmission. An address two greater than the address of the last location in the block is placed in the accumulator. The Next Command in sequence is skipped and the following command is executed.

If any response other than GRN is received, that response is placed in bit positions 0 through 8 of the Line Response register. The transmit operation terminates but no error, ERR, is transmitted. An address two larger than the address of the last location operated on is placed in the accumulator. The Next Command in sequence is executed.

If a parity error, time delay error, or memory overflow is encountered, the operation is terminated as shown in Table 8, page 52 . In these cases, no ERR will be transmitted.

When the operation is terminated in the middle of a word, a code representing the character count of that word is placed in bits 14 through 12 of the Line Response register so that the programmer may determine the number of instructions transmitted (see Table 9, page 53).

6TI Block Transmit Instructions 6-Bit Characters — Send $_{31}(B)_{24}$, $_{23}(B)_{18}$, $_{17}(B)_{12}$, $_{11}(B)_{6}$, $_{5}(B)_{0}$, $_{31}(B+1)_{24}$, $_{23}(B+1)_{18}$, . . .$_{5}(B+N-1)_{0}$



When a Block Transmit Instruction command is processed, first the 8-bit instruction in the second word of the command, then one 8-bit and four 6-bit instructions per word are transmitted to the specified piece of equipment, starting with location B and continuing through location B + N − 1 where N is the block length. A GRN is received from the external equipment before each transmission acknowledging receipt of a proper instruction.

The first instruction sent is the 8-bit instruction in bits 15 through 22 of the second word of the Block Transmit Instruction command. A "0" bit is sent on line 8 along with a parity bit on line 9 to complete the 10-bit transmission.

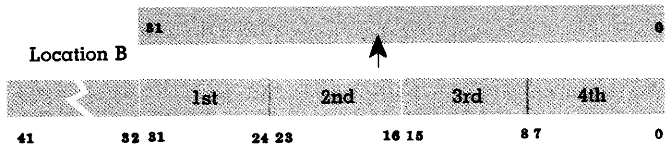
The second instruction sent contains the 8 bits, 31 through 24, of location B. A "0" bit is automatically sent on line 8 along with a parity bit on line 9 to complete the 10-bit transmission. The third instruction sent contains the 6 bits, 23 through 18, of location B. Zeros are transmitted on lines 6 and 7 to fill the instruction format. A "0" bit is automatically sent on line 8 along with a parity bit on line 9 to complete the transmission. The fourth instruction sent contains the 6 bits, 17 through 12, of location B. Zeros are transmitted on lines 6 and 7 to fill the instruction format. A "0" bit is automatically sent on line 8 along with a parity bit on line 9 to complete the transmission. The seventh instruction sent contains the 8 bits, 31 through 24, of location B + 1, transmitted as before with a "0" on line 8 and a parity bit on line 9 of the transmission. This transmission procedure continues until the last instruction is sent. This instruction contains the 6 bits, 5 through 0, of location B + N − 1. The zeros, the "0" instruction bit, and the parity bit are sent as before. Upon receipt of the next GRN, no character or instruction is sent to indicate the end of transmission. An address two greater than the address of the last location in the block is placed in the accumulator. The Next Command in sequence is skipped and the following command is executed.

If any response other than GRN is received, that response is placed in bit positions 0 through 8 of the Line Response register. The transmit operation terminates but no error, ERR, is transmitted. An address two larger than the address of the last location operated on is placed in the accumulator. The Next Command in sequence is executed.

If a parity error, time delay error, or memory overflow is encountered, the operation is terminated as shown in Table 8, page **52**. In these cases, no ERR will be transmitted.

When the operation is terminated in the middle of a word, a code representing the character count of that word is placed in bits 14 through 12 of the Line Response register so that the programmer may determine the number of characters transmitted (see Table 9, page **53**).

8RD Block Deceive Data 8-bit character — Receive in ${}_{31}(B)_{24}, {}_{23}(B)_{16}, {}_{15}(B)_{8}, {}_{7}(B)_{0}, {}_{31}(B+1)_{24}, \ldots {}_{7}(B+N-1)_{0}$



The computer will send an REQ signal to the specified piece of external equipment. It will then accept the 8-bit data characters from that piece of equipment, form machine words with them and place the machine words into memory, starting at location B and continuing through location B + N − 1 where N is the block length. Upon receipt of a data character, an REQ is sent out and the computer waits to accept the next data character. If the operation is terminated in the middle of a word by some unusual response, the word will be character-normalized and the remaining least significant positions will be filled with zeros.

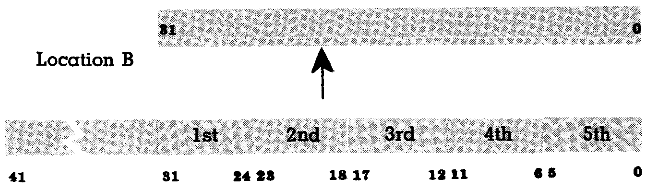
The first data character received will be placed in the least significant 8 bit positions (7 through 0) of the arithmetic unit. The REQ is sent out, the 8 bits in the arithmetic unit are shifted into positions 15 through 8, and the next 8-bit character is placed in bit positions 7 through 0. The REQ is sent out, the 16 bits are shifted into positions 23 through 8, and the next 8-bit character is placed in positions 7 through 0. The REQ is sent out, the 24 bits are shifted into positions 31 through 8 and the next 8-bit character is placed in positions 7 through 0. These 32 bits then replace the contents of location B.

This procedure continues until the last four 8-bit characters replace the contents of location B + N − 1. The computer then transmits an END character to the external device. An address one greater than the address of the last location in the block is placed in the accumulator. The Next Command in sequence is skipped and the following command is executed.

If any response other than a data character is received, that response is placed in bit positions 0 through 8 of the Line Response register. The receive operation terminates and an error, ERR, is transmitted. An address one greater than the address of the last location filled is placed in the accumulator. The Next Command in sequence is executed.

If a parity error, time delay error, or memory overflow is encountered, the operation is terminated as shown in Table 8, page 52. In these cases, ERR will be transmitted.

6RD Block Receive Data 6-bit characters — Receive in ${}_{31}(B)_{24}, {}_{23}(B)_{18}, {}_{17}(B)_{12}, {}_{11}(B)_{6}, {}_{5}(B)_{0}, {}_{31}(B+)_{24}, \ldots {}_{5}(B+N-1)_{0}$



The computer will send a REQ signal to the specified piece of external equipment. It will then accept one 8-bit and four 6-bit data characters from that piece of equipment, form a machine word with them and place it into memory in location B. Another set of one 8-bit and four 6-bit data characters is accepted and placed into memory in location B + 1. This procedure is continued until location B + N − 1 is filled. Upon receipt of each data character, a REQ is sent out and the computer waits to accept the next data character. If the operation is terminated in the middle of a word by some unusual response, the word will be character-normalized and the remaining least significant positions will be filled with zeros.

The first data character received will be placed in the least significant 8 bit positions (7 through 0) of the arithmetic unit. The REQ is sent out, the 8 bits in the arithmetic unit are shifted into positions 13 through 6, and the first 6-bit character is placed in bit positions 5 through 0. The REQ is sent out, the 14 bits are shifted into positions 21 through 6, and the next 6-bit character is placed in positions 5 through 0. The REQ is sent out, the 20 bits in the arithmetic unit are shifted into positions 27 through 6, and the next 8-bit character is placed in positions 5 through 0.

The REQ is sent out, the 26 bits in the arithmetic unit are shifted into positions 31 through 6, and the last 6-bit character of this word is placed in positions 5 through 0. These 32 bits then replace the contents of location B.

This procedure continues until the last five characters, one 8-bit and four 6-bit, replace the contents of location B + N − 1. The computer then transmits an END character to the external device. An address one greater than the address of the last location in the block is placed in the accumulator. The Next Command in sequence is skipped and the following command is executed.

If any response other than a data character is received, that response is placed in bit positions 0 through 8 of the Line Response register. The receive operation terminates and an error, ERR, is transmitted.

An address one greater than the address of the last location filled is placed in the accumulator. The Next Command in sequence is executed.

If a parity error, time delay error, or memory overflow is encountered, the operation is terminated as shown in Table 8, page 52. In these cases, ERR will be transmitted.

LINE RESPONSE REGISTER

14	Code
13	Count
12	
11	No response
10	REQ response
9	Parity error PER
8	
7	
6	Response to
5	
4	Line Instruction
3	
2	
1	
0	

TABLE 7

INPUT-OUTPUT ERROR BRANCH

Receive Data

The Block operation is terminated if a response of "parity error," a time delay error or a memory overflow is encountered.

If the response is "parity error" (PER), bit number 9 is turned ON.

If there is no response after 1 second, bit number 11 is turned ON.

An address one larger than the address of the last location operated on is placed in the accumulator. The Next Command in sequence is executed.

Transmit Data

The Block operation is terminated if a response of "parity error," a time delay error or a memory overflow is encountered.

If the response contains a parity error, bit number 9 is turned ON.

If there is no response after 1 second, bit number 11 is turned ON.

An address two larger than the address of the last location operated on is placed in the accumulator. The Next Command in sequence is executed.

Transmit Instruction

The Block operation is terminated if a response of "parity error," a time delay error or a memory overflow is encountered.

If the response contains a parity error, bit number 9 is turned ON.

If the response is REQ, but number 10 is turned ON.

If there is no response after 1 second, bit number 11 is turned ON.

An address two larger than the address of the last location operated on is placed in the accumulator. The Next Command in sequence is executed.

In all cases a memory overflow causes the operation to be terminated and an ERR is transmitted in all cases but Transmit Instruction. The illegal address indicator, bit number 0 of the Interrupt Request register, is turned ON and is processed as an internal interrupt as on page 11 .

TABLE 8

INCOMPLETE TRANSMISSION WORD CODE

Characters Remaining in the Word	The Code Representing the Corresponding Characters
1	6
2	2
3	3
4	1
5	0

TABLE 9

For example, if there are two characters remaining to be processed in a word at the termination of an 8-bit transmission, the contents of bits 14 through 12 of the Line Response register read in that order, will be the octal number 2. If there are four characters remaining to be processed in a word at the termination of a 6-bit transmission, the contents of bits 14 through 12, read in that order, will be the octal number 1.

Entries 4 and 5 are not defined for the 8-bit transmission.

STANDARD INPUT-OUTPUT INSTRUCTION CODES

Octal Code	Alpha Code	Description
000	—	Cannot be detected.
002	GRN	"Proceed" response to instructions and queries.
003	RED	"Branch" response to instructions and queries.
004	END	Signal from transmitter that a data transmission has ended. The next line signal can come only from the controlling device.
005	ERR	Signal from transmitter that the data transmission which has just ended was erroneous. The next line signal can come only from the controlling device.
010	SDT	Start Data Transfer.
011	OUT	Command to switch to the OFF-LINE state.
014	RCV	Command to switch to the INSTRUCTED state and prepare to receive a data block.
016	TRA	Command to switch to the INSTRUCTED state and prepare to transmit a data block.
060	QRD	Ready query: GRN response means "proceed"; RED response means "not ready for additional commands."
061	QER	Error: GRN response means that the device has not detected an error since the previous QER; RED indicates that an error has been detected.
062	QIL	Interlock query: GRN response means "proceed"; RED response means the device is halted because of an interlock condition.
067	QIN	Interrupt query: GRN response means that the device has not sent an interrupt request since the previous QIN; Red indicates that an interrupt has been sent.

APPENDIX

AVERAGE EXECUTION TIMES FOR G-20 CENTRAL PROCESSOR

The execution times given in the following tables include the average amount of time necessary to:

bring the command from memory

assemble the operand B, and

decode and execute the command.

All times are given in microseconds.

Command	Command Function	Address Mode 0 No Index	Address Mode 0 One Index	Address Mode 1 No Index	Address Mode 1 One Index	Address Mode 2 No Index	Address Mode 2 One Index	Address Mode 3 No Index	Address Mode 3 One Index
SINGLE PRECISION									
Address Preparation Commands									
OCA	Clear and Add to OA	9	15	15	21	15	21	21	27
OCS	Clear and Subtract from OA	9	15	15	21	15	21	21	27
OAD	Add in OA	13	19	19	25	15	21	21	27
OSU	Subtract in OA	13	19	19	25	15	21	21	27
ORS	Reverse Subtract in OA	13	19	19	25	15	21	21	27
OSS	Add & Reverse Sign in OA	13	19	19	25	15	21	21	27
OAV	Add & Take Ab-13 solute Value in OA	13	19	19	25	15	21	21	27
OSV	Subtract & Take Absolute Value in OA	13	19	19	25	15	21	21	27
Add-Subtract Commands (floating-point)									
CA	Clear & Add	6	12	12	18	12	18	18	24
CS	Clear & Subtract	6	12	12	18	12	18	18	24
AD	Add	12	18	18	24	14	20	20	26
SU	Subtract	12	18	18	24	14	20	20	26
RS	Reverse Subtract	13	19	19	25	15	21	21	27
SS	Add & Change Sign	13	19	19	25	15	21	21	27
AV	Add & Take Absolute Value	13	19	19	25	15	21	21	27
SV	Subtract & Take Absolute Value	13	19	19	25	15	21	21	27

Command	Command Function	Address Mode 0 No Index	Address Mode 0 One Index	Address Mode 1 No Index	Address Mode 1 One Index	Address Mode 2 No Index	Address Mode 2 One Index	Address Mode 3 No Index	Address Mode 3 One Index
Multiply-Divide Commands (floating-point)									
ML	Multiply	39	45	45	51	45	51	51	57
DI	Divide	74	80	80	86	80	86	86	92
RD	Reverse Divide	75	81	81	87	81	87	87	93
Add-Subtract Test Commands (floating-point)									
TP	Transfer on Positive	16	22	22	28	18	24	24	30
TN	Transfer on Negative	16	22	22	28	18	24	24	30
TL	Transfer on Contents of Accumulator are Less than Operand B	16	22	22	28	18	24	24	30
TG	Transfer on Contents of Accumulator Greater than Operand B	16	22	22	28	18	24	24	30
TU	Transfer on Contents of Accumulator not Equal to Operand B	16	22	22	28	18	24	24	30
T1	Transfer on Sum Negative	16	22	22	28	18	24	24	30
T2	Transfer on Sum Positive	16	22	22	28	18	24	24	30
T3	Transfer on Sum Not Equal to Zero	16	22	22	28	18	24	24	30
Add-Subtract Commands (Pickapoint)									
CA	Clear & Add	6	12	12	18	12	18	18	24
CS	Clear & Subtract	6	12	12	18	12	18	18	24
AD	Add	10	16	16	22	12	18	18	24
SU	Subtract	10	16	16	22	12	18	18	24
RS	Reverse, Subtract	11	17	17	23	13	19	19	25
SS	Subtract, Subtract	11	17	17	23	13	19	19	25
AV	Add & Take Absolute Value	11	17	17	23	13	19	19	25
SV	Subtract & Take Absolute Value	11	17	17	23	13	19	19	25
Multiply-Divide Commands (Pickapoint)									
ML	Multiply	30	36	36	42	36	42	42	48
DI	Divide	70	76	76	82	76	82	82	88
RD	Reverse Divide	71	77	77	83	77	83	83	89

Command	Command Function	Address Mode 0 No Index	Address Mode 0 One Index	Address Mode 1 No Index	Address Mode 1 One Index	Address Mode 2 No Index	Address Mode 2 One Index	Address Mode 3 No Index	Address Mode 3 One Index
EXTENDED PRECISION									
Add-Subtract Commands (floating-point)									
CA	Clear and Add			18	24	18	24	24	30
CS	Clear and Subtract			18	24	18	24	24	30
AD	Add			24	30	20	26	26	32
SU	Subtract			24	30	20	26	26	32
RS	Reverse Subtract			25	31	21	27	27	33
SS	Add & Change Sign			25	31	21	27	27	33
AV	Add & Take Absolute Value			25	31	21	27	27	33
SV	Subtract & Take Absolute Value			25	31	21	27	27	33
Multiply-Divide Commands (floating-point)									
ML	Multiply			51	57	51	57	57	63
DI	Divide			86	92	86	92	92	98
RD	Reverse Divide			87	93	87	93	93	99
Add-Subtract Test Commands (floating-point)									
TP	Transfer on Positive			28	34	24	30	30	36
TN	Transfer on Negative			28	34	24	30	30	36
TL	Transfer on Contents of Accumulator Less than Operand B			28	34	24	30	30	36
TG	Transfer on Contents of Accumulator Greater than Operand B			28	34	24	30	30	36
TU	Transfer on Contents of Accumulator not Equal to Operand B			28	34	24	30	30	36
T1	Transfer on Sum Negative			28	34	24	30	30	36
T2	Transfer on Sum Positive			28	34	24	30	30	36
T3	Transfer on Sum Not Equal to Zero			28	24	24	30	30	36
SINGLE PRECISION									
Non-numeric Commands									
The quantities involved in these commands are assumed to be integers.									
BD	Bring	10	16	16	22	12	18	18	24
BC	Bring Complement	10	16	16	22	12	18	18	24
AL	Logical Add	10	16	16	22	12	18	18	24

Command	Command Function	Address Mode 0 No Index	Address Mode 0 One Index	Address Mode 1 No Index	Address Mode 1 One Index	Address Mode 2 No Index	Address Mode 2 One Index	Address Mode 3 No Index	Address Mode 3 One Index
SL	Logical Subtract	10	16	16	22	12	18	18	24
XX	Extract	10	16	16	22	12	18	18	24
XC	Extract Complement	10	16	16	22	12	18	18	24
UU	Unite	10	16	16	22	12	18	18	24
UC	Unite Complement	10	16	16	22	12	18	18	24
Non-numeric Test Commands									
The quantities involved in these commands are assumed to be integers.									
JO	Jump on Zeros	14	20	20	26	16	22	22	28
J1	Jump on Ones	14	20	20	26	16	22	22	28
JG	Jump on Difference not Equal to Zero	14	20	20	26	16	22	22	28
JS	Jump on Sum not Equal to Zero	14	20	20	26	16	22	22	28
JX	Jump on Extract	14	20	20	26	16	22	22	28
XJ	Jump on Extract Complement	14	20	20	26	16	22	22	28
JU	Jump on Unite	14	20	20	26	16	22	22	28
UJ	Jump on Unite Complement	14	20	20	26	16	22	22	28
Putaway Commands									
PF	Putaway Extended Precision	21	27	27	33	27	33	33	39
PH	Putaway Single Precision	18	24	24	30	24	30	30	36
PI	Putaway Integer (Minimum Time)	14	20	20	26	20	26	26	32
PL	Putaway Logic	12	18	18	24	18	24	24	30
PZ	Putaway Zero	12	18	18	24	18	24	24	30
Control Commands									
GO	Go to Location B	6	12	12	18	12	18	18	24
GE	Go & Enable	6	12	12	18	12	18	18	24
SK	Go to Location (NC) + B	8	14	14	20	14	20	20	26
MT	Mark Place & Transfer	14	20	20	26	20	26	26	32
Index Commands									
LP	Load Index Positive	13		19		19		25	
LN	Load Index Negative	13		19		19		25	
IN	Increment Index	20		26		26		32	
DE	Decrement Index	20		26		26		32	
PT	Load Positive and Test	13		19		19		25	
NT	Load Negative and Test	13		19		19		25	

Command	Command Function	Address Mode 0 No Index	Address Mode 0 One Index	Address Mode 1 No Index	Address Mode 1 One Index	Address Mode 2 No Index	Address Mode 2 One Index	Address Mode 3 No Index	Address Mode 3 One Index
IT	Increment Index and Test	20		26		26		32	
DT	Decrement Index and Test	20		26		26		32	
Register Commands									
LR	Load Register	7		13		13		19	
XR	Selectively Reset Register I	11		17		17		26	
XO	Selectively Read Register I to OA	11		17		17		26	
XA	Selectively Read Register I to Accumulator	11		17		17		26	
Repeat Commands									
RCA, RCS, RTP, RTN		13+6N	19+6N	19+6N	25+6N	19+6N	25+6N	25+6N	31+6N
RJU, RUJ		15+9N	21+9N	21+9N	27+9N	21+9N	27+9N	27+9N	33+9N
RJ0, RJ1, RJG, RJS, RJX, RXJ		14+8N	20+8N	20+8N	26+8N	20+8N	26+8N	26+8N	32+8N
All Other Repeat Commands		13+7N	19+7N	19+7N	25+7N	19+7N	25+7N	25+7N	31+7N

N is the block length in words.

Input-Output Commands									
LC	Line Command	21	27	27	33	27	33	33	39
DC	Data Character	21	27	27	33	27	33	33	39
Block	8-bit Command	26+20N	32+20N	32+20N	38+20N	32+20N	38+20N	38+20N	44+20N
Block	6-bit Command	26+15N	32+15N	32+15N	38+15N	32+15N	38+15N	38+15N	44+15N

N is the block length in words.

Bendix Computer Division
LOS ANGELES 45, CALIFORNIA

THE Bendix CORPORATION

T23-1
DECEMBER, 1960

PRINTED IN U.S.A.