RECOMP III

GENERAL PURPOSE DIGITAL COMPUTER

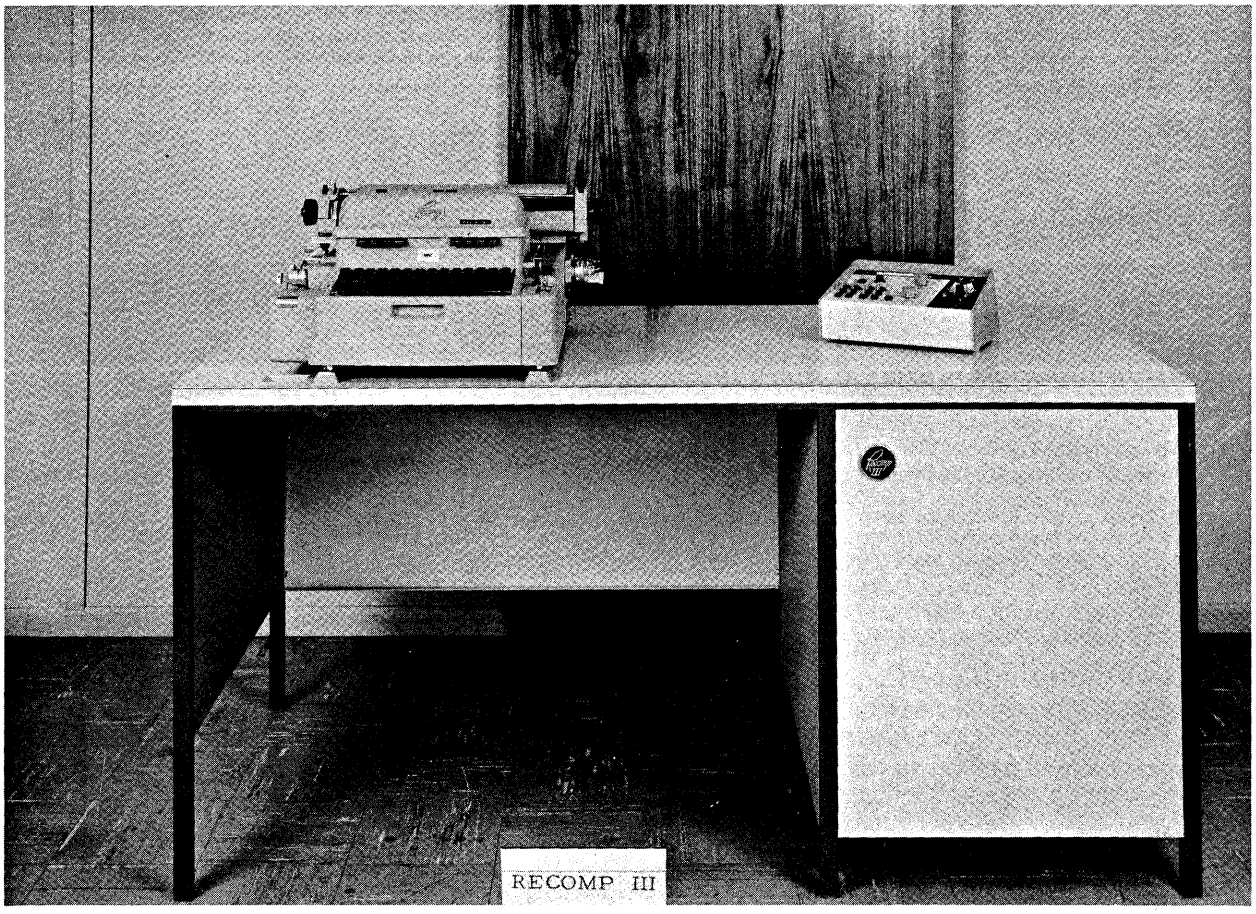
PROGRAMMING MANUAL

First Printing - August, 1961
Reissued - April, 1963
Revised and Reissued - December, 1964

CONTENTS

CONTENTS - Continued

ILLUSTRATIONS: Page


RECOMP III

I. INTRODUCTION

RECOMP III is a solid-state, general purpose, digital computer designed for scientific and industrial use. It has a 4096 - word memory with a 40-bit word length. Each word contains two program instructions, therefore, the memory can hold a program with over 8000 instructions. RECOMP III offers a large command list of 49 commands, or, when optional floating point hardware is used, 53 commands. Its Index Register makes machine language programming a simple task for any engineer or technical person.

RECOMP III will accommodate up to four inputs and four outputs, and will handle 5 through 8 channel tape. Its equipment is easily accessible for servicing, and printed circuitry reduces maintenance to a minimum. The computer operates from any standard outlet.

Operational capabilities and built-in reliability of RECOMP III have proven its value in any engineering or scientific office where results have to be computed and obtained quickly.

II. BASIC COMPUTER DESCRIPTION

The basic computer (Figure 1) consists of three assemblies: memory, Flexowriter, and control console. The operator is concerned only with the Flexowriter and control console.

MEMORY

The memory is a magnetic disk containing basic timing channels, arithmetic registers, rapid access loops, and area for information storage. It will hold 4096 40-bit words, with two instructions per word, or a total of 8192 internally stored instructions -- a capacity comparable to that of many large-scale computing systems.

In operation, the memory rotates at 3450 rpm. The disk is coated with ferrous oxide similar to that on conventional magnetic recording tape. As the disk rotates past each stationary recording (write) head, a magnetic signal is recorded into the oxide coating, and remains until replaced by new information. To extract information from the disk, a reproducing (read) head similar to the recording head is used. Extraction of information from the memory does not change its contents.

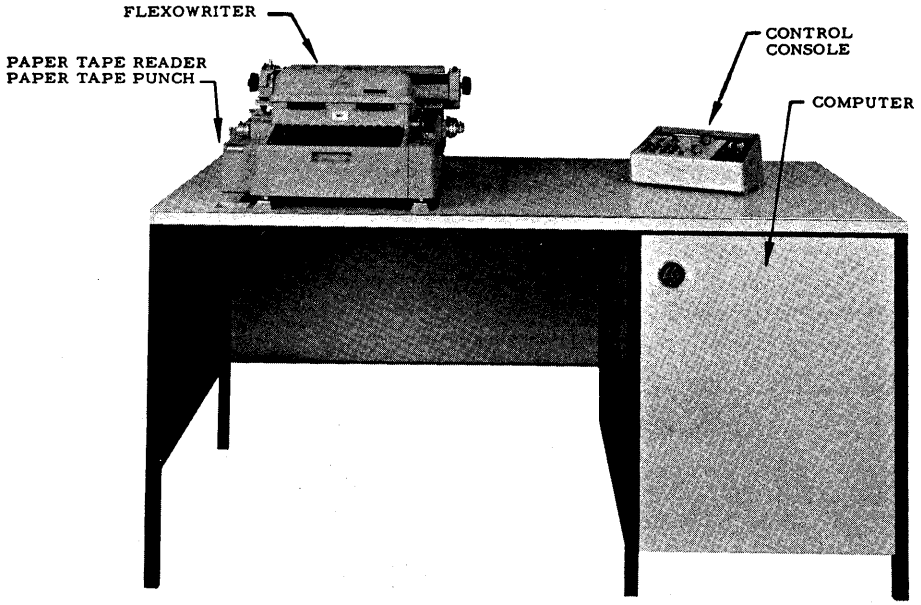


Figure 1. RECOMP III Computer Components

The disk memory is divided into two portions: the main memory and the high-speed memory.

Main-Memory

The main memory holds 4080 40-bit words. It is nonvolatile, protecting stored data from erasure unless erasure is desired. Information in the main memory will not be destroyed during power interruption or power-off condition.

Words in the main memory are addressed by four octal digits from 0000 to 7757. Average access time is 9.3 milliseconds.

The sector tract, an inaccessible portion of the main memory, contains permanently recorded bootstrap and diagnostic routines. These built-in programs facilitate program loading and easy, fast computer checkout.

High-Speed Memory

Rapid access to stored information is provided by two high-speed recirculating loops, L and V, of eight words each. Available in programming, the L and V loop addresses are indicated by octal 7760 to 7767, and by 7770 to 7777, respectively. One to eight words can be transferred by option to the L and V loops, and by option from the L loop. Information contained in the loops is volatile.

Average access time to the high-speed memory is 1.8 milliseconds.

CONTROL CONSOLE

The control console provides fast, simple communication between the operator and computer equipment. (See Figure 2.) Its few switches and compact design permit operation after a brief instruction period.

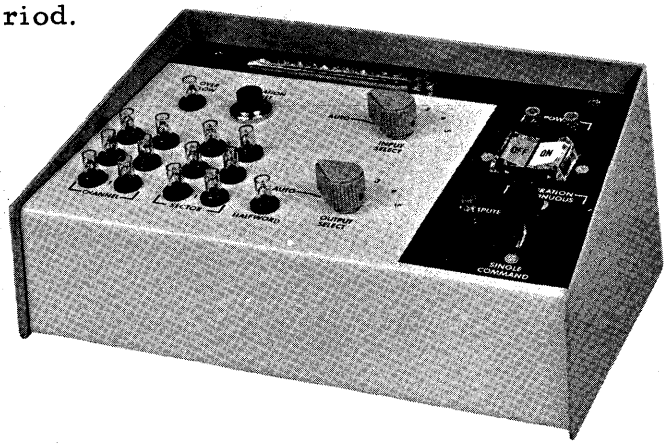


Figure 2. Control Console

Switches on the console are: POWER ON-OFF, COMPUTE, INPUT SELECT, OUTPUT SELECT, and LOCATION RESET. Indicator lights on the console are: ON, READY, COMPUTE, OVERFLOW, and 13 location indicators.

The ON light appears when the power switch is turned ON. The READY light appears approximately 45 seconds after the power is turned on. At this time, the computer is ready for operation.

The COMPUTE switch has 3 positions: CONTINUOUS, to start the computer in continuous operation under program control; HALT, to stop computation; and SINGLE COMMAND, to execute one command at a time.

The COMPUTE light is on during all computations.

The OVERFLOW light appears when an overflow condition is detected. It does not halt computation. The OVERFLOW light must be turned off by program control.

The 13 location indicator lights display the octal location of the next instruction to be executed.

The LOCATION RESET switch sets the location counter to 0000.0; i.e., when computation is started, the next instruction will be taken from 0000.0.

The INPUT SELECT switch provides a means for operator selection of one of four input devices. When this switch is placed on "automatic", the program has control of the input. The operator may override programmed input control by selecting some alternate device, such as the Flexowriter keyboard or tape reader, a FACITAPE reader, or a card reader, etc.

The OUTPUT SELECT switch provides a means for operator selection of one of four output devices. When this switch is placed on "automatic", the program has control of the output. The operator may override programmed output control by selecting some alternate device, such as the Flexowriter typewriter, or tape punch, a FACITAPE punch, or a card punch, etc.

REGISTERS

In addition to the main memory and the high-speed memory, the RECOMP III computer unit includes five recirculating registers: A-register, R-register, G-register, B-register, and the C-register. In conjunction with appropriate switching and control

elements, these registers are devices for retaining information and carrying out basic arithmetic and logical operations.

The five registers each contain one word. The action and processing of information within them controls and processes all data. Contents of the various recirculating registers are not retained when power to the computer is off.

A-Register:

The most important register is the A (or Accumulator) register, used in all arithmetic operations. This register holds the results of arithmetic and logical operations and input/output instructions.

R-Register:

The R-register, often termed the lower accumulator or remainder register, acts as temporary storage for the remainder in a division, or for the least significant half of a product obtained in multiplication. It may be considered in this way an extension of the A-register. It is also used to extend the range of the divident in the DIVIDE command, and permits the accumulation of a double length product resulting from the MULTIPLY command.

B-Register:

The B-register, often referred to as the number register, is an intermediate storage register which is not directly addressable by the programmer. It holds the number or operand whose address is found in the command.

C-Register:

This is the command register from which all instructions are executed. Address modification directly affects the operand address of the current instruction under execution within the C or command register.

G-Register:

The left-half of the G-register contains the location counter (the current memory address for input control or instruction execution); and the right-half of the G-register contains the index register.

INDEX REGISTER

One of the major advantages of the RECOMP III over other low-cost computers is its built-in index register. This register, under programmed control, modifies addresses and controls the number of times a given set of instructions will be executed. It allows a set of instructions to be repeated at very high speeds, using new data each time. This register can cut programming time by 35 to 50 per cent, thus substantially lowering operating costs.

OPTIONAL FLOATING POINT HARDWARE

One of the most tedious tasks encountered by a programmer is the problem of scaling. Scaling problems can be reduced substantially by the use of floating point. Two approaches are available for obtaining floating point capabilities in RECOMP III:
(1) Floating point logic can be simulated by the use of programs designed for this purpose. Because of the machine time and space required for this technique, it is considered to have limited use.
(2) The RECOMP III optional floating point hardware can be, and should be, obtained when the application of floating point programs is to be at all extensive.

III. INPUT - OUTPUT

A Flexowriter is used for input/output operations with the basic RECOMP III computing unit. (See Figure 3.) For increased input/output capacity, a FACITAPE reader/punch console may be attached to the computer as an optional device.

FLEXOWRITER

Information can be entered into the computer either by typing in commands and data on the Flexowriter keyboard, or by entering through the reader commands and data which were prepared previously on punched paper tape. (See Figures 3 and 4.) Results can be either typed out or punched. The speed of input and output is 10 characters per second.

FACITAPE

The FACITAPE console, housed in a handsome cabinet, offers input/output facilities which are considerably faster than the standard Flexowriter on RECOMP III. (See Figure 5.) Whenever special data handling is required, the use of FACITAPE punch and reader accessories is recommended.

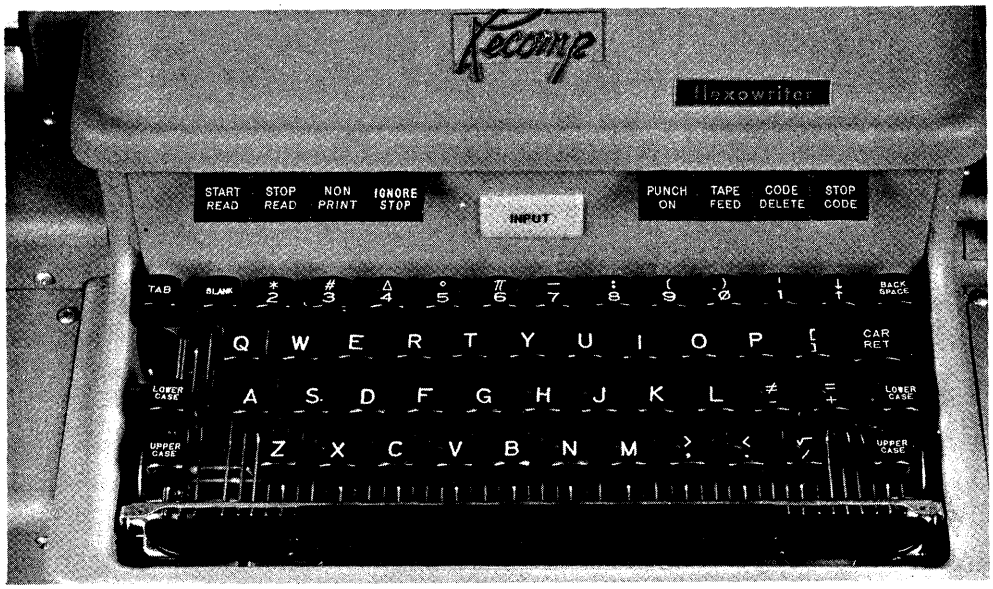


Figure 3. Flexowriter Keyboard

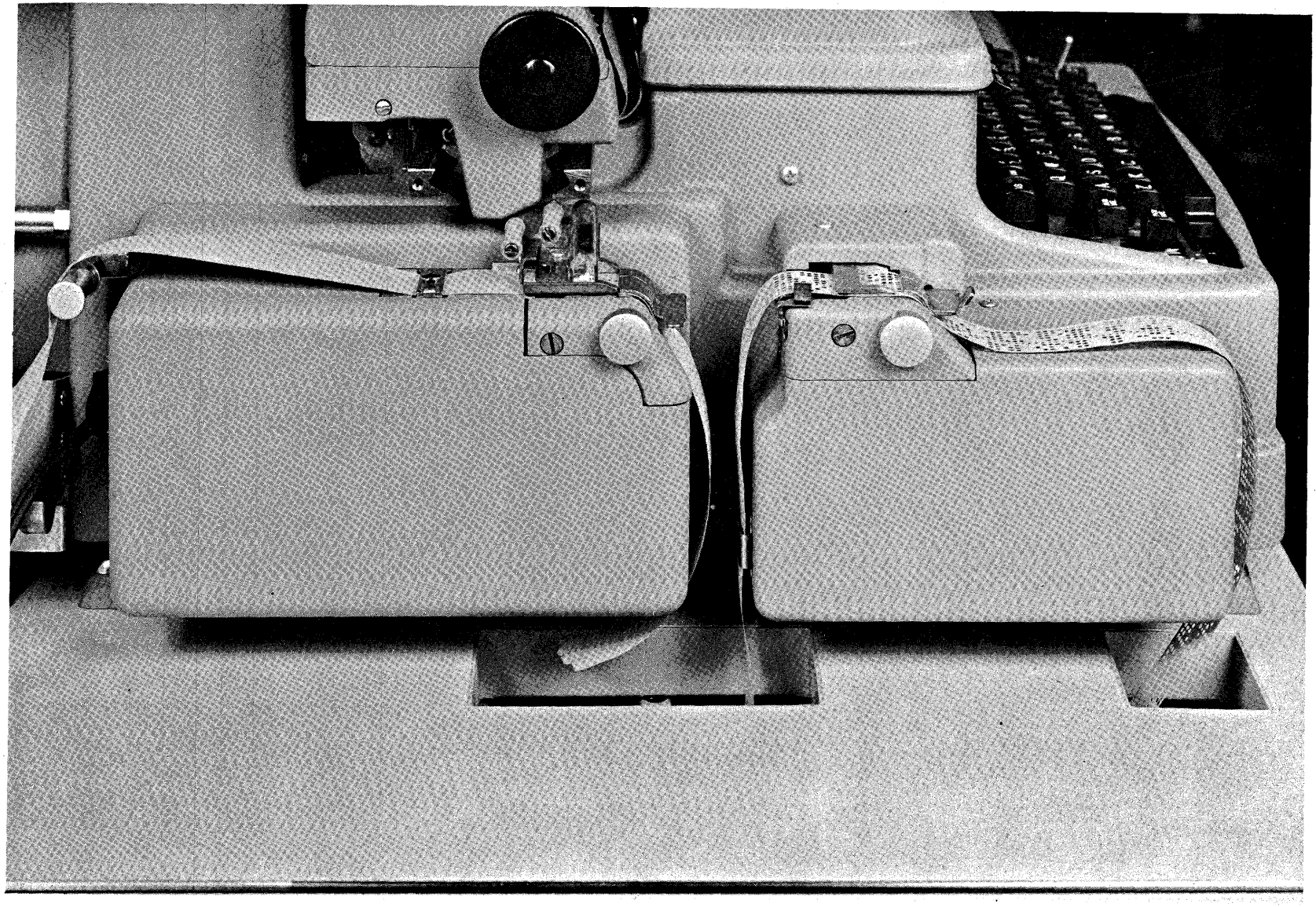


Figure 4. Flexowriter Tape Punch and Tape Reader

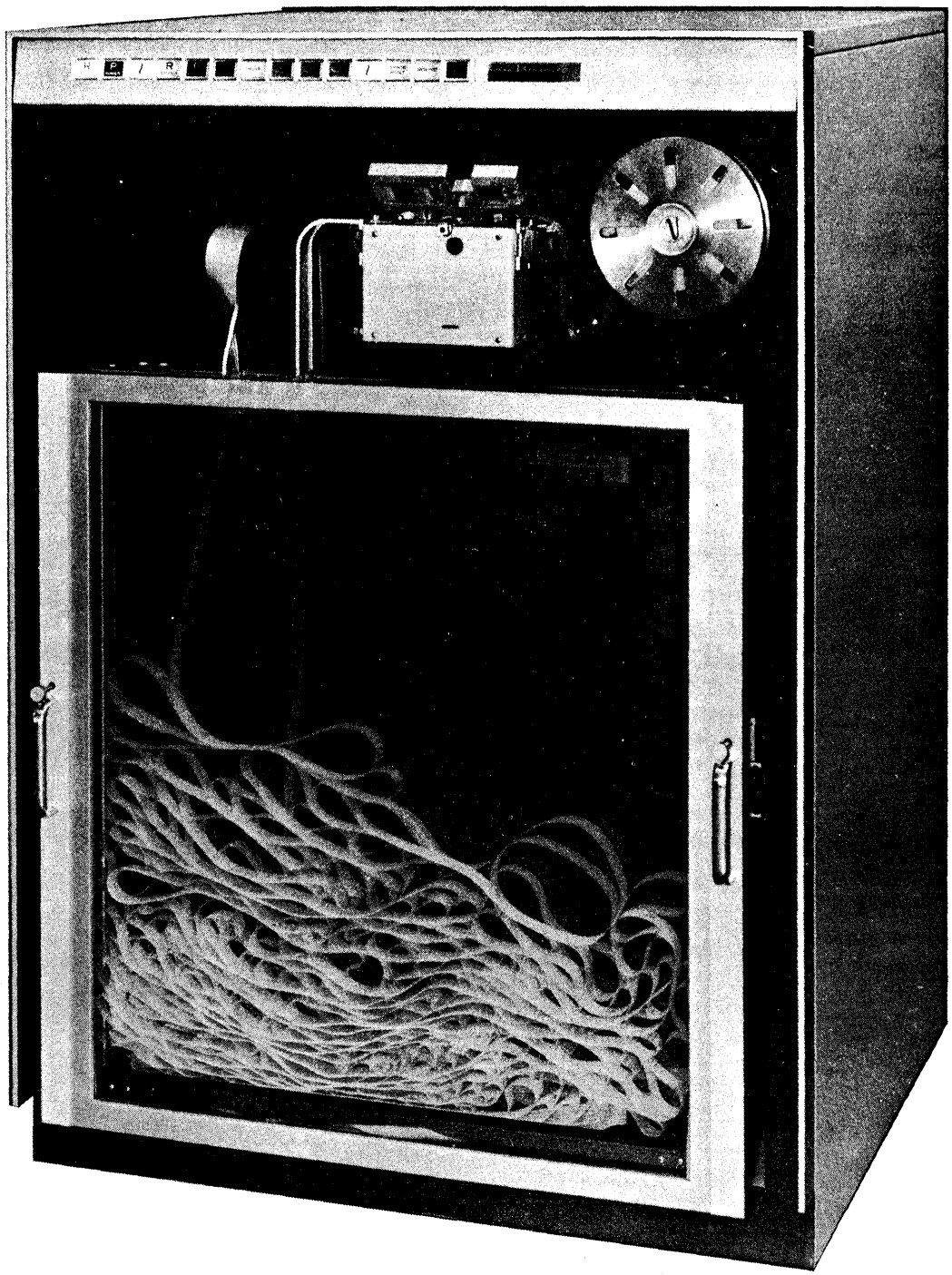


Figure 5. Facitape Console

FACITAPE High-Speed Tape Reader

This high-speed tape reader has a read rate of 600 characters per second. Its positive braking method stops tape within a character. It is a capacitance reader light.

The FACITAPE reader will handle any color and all types of paper tape. It is inserted or removed simply by lifting the hinged cover.

FACITAPE Hign - Speed Tape Punch

This high-speed heavy-duty punch can operate at the rate of 150 characters per second.

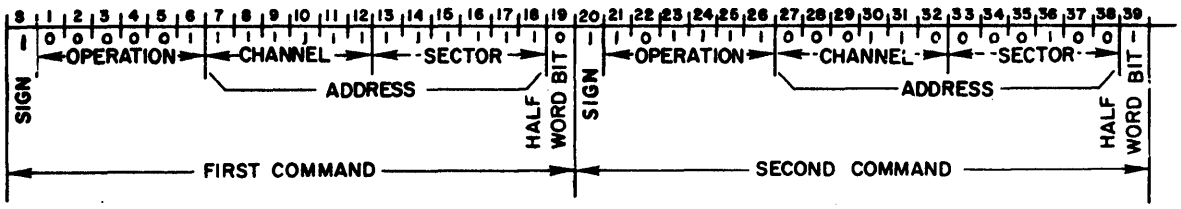
OTHER INPUT/OUTPUT

Up to four input and output devices can be attached to the RECOMP III computer, including plotters, A-D converters, D-A converters and other devices.

IV. PROGRAMMING

COMMAND STRUCTURE

The computer executes commands sequentially unless a transfer command is encountered. There are two commands per word as shown below.



In general, a command pair (word) is copied into the command C-register; the left command is executed; then the right command is executed; and then the next command pair is copied into the command C-register, etc. (The Location Counter is increased by one-half word for each command.)

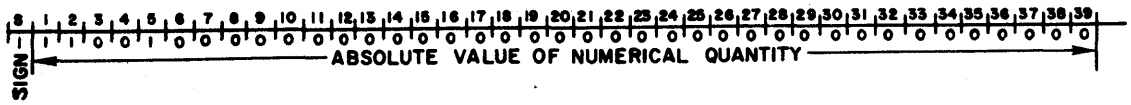
The sign of a left-half command is the actual sign of the word. This sign is ignored by the computer. The effective sign of a right-half command is the 20th bit. (See diagram above.) If the sign of a right-half command is minus (a 0 bit), the contents of the Index Register will be subtracted from the address portion of the command as the command pair is being copied into the command register.

The 6 bits of the operation tell the computer what to do, and the channel and sector portions of the command tell the computer where to get the data or operand. The half-word bit tells the computer which half-word to refer to (0 for left-half---1 for right-half) in transfer commands and a few others. The half-word bit is ignored in most commands.

DATA STRUCTURE

Data words are identical to command pairs in that they consist of 39 bits plus a sign bit, but they are interpreted by the computer as shown in the following diagram.

DATA WORD



The binary point (analogous to decimal point) may be thought of as lying between the sign and first bit (b = 0) such that all numbers are fractions less than 1; or it may be thought of as lying after the 39th bit, such that all numbers are integers; or (preferably) as lying in the position which would give the number its true (unscaled) value (analogous to a desk calculator).

(The programmer usually keeps track of the binary points of his data and results in the "remarks" column of his coding sheet.)

If the binary point is numbered from 0 (left end of word) to 39 (right end of word), some simple rules suffice to determine the binary point of the result of arithmetic operations. (Note that binary points less than 0 and greater than 39 are permissible.) A small number such as .0000010110 would "fit" at b = -5 since it has no integral part and has 5 leading zeros in the fractional part. (The absolute value of a number at a binary point of b is less than 2^b.)

These rules are as follows:

(1) In addition and subtraction, the binary points of the two operands must be the same. Shift commands are provided to allow the binary points to be lined up.

(2) In multiplication, the binary point of the product is equal to the sum of the binary points of the two operands; e.g., A b = 20 x B b = 10 = AB b = 30; A b = -5 x B b = 50 = AB b = 45 (six bits to the right of the "A" register in the "R" register).

(3) In division, the binary point of the quotient is equal to the binary point of the dividend minus the binary point of the divisor; e.g., A b = 30 / B b = $\frac{A}{B}$ b = 20.

The binary point of the remainder (in the "R" register) is equal to the binary point of the dividend minus 39 (or to the original binary point of the dividend counted from the left end of the "A" register).

COMMAND FORMAT

Commands are usually written in a form which uses octal (three bits at a time) for the operation and address and binary for the half-word bit. For example: +7312340+7243210 means,

ADD the word in channel 12 word 34 (half-word bit zero)
SUB the word in channel 43 word 21 (half-word bit zero)

A program exists which will accept this form of coding and convert it to binary for storage in memory. (R3P-1, see Appendix IV)

LIST OF COMMANDS

The operation codes usually consist of two octal digits each. However, some operation codes must have a one in the highest, next to highest, or half-word bit, of the address. These will be listed as XX.4, XX.2, or XX.1, respectively.

Registers not mentioned are not affected. No source is ever affected (non-destructive readout). The half-word bit is ignored unless mentioned. The A, R, and Index Registers will be referred to as A, R, and I.

Mnemonic	Octal	Description
CLA	37	Clear and add. Replace the contents of A with the contents of the word addressed.
CLS	36	Clear and subtract. Replace the contents of A with the negative of the contents of the word addressed.
RCA	77	Replace the contents of R with the contents of A. Then replace the contents of A with the contents of the word addressed.
RCS	76	Replace the contents of R with the contents of A. Then replace the contents of A with the negative of the contents of the word addressed.
ADD	73	Add arithmetically the contents of the word addressed to the contents of A and put the sum in A. If overflow occurs, the overflow indicator will be turned ON, and the sign of A will be reversed.

Mnemonic	Octal	Description
SUB	72	Subtract arithmetically the contents of the word addressed from the contents of A and put the difference in A. If overflow occurs, the overflow indicator will be turned ON, and the sign of A will be reversed.
MPY	63	Multiply the contents of A by the contents of the word addressed, and replace the contents of A and R with the product. Both A and R will have the sign of the product.
DIV	66	Divide the contents of A and R (dividend has sign of A) by the contents of the word addressed and replace the contents of A by the quotient and the contents of R by the remainder (with original sign of A). If overflow occurs, the overflow indicator will be turned ON.
ALS	02	Shift the contents of A left the number of places in the least significant bit of the channel portion and the 6 bits of the sector portion of the address (0 to 127 decimal). Bits leaving the left end of A are lost. Bits entering the right end of A will be zeros. The sign position of A is neither shifted nor affected.

NOTE: All shift commands utilize the same seven bits to indicate the amount of shift, either 12-18 for a left-half command or 32-38 for a right-half command.

ASC	02.4	Shift the contents of A left the amount of shift specified or until A is normalized (the left-most bit of A is a one), whichever occurs first. Decrement I by the number of places actually shifted.
ASV	02.2	Shift the contents of A left the amount of shift specified. Turn the overflow indicator on if any of the bits lost from the left end of A was a one.

NOTE that the command 02.6, which has ones in both of the two high positions of the address, acts as an 02.4 command since no overflow can occur.

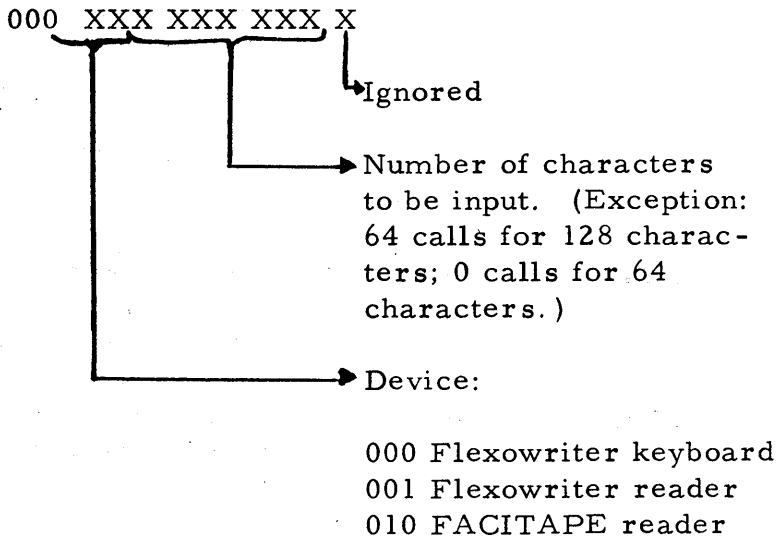
Mnemonic	Octal	Description
ARS	03	Shift the contents of A right the amount of shift specified. Bits leaving the right end of A are lost. Bits entering the left end of A will be zeros.
XAR	56	Exchange the contents of A with the contents of R.
EXT	70	Extract (bitwise logical and). Compare the contents of A with the contents of the word addressed bit by bit, including the sign position. Whenever the corresponding bits of A and the word addressed are both 1's, leave the 1 in A. Whenever either of the corresponding bits of A and the word addressed is a zero, put a zero into that position of A. (A plus sign is a one; a minus sign is a zero.)
TRA	51	Transfer. Take the next command from the half-word specified by the address instead of sequentially. A half-word bit of 1 indicates the right-half command.
TPL	55	If the sign of A is positive, take the next command from the half-word specified in the address.
TMI	53	If the sign of A is negative, take the next command from the half-word specified in the address.
TOV	52	If the overflow indicator is on, turn it off and take the next command from the half-word specified in the address.
HTR	71	HALT. When computing is resumed, take the next command from the half-word specified in the address.

Mnemonic	Octal	Description
TZE	50	If the contents of A are equal to zero (plus or minus), take the next command from the half-word specified in the address.
TNZ	11	If the contents of A are not equal to zero, take the next command from the half-word specified in the address.
TLB	15	If the lowest order (rightmost) bit of A is 1, take the next command from the half-word specified in the address.
STO	45	Replace the contents of the word addressed by the contents of A.
STR	05	Replace the contents of the word addressed by the contents of R.
STA	65	Replace the address portion of the half word addressed by the corresponding address portion of A.
LLS	42	Shift the contents of A and the contents of R left the amount of shift specified. Make the sign of A the same as the sign of R. Bits leaving the left end of A are lost. Bits entering the right end of R will be zeros. Bits leaving the left end of R enter the right end of A.
LCS	42.4	Shift the contents of A and the contents of R left the amount of shift specified or until A is normalized (the leftmost bit of A is a 1), whichever occurs first. Decrement I by the number of places actually shifted. Make the sign of A the same as the sign of R.
LSV	42.2	Shift the contents of A and the contents of R left the amount of shift specified. Turn the overflow indicator ON if any of the bits lost from the left end of A was a 1. Make the sign of A the same as the sign of R.

Mnemonic	Octal	Description
CTL	40	Replace the words in the L loop, starting with the word whose least significant address digit is the same as that of the address, with consecutive words starting with the word addressed, until 7767 is filled. For example:
		CTL 12300 will replace words 7760-7767 with words 1230-1237.
		CTL 12360 will replace words 7766 and 7767 with words 1236 and 1237.
		CTL 12370 will replace word 7767 with word 1237.
CTV	41	Replace words in the V loop until word 7777 is filled. This command is completely analogous to CTL 40 (above).
CFL	44	Replace words in memory from the L loop. Analogous to CTL 40.

NOTE: Copy to or from L commands with loop addresses (7760-7777) will refer to the so-called "gray" area which is the last 16 words of channel 77. The CTV command may not be used with a loop address. It copies into V, but from main memory.

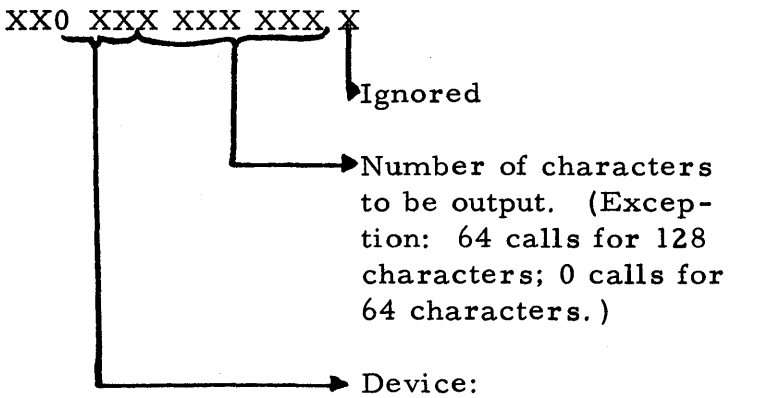
Mnemonic	Octal	Description
ICH	00	Input from 1 to 128 8-bit characters (all but the last 5 are lost) into A from the input device named in the address (or set into the input switch on the console). The console switch overrides the device selected in the address, so it should be set to AUTO to allow the program to specify its input device(s). The form of the address is as follows:



As each character is input into the right-most 8 bits of A, A is shifted left 8 bits through (including) the sign position. Since the typewriter keyboard uses a 6 bit code, the high 2 bits of each character input from it will be zeros. Either of the tape readers will read 8 bits per character with a hole entering as a one and no-hole as a zero.

(If the first 2 bits of the address are not zeros, the first character input will have its low 2 bit(s) forced to ones corresponding to the one(s) in the address.)

Mnemonic	Octal	Description
OCH	01	Output from 1 to 128 characters from A to the output device named in the address (or set into the output switch on the console). The console switch overrides the device selected in the address, so it should be set to AUTO. to allow the program to specify its output device(s). The form of the address is as follows:



As each character is output from the sign and left-most 7 bits of A, A is shifted left 8 bits and the 8 bits of the last (not the current) character output enter the right end of A. After output of the first character, these bits will be 001001XX, where XX are the first 2 bits of the address. In general, any output of over 5 characters will be meaningless, although 5 out of every 6 characters will be the original A register contents.

In output to the Flexowriter keyboard, the 2 high bits of each character are ignored. Characters with no key to represent them will type as blanks.

LRS	43	Shift the contents of A and the contents of R right the amount of shift specified. Make the sign of R the same as the sign of A. Bits leaving the right end of R are lost. Bits entering the left end of A are zero. Bits leaving the right end of A enter the left end of R.

Mnemonic	Octal	Description
NOP	54	No operation. This command has no effect.
CAZ	16	Replace the contents of A with a positive zero.
CMZ	16.4	Replace the contents of A with a negative zero.
CSA	13	Reverse the sign of A.
OVN	13.4	Turn overflow ON.
SAP	12	Make the sign of A positive.
SAN	12.4	Make the sign of A negative.
CAR	57	Replace the contents of R with the contents of A.
CMP	17	Complement A. (Reverse every bit of A including the sign.)
RND	17.4	If the highest order bit of R is 1, increase the magnitude of A by 1 in the least significant position. If overflow occurs, the overflow indicator will be turned ON.
CMG	35	Compare the contents of A with the contents of the word addressed. Turn the overflow indicator ON if the word addressed is arithmetically greater than the contents of A. Turn the overflow indicator OFF if the word addressed is arithmetically less than the contents of A. If the word addressed and the contents of A are both positive and equal, do nothing. If the word addressed and the contents of A are both negative and equal, reverse the overflow indicator. For the purpose of this command, plus zero is considered to be greater than minus zero.

Mnemonic	Octal		Description
CME	35. 1		If the contents of A are not equal to the word addressed, turn the overflow indicator ON. For the purpose of this command, plus zero does not equal minus zero. Note that, although the half-word bit of the address is 1, this command refers to a full word.
LDI	31. 1		Replace the contents of the index register with the right-hand address portion of the word addressed. Note that if the half-word bit of the address of this command is a zero, the command acts as a NOP.
SLC	25	R	Replace the left-hand address portion of the word addressed with the complement of the location counter (which will contain the location of this command).
STI	25. 1		Replace the right-hand address portion of the word addressed with the contents of the index register.
TIX	10		Subtract 1 at b = 38 from the index register. If the index register is not now zero, take the next command from the half-word specified in the address. This command (the TIX) must be in a left-half word itself, but it can transfer to either half word.

USE OF INDEX REGISTER

When the sign of a right-half command is negative, the indes register will be subtracted from the address of the command as the command pair is being copied into the command register. For example, if the index register contained a 5 and a right half command of -ADD 12460 were to be executed, the command would enter the command register as -ADD 12410.

In the TIX command, a one is subtracted from the index register. The TIX command does not transfer whenever the index is zero, either before or after the subtraction on the one. The test is done before the subtraction of the one and no transfer will occur when the first 11 bits are zeros. This occurs both when the index is 00010 or 00000.

An address with a half-word bit of one will be loaded into the index register as if its half-word bit were zero.

PROGRAMMING EXAMPLES

Note that commands can be operated upon just as though they were data. For example, if the command pair + CLA 12340 + SUB 54320 is in location 1000, the following coding would change it to + CLA 12340 + SUB 54330:

```
+ CLA 10000

+ ADD 20000 (where 2000 contains + 0000000 - 0000010)

+ STA 10001
```

However, this address modification is usually done by tagging the command with a minus sign. For example, suppose 64 numbers are stored from location 0123 through location 0222, and that these 64 numbers are to be totaled. One code to do this would be:

```
        + CLA 01230
        + ADD 01240
            .
            .
            .
            .
        + ADD 02220,
```

but this requires 32 words of coding and is tedious. (If only 3 numbers were to be added, this would be the way to do it.)

Another code is as follows:

```
COM     + CLA  SUM    (previously set to zero)
        + ADD  01230  (previously set to 01230)
        + STO  SUM
        + CLA  COM
        + ADD  1      Increase the address of the ADD command by 1
        + STO  COM
        + CLA  COUNT  (count previously set to 64)
        + SUB  1      Decrement count by 1 and test to see if finished
        + STO  COUNT
        + TNZ  COM
```

This saves space, but takes more time, since 64 ADD's are still executed plus all the other coding 64 times.

The same program, with the index register, is coded as follows:

```
        + LDI  COUNT  (COUNT equals 64. Any unused right
                       address can be used for the 64)
        + CAZ          Make A zero
COM     - ADD  02230
          TIX  COM
```

Note that this coding could just as well be used to add up 4000 numbers, or n numbers up to and including the contents of 0222.

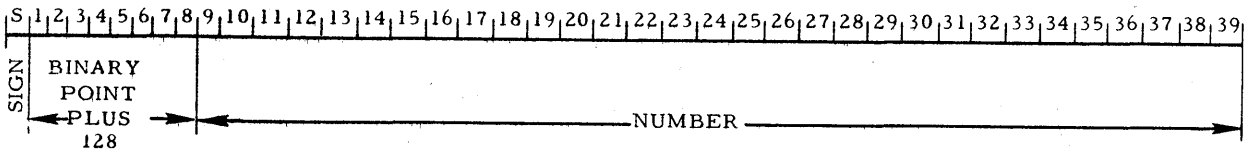
Since commands are picked up 2 at a time, it is impossible for a left-half command to modify the right-half command in the same word so that it can be executed in its modified condition. Thus, 1000 + STO 10000 + ANYTHING will change word 1000 but not the

command register. Similarly, 1000 ↓ STA 10001 ↓ ANYTHING and 1000 ↓ LDI COUNT - ANYTHING. The - ANYTHING is modified as the command pair enters the command register (before the index is loaded with the new count).

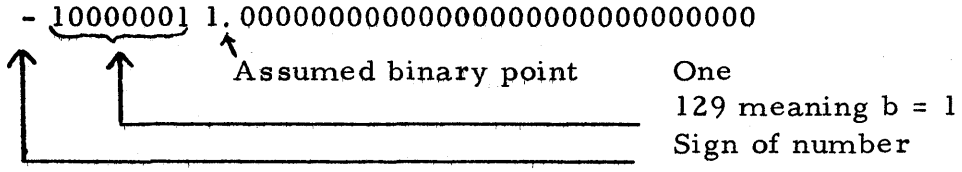
FLOATING POINT

Floating Point Format

A floating point word is a word with its binary point carried as part of the word. The programmer need not be concerned with scaling, overflow, etc., because the arithmetic operations automatically scale the numbers correctly. The format is as follows:



where the number (unless it is zero) is normalized; i. e., bit 9 is a one. The binary point is counted from bit 9 instead of bit one. The binary point plus 128 is used instead of a signed binary point (this is known as "excess 128"). Thus -1 would look as follows:



The maximum number would thus be 31 one's (approximately 1 times 2^{127}) or about 10^{38}. The smallest number would be 1/2 times 2^{-128} or about 10^{-38}. Floating point numbers are also commonly thought of as fractions times a power of 2 (the exponent).

Floating Point Commands (Optional)

Mnemonic	Octal	Description
FAD	75	Add arithmetically the floating point word addressed to A. R is destroyed. If the sum lies outside the permissible range, turn the overflow indicator ON.
FSB	74	Subtract arithmetically the floating point word addressed from A. R is destroyed. If the difference lies outside the permissible range, turn the overflow indicator ON.
FMP	23	Multiply A by the floating point word addressed. R is destroyed. If the product lies outside the permissible range, turn the overflow indicator ON.
FDV	26	Divide A by the floating point word addressed. R is destroyed. If the quotient lies outside the permissible range, turn the overflow indicator ON.

All floating point numbers must be normalized. If they are not, the computer assumes they are zeros. A floating point zero must consist of 39 zeros plus sign. (The computer will normalize floating point results automatically as well as insure that a zero result gets a zero exponent.)

To convert a floating point integer to a fixed point integer:

```
CLA  n
FAD  k
SUB  k
STO  n
```

To convert a fixed point integer to a floating point integer:

```
CLA  n          n = number to be converted
ADD  k          k = + 4774000-0000000
FSB  k
STO  n
```

V. PROGRAMMING SYSTEMS

RECOMP III INTERPRETIVE PROGRAM

There is a need for a program which will allow relatively untrained scientists and engineers to run their problems on an open shop basis.

There exists such a program, RIP, which allows the RECOMP III computer to be programmed as though it were a 9-index register, floating decimal point computer with built-in input/output and elementary functions (such as square root, sine-cosine, etc.). This program utilizes a 1-character operation code (such as + for addition, i for input, etc.) and a 4-digit decimal address, which can range from 0 to 3000, plus a 1-digit index register tag.

An example of the simple coding for a program which will input 10 numbers, add them up, and output their sum, is as follows:

Loc.	Op.	Addr.	Index	
0000	l	10	1	Load index number 1 with 10.
0001	i	20	1	Input numbers to locations 10 through 19.
0002	x	1	1	Transfer on index 1 to location 1.
0003	l	10	1	Load index 1 with 10.
0004	z			Clear pseudo-accumulator to zero.
0005	+	20	1	Add locations 10 through 19.
0006	x	5	1	Transfer on index 1 to location 5.
0007	o	a		Output accumulator (which contains sum).
0008	f	3	1	Output 1 carriage return (code 3).
0009	t	0		Transfer to location 0 (and call for 10 more inputs).

For further details, please refer to the write-up of R3P-16. For fixed point machines and R3P-39 for floating point machines.

RECOMP III COMPILER

The RECOMP III Compiler is a FORTRAN-like automatic coding system which accepts a source program written in a language closely resembling the ordinary language of mathematics, and produces, in one pass, a ready-to-be-run object program in RECOMP III machine language.

Although this compiler is designed to be run on a basic RECOMP III, it will efficiently utilize such additional peripheral equipment as the user may have.

The RECOMP III Compiler in effect transforms the RECOMP III into a machine with which communication can be made in a language more concise and more familiar than the RECOMP III language itself. The result is a considerable reduction in the training required to program, as well as in the time consumed in writing programs and eliminating their errors.

The object programs produced by this compiler minimize the access time to both data and instructions and are nearly as efficient as those written by experienced programmers.

APPENDIX I LIST OF COMMANDS

Mnemonic	Octal	Description
CLA	37	Clear and Add
CLS	36	Clear and Subtract
RCA	77	R Clear and Add
RCS	76	R Clear and Subtract
ADD	73	Add
SUB	72	Subtract
MPY	63	Multiply
DIV	66	Divide
ALS	02	A Left Shift
ASC	02.4	A Shift and Count
ASV	02.2	A Shift and Overflow
ARS	03	A Right Shift
XAR	56	Exchange A and R
EXT	70	Extract
TRA	51	Transfer
TPL	55	Transfer on Plus
TMI	53	Transfer on Minus
TOV	52	Transfer on Overflow
HTR	71	Halt and Transfer
TZE	50	Transfer on Zero
TNZ	11	Transfer on Non-Zero
TLB	15	Transfer on Low Bit
STO	45	Store
STR	05	Store R
STA	65	Store Address
LLS	42	Long Left Shift
LSC	42.4	Long Shift and Count
LSV	42.2	Long Shift and Overflow
CTL	40	Copy to L
CTV	41	Copy to V
CFL	44	Copy from L
ICH	00	Input Character
OCH	01	Output Character
LRS	43	Long Right Shift
NOP	54	No Operation
CAZ	16	Clear A to Zero
CMZ	16.4	Clear A to Minus Zero
CSA	13	Change Sign of A
OVN	13.4	Turn Overflow On
SAP	12	Set A Postive
SAN	12.4	Set A Negative
CAR	57	Copy A to R
CMP	17	Complement
RND	17.4	Round
CMG	35	Compare for Greater
CME	35.1	Compare for Equal

Mnemonic	Octal	Description
LDI	31.1	Load Index
SLC	25	Store Location Complement
STI	25.1	Store Index
TIX	10	Transfer on Index

Optional

FAD	75	Floating Add
FSB	74	Floating Subtract
FMP	23	Floating Multiply
FDV	26	Floating Divide

APPENDIX II

CHARACTER CODES

ØØØØØØ	Blank	1ØØØØØ	
ØØØØØ1	Upper Case	1ØØØØ1	
ØØØØ1Ø	Lower Case	1ØØØ1Ø	
ØØØØ11	Back Space	1ØØØ11	
ØØØ1ØØ		1ØØ1ØØ	
ØØØ1Ø1		1ØØ1Ø1	
ØØØ11Ø	a A	1ØØ11Ø	
ØØØ111	b B	1ØØ111	
ØØ1ØØØ	c C	1Ø1ØØØ	
ØØ1ØØ1	d D	1Ø1ØØ1	
ØØ1Ø1Ø	e E	1Ø1Ø1Ø	Stop
ØØ1Ø11	f F	1Ø1Ø11	Space
ØØ11ØØ	g G	1Ø11ØØ	C. Ret.
ØØ11Ø1	h H	1Ø11Ø1	Tab
ØØ111Ø	i I	1Ø111Ø	, >
ØØ1111	j J	1Ø1111	. <
Ø1ØØØØ	k K	11ØØØØ	Ø)
Ø1ØØØ1	l L	11ØØØ1	1 '
Ø1ØØ1Ø	m M	11ØØ1Ø	2 *
Ø1ØØ11	n N	11ØØ11	3 #
Ø1Ø1ØØ	o O	11Ø1ØØ	4 Δ
Ø1Ø1Ø1	p P	11Ø1Ø1	5 °
Ø1Ø11Ø	q Q	11Ø11Ø	6 π
Ø1Ø111	r R	11Ø111	7 _
Ø11ØØØ	s S	111ØØØ	8 :
Ø11ØØ1	t T	111ØØ1	9 (
Ø11Ø1Ø	u U	111Ø1Ø	- ≠
Ø11Ø11	v V	111Ø11	+ =
Ø111ØØ	w W	1111ØØ	/ √
Ø111Ø1	x X	1111Ø1	] [
Ø1111Ø	y Y	11111Ø	↑ ↓
Ø11111	z Z	111111	Code Delete

APPENDIX III

POWERS OF TWO

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125

APPENDIX IV

PROGRAM TITLE: RECOMP III, LOAD/START

1. PURPOSE

To provide means for starting a program, for loading of command format, alphanumeric format, or relocatable format program tapes, for output of command format, for alphanumeric information on tapes, and for basic debugging aids.

2. PROCEDURE

2.1 Loading procedure:

(1) The beginning of this program tape contains a bootstrap, self-loading, routine that requires a special loading procedure.

a. Place the tape in the reader, Flexowriter, or Facit, and turn the input switch on the console to the selected device.

b. Note: The operator must hold the reset button down while loading the bootstrap routine. While holding the reset button down, place the operation switch on continuous. When reading stops, take the operating switch out of continuous first, and release the reset switch.

(2) Now place the operation switch on continuous and the bootstrap routine will load the program.

2.2 Initial entry to the program is by a transfer to location 0000.0 or by depressing the "reset" button on the computer. (Before depressing the "reset" button, the operation switch should be in its off position.) After releasing the "reset" button, set the operation switch to the "continuous" position.

2.3 To perform a particular function, follow the instructions given in paragraph #3 for the desired function. The input and output control switches are presumed to be in the "automatic" position unless stated otherwise in the function description.

TITLE: LOAD/START

2.4 In all except the HALT, START, or ALPHA functions, return is made to step 2.3 for additional function selection.

3. FUNCTIONS

3.1 Alphanumeric Program Input:

Place any alphanumeric program tape (that was prepared by the Dump function of this program or any program that produces similar format) in the desired reading device and set the input control switch to the device chosen. After the tape is read, a check sum will be compared and if in agreement with the data just read, will continue reading until a stop code (visual E on tape) is read. If check sum did not agree, a slash (/) will be typed, but reading will continue. When all alphanumeric loading is completed, reset the input control switch to "automatic".

3.2 Manual Alphanumeric Input:

Type axxxx(CR) where xxxxx is the first location to be filled with alphanumeric data. Follow the carriage return by the desired alphanumeric characters. These characters will be loaded 5 to a word in sequential locations starting at the address initially entered in this function. As each computer word is filled, the "input" light on the typewriter will blink slightly. The last computer word should be padded (if necessary) with blanks. Exit from this mode with the "reset" button as described in 2.2. A rough check of the data entered can be made by setting the output control switch to the typewriter position (position 1) and performing an alphanumeric dump (3.7) of the area affected. The first and last 5 output characters will be meaningless.

3.3 Command Format Input:

Type cxxxxx(CR) where xxxxx is the first location to be filled with command format data. Follow the carriage return with the command format data of form ±xxxxxxx±xxxxxxx(CR) where x is an octal digit. The inside sign may be 1 or Ø.

TITLE: LOAD/START

3.3 (continued):

If at any time an error is detected by the operator, type a slash and re-enter that word. (See 3.18 for error correction.) The command format data is stored sequentially starting from the address given until another C function is received. After any carriage return, a new function may be initiated. If this data is on tape, load the tape and set the input selector switch to the input device chosen.

3.4 Repeat Previous Command Format Input:

By typing a decimal point (.), the last command format word entered will be entered again and the command format input location counter will be stepped to the next location. This function must be preceded by either functions 3.3, 3.4, or 3.17. This function is most useful when a large amount of identical command format data is to be entered sequentially.

3.5 Command Format Print:

Type pxxxxyyyy(CR) where xxxx is the first location to be typed and yyyy is the last location to be typed. Each word is followed by a carriage return. If this area is to be punched instead of typed, set the output selector switch to the desired device. After the output is completed, a new function may be performed. (See note 4.5.)

3.6 Register List:

Since the original contents of all the registers and loops are saved when entering the Load/Start routine and restored just before exit, the original contents of A, R, and Index Registers may be typed in command format in that order by typing the character r.

3.7 Memory Dump:

Type dxxxxyyyy(CR) where xxxx is the first location to be punched in alphanumeric format and yyyy is the last location to be punched. This information is preceded by an assignment

TITLE: LOAD/START

3.7 (continued):

word and followed by a checksum word. These memory dumps may be read back into memory by using function 3.1 or any similar program. If the information is to be dumped on any device other than the Flexowriter punch, set the output selector switch to the desired device.

3.8 Tape Feed:

By typing the character ↑, approximately five inches of blank tape will be punched. This function would be used before any memory dump and perhaps between sections of a tape containing more than one dumped area.

3.9 Tape Trailer:

By typing the character e, a stop code will be punched on tape followed by a visual E (in order to recognize the end of the tape) and followed again by approximately five inches of blank tape. This function would normally be used to end all memory dump tapes.

3.10 Verify Memory Dump:

Load a memory dump tape (one which was punched by the Memory Dump function (3.7) or similar program) on the Flexowriter tape reader and type the letter v. The data will be read (not into memory) and the sum of this data will be compared with the last word on tape. (No memory comparison is made.) If in agreement, a new function code will be called for. If not in agreement, a slash will be typed before asking for a new function code. If a memory dump tape contains more than one area, it will be necessary to type a v as each new area is reached. If a device other than the Flexowriter reader is to be used, set the input selector switch to the desired device.

3.11 Relocatable Input:

Load a relocatable program tape (prepared by program R3P-4 or a similar program) in the Flexowriter tape reader and type ixxxxx(CR) where xxxxx is the desired first word location of the program being read. After reading is completed, if the

TITLE: LOAD/START

3.11 (continued):

checksum is in agreement with the data read, a new function code will be called for. If not, a slash will be typed before a new function code is called for. If the tape is to be read by other than the Flexowriter tape reader, set the input selector switch to the desired device. NOTE: If the program being relocated contains loop instructions (40, 41, 44), the program should start with an address that is a multiple of 8 (last octal digit zero).

3.12 Start:

Type sxxxxx(CR) where xxxxx is the location to be transferred to. Before the transfer is made, the A, R, Index, and L and V loops are restored to the exact conditions they were when the Load/Start program was last entered. (See note 4.5 and 4.6.)

3.13 Halt and Transfer:

Type hxxxxx(CR) where xxxxx is the desired address for the HTR instruction at the end of this function. Before performing this HTR instruction, the A, R, Index, and L and V loops are restored to the exact conditions they were when the Load/Start program was last entered. (See note 4.5 and 4.6.)

3.14 Overflow Reverse:

Since while entering the Load/Start program, the overflow alarm was turned off, it may be desirable to turn the overflow on (or off) before performing a transfer function (see 3.12 or 3.13). To reverse the present state of the overflow alarm, type the letter o.

3.15 Quick Check of Load/Start Program:

As a check to see if the Load/Start program is in memory correctly, type the letter q. If correct, a new function code will be called for. If incorrect, a slash will be typed before calling for a new function code.

TITLE: LOAD/START

3.16 Stop:

Whenever a stop code (52) is detected while the Load/Start program is looking for a function code, a HTR 0002.0 instruction will be executed. Put the compute switch to its center position and back to compute in order to continue.

3.17 Ignore:

At any time in all functions, except alphanumeric inputs (3.1, 3.2, 3.10, 3.11), a delete code (77) is ignored. When the Load/Start program is looking for a function code, it also ignores blank (00), lower case (02), space (53), carriage return (54), and tab (55).

3.18 Error:

If at any time an error is detected by the operator while entering data for a function, the slash character (/) will cause another slash to be typed followed by a carriage return before calling for a new function code. If at any time, the Load/Start program detects an error, a slash and carriage return will also be typed before calling for a new function code.

4. NOTES

4.1 All functions that end with a carriage return could just as well be terminated with a space or tab.

4.2 Many errors on input will be detected, but not all.

4.3 Although this program occupies channels 0 and 1, channel 1 may be used for other purposes if only the following functions are needed: Alphanumeric Program Input, Command Format Input, Repeat Previous Command Format Input, Start, Halt and Transfer, Stop, Ignore, and Error.

4.4 During the input of a five digit address, an 8 digit limit, or all but the first sign position of command format inputs, the letter L is accepted as the number one (1).

TITLE: LOAD/START

4.5 The L and V loops may be listed if desired by the function p77607777(CR). Note, however, that the memory dump function (3.7) and relocatable input function (3.11) modify the V loop.

4.6 The L gray area is not modified; however, the V gray area will contain the original contents of the L loop when exit is made from this program.

4.7 None of the data inputs allow entry into the L loop. V loop entry is permissible.

5. LOCATION

This program is not relocatable and must occupy locations 0000 to 0177. (See note 4.3.) Load this program using the Bootstrap Loading Procedure as outlined in RECOMP III Technical Bulletin No. 1.

6. PROGRAM PREPARATION FORMAT

The following examples show several acceptable formats for typing a program the first time for entry through the Load/Start program.

6.1

```
c10000(CR)
+1234560-1615141(CR)
+5177600+3700340(CR)
-4220240+5200130(CR)
+1200000+5100130(CR)
+4000600+5101260(CR)
-0000000-0000000(CR)
...(CR)
+4500340+4200240(CR)
+4000050+5100731(CR)
+4077600+7200020(CR)
+3100171+0000010(CR)
+4300030+1500170(CR)
      (ETC)
```

TITLE: LOAD/START

6.2

```
c10000(CR)
+1234560-1615141(CR)
+5177600+3700340(CR)
-4220240+5200130(CR)
+1200000+5100130(CR)
+4000600+5101260(CR)
-0000000-0000000(CR)
..(CR)
(CR)
c10100(CR)
+4500340+4200240(CR)
+4000050+5100731(CR)
+4077600+7100200(CR)
      (ETC)
```

6.3

```
+1234560-1615141(sp)+5177600+3700340(sp)-4220240+5200130(CR)
+1200000+5100130(sp)+4000600+5101260(sp)-0000000-0000000(CR)
(sp)..(CR)
(CR)
c10100(CR)
.+4500340+4200240(sp)+4000050+5100731(sp)+4077600+7100200(CR)
(ETC)
```

7. MEMORY DUMP FORMAT

All data punched by the memory dump function and read by the verify or alphanumeric input function will be in the following format.

7.1 First word (5 characters) will be negative, contain a 4 at binary 7, contain the number of words in the memory dump at binary 19, and contain the last word location +1 of the data in the right address portion.

7.2 Following the first word is the data itself, punched 5 characters per word with the first data word also first on tape.

7.3 The last data word (5 characters) will be a checksum derived by the addition of all the data on tape (not counting the first assignment word) and ignoring overflow.

TITLE: LOAD/START

7.4 In most cases, the end of the tape will contain a stop code and a visual E following the checksum word.

8. RELOCATABLE INPUT FORMAT

All programs read by the relocatable input function must be in the following format:

8.1 The first non-blank character on tape is a code character, the next 5 characters form the first program word with all relative addresses in this word relative to zero.

8.2 The code character has 5 states. State 1 (1001000 that the word that follows has no relative addresses. State 2 (01110110) indicates that the word that follows has a relative right address only. State 3 (00100110) indicates that the word that follows has a relative left address only. State 4 (01001110) indicates that the word that follows has a relative left and right address.

8.3 Following the last data word is a code character that contains a low bit such as 00000001.

8.4 Following this low bit character is a checksum word (5 characters) which is derived by the addition of all the data words on tape (not counting code characters and all relative addresses relative to zero) and ignoring overflow.

8.5 In most cases, the end of the tape contains a stop code and a visual E following the checksum word.

TITLE: LOAD/START

9. FUNCTION SUMMARY

(1)	Alphanumeric Program Input	set input selector switch to desired tape reading device.
(2)	Manual Alphanumeric Input	axxxxx(CR)
(3)	Command Format Input	cxxxxx(CR) ±xxxxxxx±xxxxxxx(CR)--
(4)	Repeat Previous Command	type decimal point (.)
(5)	Command Format Print	pxxxxyyyy(CR)
(6)	Register List	type letter r
(7)	Memory Dump	dxxxxyyyy(CR)
(8)	Tape Feed	type character "f"
(9)	Tape Trailer	type letter e
(10)	Verify Memory Dump	type letter v
(11)	Relocatable Input	ixxxxx(CR)
(12)	Start	sxxxxx(CR)
(13)	Halt and Transfer	hxxxxx(CR)
(14)	Overflow Reverse	type letter o
(15)	Quick Check of Load/Start Program	type letter q
(16)	Stop	when stop code is read
(17)	Ignore	Always code delete (77), usually blank (00), lower case (02), space (53), carriage return (54), and t[illegible]55)
(18)	Error	if recognized by operator, type a slash (/). If computer recognized, a slash will be typed.