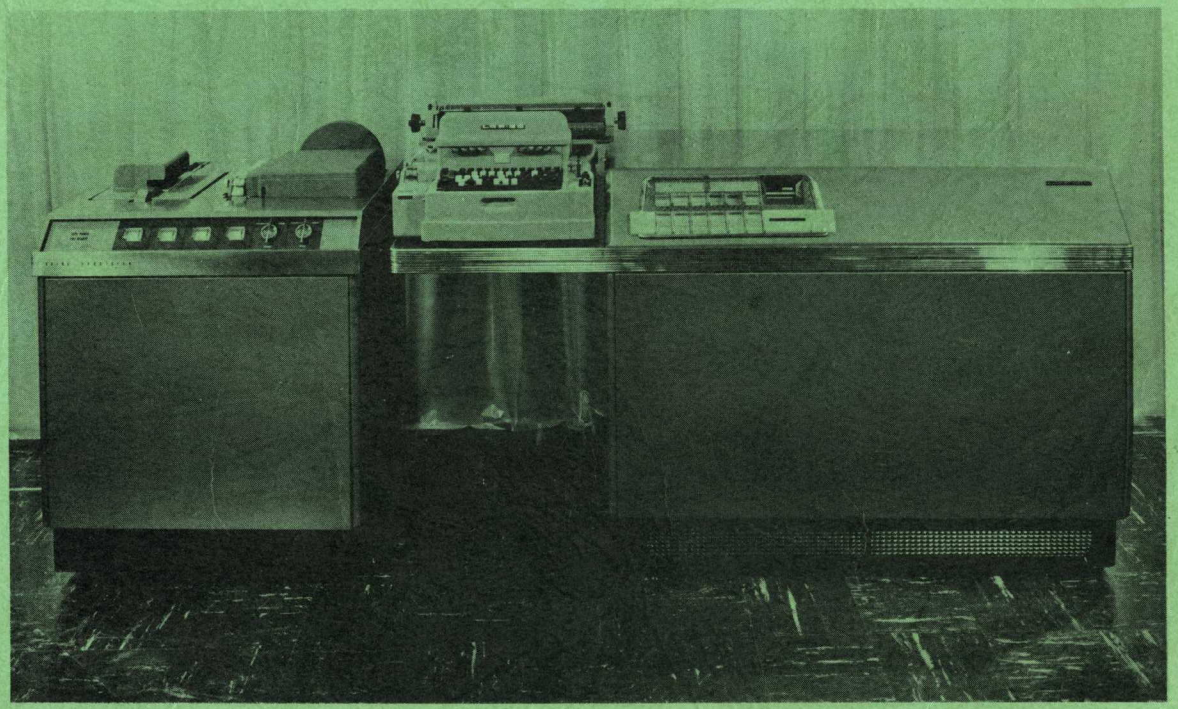
Royal Precision Electronic Computer

LGP-30

SUBROUTINE MANUAL



Royal McBee Corporation

The subroutines in this reference manual were compiled by the Royal McBee Electronic Computer Department to facilitate coding of problems for the LGP-30 Electronic Computer. The programming was done by the Royal McBee Electronic Computer Department and LGP-30 users.

These subroutines provide means for machine input, output, and program checkout, as well as evaluation of basic functions which are used in many problems. For each subroutine the calling sequence, running time, and storage requirements are given. Also, where applicable, the range of variables, scaling, and accuracy are given. The coding sheets for these subroutines are available in the LGP-30 SUBROUTINE MANUAL CODING SHEETS.

Only a small part of the programs available to LGP-30 users are contained in this manual. A complete list of LGP-30 programs is published periodically in the LGP-30 Newsletter. Programs may be obtained by writing to Royal McBee Corporation, Electronic Computer Division, 1532 North Cahuenga Blvd., Hollywood 28, California.

Royal McBee Corporation
Electronic Computer Department
Port Chester, New York

DATE: 10-5-60

PRINTED IN U.S.A.

INDEX OF ROUTINES & SUBROUTINES

DEFINITIONS

1. A routine is a logical subdivision of a program, complete in itself, and serving a specific function in the problem. There is no fixed length to any routine, and each routine occupies only as much storage as is actually needed.

2. A subroutine consists of a set of instructions to perform a standard task which is of a sufficiently general nature to be used in a number of different programs. Examples are subroutines to input and output data, compute square roots, arctangents, etc. This Subroutine Manual is a compilation of the specifications of the subroutines, completely describing the function and use of each.

3. A calling sequence is a set of instructions used for transferring from the main routine to a particular subroutine. It may also include information needed by the subroutine, such as constants and the locations of certain quantities. The calling sequence for each of the subroutines is given in the Subroutine Manual.

4. Minimum Time Programs

There are occasions when it is necessary to write programs which will be executed in as little time as possible. These minimum time programs are referred to as "optimum" programs. Since the subroutines contained in the Manual are to be used over and over again, they have been optimized. (The process of optimizing requires placing the sector of the operand α at $\alpha + (7k + 1)$ where $2 < k < 6$ for most instructions). The programmer should bear in mind that 10,000 executions of all nonoptimum instructions would take less than 3 minutes longer than 10,000 executions of optimum instructions. If the programmer spends 15 - 30 minutes on each routine trying to save machine time by optimizing, this time may never be made up in the actual running of the problem.

5. A scale factor of a scaled number in memory is defined as the power of 2 by which this scaled number must be multiplied to get the original or unscaled number.

CONVENTIONS USED IN THIS MANUAL

1. α is the base memory location from which entry to a subroutine is executed. Locations used by the subroutine are in reference to location α. E.G., $\alpha + 1$, $\alpha + 2$, $\alpha + 3$.....

2. L_o designates an initial location. L_f designates the final location.

3. The "Stop" and "Stop Codes" referred to in these write-ups and on the coding sheet are synonymous with "Conditional Stop Code".

4. For explanation of our scaling convention, see write-up on "SCALING".

5. Track 63 is used by some of these subroutines for temporary storage. The track 63 sectors used by the subroutines are enumerated in the respective write-ups. This practice was found useful for "optimum" programming of subroutines. However if the subroutines which use this temporary storage are to remain optimum, the L_o of the subroutine must be the beginning of a track. It is suggested that the programmer may also use track 63 for temporary storage of intermediate calculations. He should not place a number in a track 63 location used by one of these subroutines and expect that number to be there after exit from the subroutine.

6. All output subroutines should begin at sector 00. This will maintain the present timing relationship between the typewriter unit and the computer. If this relationship is not maintained the output routine may not function properly.

7. The term "Operation" (Op.) appearing on the LGP-30 coding sheets is synonymous with the term order.

8. All input times refer to mechanical reader.

PUNCHING TAPES FROM CODING SHEETS

See "Sample Program" page for example of coding sheet.

1. Only the "Program Input Codes" and "Instruction" columns of the coding sheets are to be punched, with appropriate stops. Never punch "Location", "Contents of Address", or "Notes" columns.

2. Each entry on a line must be followed by a conditional stop code -- "Stop" column, symbol (/). A line left blank must have the stop code punched.

3. Punch the "Program Input Codes" column only when there is an entry in the column. The "Program Input Codes" must be followed by the stop (/). This punching must precede the punching of the "Instruction" column on the same line of the coding sheet.

4. Leading zeros need not be punched. All other zeros must be punched. E. G., 00013086' only 13086' need be punched. ,0000017' must be punched ,0000017'. For T0059' punch T0059'.

5. Consider brackets as containing zeros. E.G., for [.....]'= [00000000]', only the stop code need be punched. For B[....]' = B[0000]' punch B0000'.

6. All punching may be done in lower case. B0627' will appear as b0627'

7. The placing of carriage returns is left to the discretion of the person preparing the tape. Carriage returns do not affect the input operation. We have arbitrarily placed a carriage return (X) after every 4 words on each coding sheet.

8. A heading may precede a punched program to identify the tape. Anything except a stop code may be punched as a header. Then as the tape is fed through the input reader the heading will print but will not affect the operation of the computer.

9. Each tape should be verified after punching. This can be done by placing the punched tape in the reader and "listing" the tape by the following process.

   a. Depress the COND. STOP Lever on the Typewriter.
   b. Depress START READ Lever on the Typewriter.
   c. When printing stops, depress the STOP READ Lever on the Typewriter.

   Then the printing may be visually checked against the coding sheets for correctness and presence of stop codes.

10. It should be the programmer's responsibility to enter "Program Input Codes" (and the associated stop codes) on the coding sheet. This will usually consist of a start fill (;), and set modifier (/), and possibly some hex. words (,) and/or stop and transfer (.) codes.

LGP-30 CODING SHEET

PREPARED FOR: | PAGE 1 OF 1

JOB NO.	PROGRAM NO.	PROGRAM PREPARED BY:	PROGRAM CHECKED BY:	DATE
XXX	YY.Y	Mel Kaye		Rev. 6/16/59

PROBLEM: EVALUATION OF 4th DEGREE POLYNOMIAL (Fixed Point) | TRACK 10

PROGRAM INPUT CODES	STOP	LOCATION	INSTRUCTION OPERATION	INSTRUCTION ADDRESS	STOP	CONTENTS OF ADDRESS	NOTES
;0000100	/						
/0000100	/	X					
		0000	x	R0158	/		Transfer to Data Input #1 to read convert, scale and
		0001	x	U0150	/		store data in memory
		0002	B	0023	/	A0027	Initial "Add" Inst.
		0003	C	0007	/	X	
		0004	H	0025	/		Set working storage to zero
		0005	B	0025	/		Working storage at q = 0
		0006	M	0026	/		X at q = 0
		0007	A	[ ]	/	X	A_n at q = 0
		0008	H	0125	/		working storage at q = 0
		0009	B	0107	/		A (0027 + N)
		0010	A	0122	/		1 at 29
		0011	Y	0107	/	X	A (0027 +N + 1)
		0012	S	0024	/	A0032	Flag
		0013	T	0015	/		
		0014	B	0125	/		Final result
		0015	x	R1412	/	X	Transfer to Data Output#1
		0016	x	U1400	/		To print final result
		0017	x	Z0000	/		Code word = 0 (for q = 0)
		0018	x	P1600	/		Carriage return
		0019	x	Z0000	/	X	
		0020	x	% 0800	/		Stop unless brk.pt. sw 8 dow
		0021	U	0000	/		Return to read more data
		0022	x	Z0001	/		1 at 29 } constants
		0023	A	0027	/	X	A(Lo) } And flags
		0024	A	0132	/		$A(L_f+1)$)
		0025	[ ]		/		Working storage at q = 0
		0026			/		X)
		0027			/	X	A_0 { These quantities
		0028			/		A_1 { are read from tapes converted to binary
		0029			/		A_2 { Scaled to q = 0 and stored at these
		0030			/		A_3 { addresses by data
		0031			/	X	A_4) Input #1 subroutine

FORM LP-10 Royal McBee Corporation DATA PROCESSING DIV. PORT CHESTER, NEW YORK

X = CARRIAGE RETURN

/ = CONDITIONAL STOP CODE

The LGP-30 normally handles all numbers as if they were of the form .XXXX.....that is, numbers numerically less than 1. However, it is quite simple to carry any number in the machine at any number of binary places, and this arithmetic is explained below. In talking about the placement of the radix point in the LGP-30, it is simpler to talk of the number of whole places in front of the radix point, rather than the number of places after the point. Hereafter, a number will be referred to as being carried at q places, q being the number of binary digits to the left of the radix point, and 30-q as the number to the right of the point.

Addition: Addition of course poses no problem if the two numbers to be added are at the same number of places. If not, either may be shifted before addition by multiplying or dividing by "One" at an appropriate q.

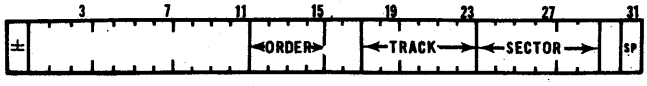
Multiplication: The LGP-30 multiplies a number at q_1 places by a number at q_2 places and forms the product in the accumulator at q_1 plus q_2 places.

Division: The LGP-30 divides the accumulator at q_1 places by a number at q_2 places and forms the quotient in the accumulator at $q_1 - q_2 = q_3$ places. It should be noted that overflow will occur if the quotient developed is not less than 2^{q_3} in absolute value.

Symbol	Command	Binary	Hex	Dec
Z	Stop	0000	0	0
B	Bring	0001	1	1
Y	Store Add.	0010	2	2
R	~~Return Add.~~	~~0011~~	3	3
I	Input	0100	4	4
D	Divide	0101	5	5
N	N Multiply	0110	6	6
M	Multiply	0111	7	7
P	Print	1000	8	8
E	Extract	1001	9	9
U	Transfer	1010	F	10
T	Test	1011	G	11
H	Hold	1100	J	12
C	Clear	1101	K	13
A	Add	1110	Q	14
S	Subtract	1111	W	15

Keyboard Code					
)0	02	02	Zz	01	01
Ll	06	06	Bb	05	05
*2	0f	10	Yy	09	09
"3	0q	14	Rr	0k	13
△4	12	18	Ii	11	17
%5	16	22	Dd	15	21
$6	1f	26	Nn	19	25
π7	1q	30	Mm	1k	29
∑8	22	34	Pp	21	33
(9	26	38	Ee	25	37
Ff	2f	42	Uu	29	41
Gg	2q	46	Tt	2k	45
Jj	32	50	Hh	31	49
Kk	36	54	Cc	35	53
Qq	3f	58	Aa	39	57
Ww	3q	62	Ss	3k	61
Sp.	03	03	LC	04	04
_	07	07	UC	08	08
=+	0g	11	CS	0j	12
:;	0w	15	CR	10	16
?/	13	19	BS	14	20
].	17	23	Tab	18	24
[,	1g	27	Del.	3w	63
Vv	1w	31		20	32
Oo	23	35			
Xx	27	39			

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



00	0
q4	57
j8	50
fj	43
90	36
74	29
58	22
3j	15
20	8
04	1
q8	58
jj	51
g0	44
94	37
78	30
5j	23
40	16
24	9
08	2
qj	59
k0	52
g4	45
98	38
7j	31
60	24
44	17
28	10
0j	3
w0	60
k4	53
g8	46
9j	39
80	32
64	25
48	18
2j	11
10	4
w4	61
k8	54
gj	47
f0	40
84	33
68	26
4j	19
30	12
14	5
w8	62
kj	55
j0	48
f4	41
88	34
6j	27
50	20
34	13
18	6
wj	63
q0	56
j4	49
f8	42
8j	35
70	28
54	21
38	14
1j	7

ROYAL MCBEE CORPORATION

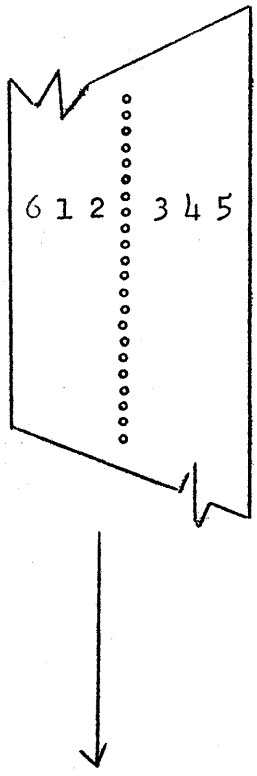
LGP-30 Input - Output

Keyboard Code

Numerical	123456	Commands	123456	Controls	123456
)0	000010	Zz	000001	Lower Case	000100
Ll	000110	Bb	000101	Upper Case	001000
*2	001010	Yy	001001	Color Shift	001100
"2	001110	Rr	001101	Carr. Return	010000
△4	010010	Ii	010001	Back Space	010100
%5	010110	Dd	010101	Tab	011000
$6	011010	Nn	011001	Condt. Stop (')	100000
π 7	011110	Mm	011101	Start Read	000000
Σ 8	100010	Pp	100001	Space	000011
(9	100110	Ee	100101	Delete	111111
Ff	101010	Uu	101001		
Gg	101110	Tt	101101	Signs	
Jj	110010	Hh	110001		
Kk	110110	Cc	110101	= +	001011
Qq	111010	Aa	111001	_ -	000111
Ww	111110	Ss	111101		

Balance of Keyboard

	123456
:;	001111
?/	010011
].	010111
[,	011011
Vv	011111
Oo	100011
Xx	100111



BOOTSTRAP ROUTINE
(Program 09.0)

FUNCTION:

To load the input routine on tracks 00, 01 and 02. After the bootstrap program has loaded the entire input routine, a halt is executed at track 63 sector 13 (3W34). Depressing the Start button transfers control to the first instruction of the input routine.

PROCEDURE:

The tape containing the bootstrap (and the program input routine) is placed in the tape reader and then the following manual operations are performed.

1. Connect Switch to "off" position.
2. Depress Typewriter START READ lever.
3. Depress MANUAL INPUT button on console.
4. Depress Typewriter START READ lever.
5. Depress FILL INSTRUCTION button on console.
6. Depress Typewriter START READ lever.
7. Depress ONE OPERATION button on console.
8. Depress EXECUTE INSTRUCTION button on console.
9. Repeat steps 3 through 8 five more times before proceeding to step 10.
10. Depress Typewriter START READ lever.
11. Depress NORMAL button on console.
12. Connect switch to "on" position.
13. Depress START button on console.

The entire tape will automatically read in after manually performing step 13 above.

OUTPUT:

Program input routine on tracks 00, 01, 02.

STORAGE:

The bootstrap routine uses 21 words on track 63. (sector 00 through 14, 22 thru 26 and 46).

TIME:

The time to read in the two programs after depressing the START button in Step 13 is approximately three minutes.

LGP-30 PROGRAM INPUT ROUTINE
(Program 10.4)

Original Program 10.3 Modified by Rice Institute

The Primary function of this routine is the entry of programs from paper tape or typewriter keyboard to storage on the LGP-30 memory drum. The general characteristics of the LGP-30 are described in the LGP-30 Programming Manual. A program consists of two types of words, instruction and data. This routine deals exclusively with the inputing of instruction words and hexadecimal representations of data words. This routine does not handle data words expressed in decimal form.

There are several functions to be performed by a useful program input routine.

1. The most direct way of entering instructions into the LGP-30 is to present it with binary words. But since it is difficult to program in this number system, we prefer to do our programming in decimal notation. If we are to write instructions in decimal form, we must provide the machine with a means of converting such instructions into binary form.

2. Most programs contain instructions which refer to locations within that program. Hence if we wish to place the program in another portion of memory, we must modify some of these instructional addresses.

3. It is sometimes convenient to express a number in binary form. e.g., π or other universal constants.

4. It may be necessary to make instructional or data changes to a program that has already been stored in memory.

These are the functions which this input routine is designed to perform.

This routine recognizes seven types of code words. After a code word has entered the accumulator (from tape or typewriter keyboard), the accumulator sign position and high order 3 binary bits are analyzed to identify the type of code word. Then a transfer to the appropriate interpretive subroutine within the program input routine is executed. These seven code words and their symbols are as follows.

1. Instruction (none) consists of an order and decimal address. The address consists of two decimal digits for track and two decimal digits for sector. The entire instruction is converted to its binary equivalent and stored in a given location. The address portion is incremented by the contents of the "Modifier" (to be discussed below) unless an "x" precedes the order. e.g., b4000 will be incremented: xb6310 will not be incremented. The x will not appear in the stored instruction.

2. Command (+). This code word will be treated as an instruction to the Program Input Routine. The instruction will be executed after the entry of another word. The command code word is input in decimal and is not incremented by the modifier. The second word, presumably data, is assumed to be in hexadecimal. e.g., the command +00h1637 followed by 73W08 will cause the hexadecimal word 00073W08 to be held (stored) in memory location track 16, sector 37.

3. Start fill (;). This code word tells the input routine where to begin filling input words (instructions and/or hexadecimal representations of data). Each succeeding word will be stored consecutively until or unless a new start fill interrupts the procedure. The address portion of the start fill code

word is decimal, and consists of both a track and a sector. e.g., the code word ;0003128 will cause the first input instruction or hexadecimal data word to be stored in track 31, sector 28, the second in location 3129, the third in 3130, etc.

4. Set modifier (/). The magnitude of the address of the set modifier code word is stored in a memory cell within the Program Input Routine known as the "modifier". This modifier is added to the address portion of all subsequently input instructions not preceded by an "X". The modifier remains unaltered until replaced by another set modifier code word. The set modifier code word will usually follow a start fill code and will usually be identical to it in magnitude. This word is for use by the Program Input Routine only. It will appear no where else on the drum as such.

5. Stop and transfer (.). This code word is executed in two parts. The first part causes the computer to execute a stop instruction. The second part, initiated by a depression of either the computer START button or the typewriter START COMP. lever, will cause the computer to transfer control to the memory location contained in the stop and transfer code word. The "stop" part of the stop and transfer code is ignored if BREAK POINT 32 on the computer console is in the "down" position, e.g. the code word .0001700 will stop reading. Then, upon depression of the START button, control will be transferred to memory location 1700.

6. Hex. words (,). This code word causes the next N_1 N_2 words to be filled without decimal translation. N_1 N_2 is specified in the sector portion of the "Hex. words" code word. $1 \leq N_1 N_2 \leq 63$ The code word ,0000014 means that the next 14 input words are to be stored in the next 14 locations of memory. The input words must be in hexadecimal and they will not be incremented by the modifier.

7. Hex. fill (v). This code word causes the next n hexadecimal words to fill consecutively beginning in location m The format of the Hex fill code word is vnl n2 n3 ml m2 m3 m4. During the filling procedure a summation of the binary bits actually stored into memory (a check sum) is generated. After all (n) words have been stored on the drum, and if the TRANSFER CONTROL button is not depressed, this check sum is compared against a previously computed check sum * placed at the end of the m hexadecimal words. If the two check sums are identical the hex. fill procedure is successfully completed. If the two check sums are not identical the Program Input Routine will return the typewriter carriage and print "error". The code word vlj02w00 will cause the next $1j0_{16} = (448)_{10}$ words consecutively beginning in location $2W00 = m[(2W00)_{16} =$ location $(4710)_{10}]$. As many as $(7WW)_{16} = (2047)_{10}$ words can be filled by a single Hex. fill code word.

Leading zeros need not be punched on any input word. All other zeros must be punched. e.g. The code word /0001357 must be completely punched; for 000B3749 only the last 5 characters need be punched. e.g. B3749.

*Either program 13.1 or 13.2 will prepare the previously computed check sum and place it at the end of the Hex fill input words.

LGP-30 PROGRAM INPUT ROUTINE

When the overall coding for a problem is surveyed, it is found that the instructions separate logically into independent groups, some of which can be used in any number of problems. Examples of these groups are subroutines of all types, standard input and output routines, and the mathematical subdivisions of the problem. It would be desireable to code these pieces without reference to the other pieces. In order to separate these pieces completely, it is necessary to assign a group of instructions a block of storage locations which does not correspond to actual memory locations; otherwise two blocks of coding might be found to occupy the same section of storage, requiring a change in the coding for one of the two pieces. The "Set Modifier" code word was intended to facilitate this type of coding. A group of instructions can be coded without reference to actual memory locations by starting that group at relative address 0000. Then by "setting the modifier" to the "start fill" locationthe programmer may position a routine to any part of memory. An instruction preceded by an "x" will not be incremented, and this instruction will still refer to an absolute memory location. It should be noted that instructions may be coded for actual locations merely by setting the modifier to zero (i.e., input code word /0000000). Thus no particular restrictions are imposed upon the programmer by this system.

If the Program Input Routine detects an erroneous code word it will return the typewriter carriage, print "error", and stop. The last word read from tape (or keyboard) contains the erroneous code.

A tape prepared for this Program Input Routine must contain typewriter format control. It is suggested that a carriage return be placed after every four or six words on tape. If there is no format control and the typewriter carriage is permitted to space into the automatic carriage return tab a stop may result. The computer will continue unaffected if the carriage return key is depressed.

TIME:

Instructions are loaded and converted to binary at the rate of one track every 60 - 70 seconds. The Hex. fill code loads and computes the check sum at the rate of one track every 50 - 60 seconds.

STORAGE:

Locations 0000 through 0263. (3 tracks). No temporary storage.

PROGRAM STOPS:

Location	Meaning
0062	Erroneous input code word.
0062	The computed check sum is not identical with the check sum on tape.

PROGRAM INPUT ROUTINE CODE WORDS
(Program 10.4)

Code	Order	Track	Sector	INTERPRETED AS:
	Ø	T_1 T_2	S_1 S_2	Instruction - modified
	X Ø	T_1 T_2	S_1 S_2	Instruction - Not modified
8 0 0	Ø	T_1 T_2	S_1 S_2	Negative instruction - modified
8 0 X	Ø	T_1 T_2	S_1 S_2	Negative instruction - Not modified
+ 0 0	Ø	T_1 T_2	S_1 S_2	Command. This word is treated as an instruction to the input routine, using the following word as data. Following word is in hex.
; 0 0 0	T_1	T_2	S_1 S_2	Start fill - The first input word will be stored in location T_1 T_2 S_1 S_2, and succeeding words will be stored sequentially.
/ 0 0 0	T_1	T_2	S_1 S_2	Set modifier - Set "modifier" location to T_1 T_2 S_1 S_2. Add modifier to all succeeding instructions not preceded by an "X".
. 0 0 0	T_1	T_2	S_1 S_2	Stop and transfer - Stop (unless BREAK POINT 32 is down) and transfer to T_1 T_2 S_1 S_2 upon depression of START button or START COMP. lever.
, 0 0 0 0 0	N_1	N_2		Hexadecimal words. The next N_1 N_2 (dec.) words are hex to be filled sequentially. $1 \leq N_1 N_2 \leq 63$.
v n_1 n_2 n_3	m_1	m_2	m_3 m_4	Hexadecimal fill - Fill n_1 n_2 n_3 (hex) words hexadecimally beginning in location m_1 m_2 m_3 m_4 (hex). $1 \leq n_1$ n_2 $n_3 < (2047)$. Compute a check sum for these filled locations and verify this computed check sum against a previously computed check sum placed on the tape after the last word on tape. Do not read and verify the check sum if the TRANSFER CONTROL button is depressed.

Ø represents any one of the 16 LGP-30 orders (or commands) T_1 T_2 are the decimal digits for the track part of an address. S_1 S_2 are the decimal digits for the sector part of an address.

DATA INPUT NO. 1 SUBROUTINE
(Program 11.0_R)

FUNCTION:

To read a decimal number from tape, convert to binary, scale to the proper binal point location, and store the word in a specified drum location. For each number the following is punched on tape.

1. The decimal point location of the number on tape, counting from right to left. (One decimal digit designated as "P").
2. The binal point location desired for the number to be placed on drum. (Sign and two decimal digits designated as "q").
3. The drum location to which the number is to be sent. (2 decimal digits for track and 2 decimal digits for sector).
4. The number to be entered (Seven decimal digits plus sign).

INPUT:

For each word to be stored an identification word (parts 1, 2, & 3 above) and the signed number (part 4 above) are required.

CALLING SEQUENCE:

Location	Order	Address
α	R	$(L_0 + 8)_{10}$
$\alpha + 1$	U	L_0
$\alpha + 2$	etc.	

OUTPUT:

The scaled binary representation of the number will appear at its proper drum location.

EXIT:

A "zero"identification word will cause the routine to exit to $(\alpha + 2)$.

SCALING:

The location of the decimal point in the decimal number is specified by a number P, which denotes the number of places following the point in the seven digit field. P can be in the range $0 \leq P \leq 9$. The location of the binary point in the full 30 bit binary word is specified by q, the number of digits preceding the point in the full word. The q can lie in the range given in the following table.

In order for the number to be representable at a given q, the number must be less (in absolute value) than 2^q However, if too large a q is used, the number will not reconvert exactly on output, since there will be too few binary digits following the point to adequately represent the fractional part of the number. The following table also gives the maximum conversion correspondence between P and q.

TABLE OF P vs q:

P	Max q	Min q	Max q for Exact Reconversion	Min q for Max No. Size (all 9's)
0	+47	+02	+30	+24
1	+43	-02	+26	+20
2	+40	-05	+23	+17
3	+37	-08	+20	+14
4	+34	-11	+16	+10
5	+31	-14	+13	+07
6	+28	-17	+10	+04
7	+24	-21	+06	00
8	+21	-24	+03	-03
9	+17	-28	00	-06

R0308
U0300
B6200
R0M12
U0700
Z0000

TIME:

20 - 25 words per minute.

ACCURACY:

The scaled number in memory may be inaccurate in the 30th binary position.

STORAGE:

192 locations of instructions and constants. Eight locations of temporary storage (Track 63, sectors 03, 04, 45, 52, 54, 55, 56, 57.).

PROGRAM STOPS:

Location	Meaning
$(L_o + 0234)_{10}$	Divide check in scaling data word $\|N\| > 2^q$

EXAMPLES: (See LGP-30 Data Input 1 load sheet)

1. Place +96.40236 in drum location 6234 at a q of +7.
2. Place -.000000597 in drum location 2363 at a q of -14.
3. Place +330000. in drum location 2100 at a q of +30.

TAPE PUNCHING INSTRUCTIONS:

1. All characters of the I. D. word must be punched. E.g. punch 0+096300' completely. The first three characters should not be omitted. The stop code (') must be the last character.
2. Leading zeros of a positive number need not be punched. To enter all zeros merely punch a stop code. All digits of a negative number must be punched.
3. Be sure to check each load sheet to see whether an additional stop code should follow the last number punched.

2+246200'

LGP-30 DATA LOAD SHEET

PREPARED FOR:

PAGE 4 OF 4

JOB NO.

PROGRAM NO. 11.0R

PROGRAM PREPARED BY:

PROGRAM CHECKED BY:

DATE

PROBLEM: EXAMPLES FOR DATA INPUT # 1

DATA INPUT NO.

NOTES	p	± q	LOCATION	STOP	± NUMBER	STOP	CGE RET
	5	+07	6234	'	96402316	'	
	9	-14	2363	'	-00000597	'	X
	0	+30	2100	'	330000	'	

$0 \leq p \leq 9$
$-28 \leq q \leq 47$
$0000 \leq Loc \leq 6363$

Royal McBee Corporation
DATA PROCESSING DIV.
PORT CHESTER, NEW YORK

FORM LP-16

PUNCH A STOP CODE AFTER THE LAST NUMBER? YES [X] NO [ ]

DATA INPUT NO. 2 SUBROUTINE
(Program 11.1)
(T.Kampe - Librascope Incorporated)

FUNCTIONS:

1. To input a sequence of "N" signed seven decimal digit numbers, each with the same decimal point, convert to binary (all at the same q) and store sequentially in M, M + 1 .....

OR

2. To input one signed seven decimal digit integer and convert to a signed binary integer at q = 30.

I. SEQUENTIAL FILL:

Input:

"N" signed decimal numbers on tape and the hexadecimal code word in the accumulator.

CALLING SEQUENCE:

LOCATION	ORDER	ADDRESS
α - 1	B	L (Code word)
α	R	$(L_o + 0121)_{10}$ Track 01, Sector 21.
α + 1	U	$(L_o + 0104)_{10}$ Track 01, Sector 04.

α - 1 need not contain a B order. Any order or sequence of orders that leaves the code word in the accumulator is permissible.

The code word must be in the form: $n_1\ n_2\ n_3\ C\ m_1\ m_2\ m_3\ m_4$. N = $(n_1\ n_2\ n_3)$ = number of words to be filled (in hexadecimal at q = 11) $0<N<2048$.
C = characteristic of numbers to be filled (number of integers) $0<C<9$. See correspondence of C and q under "Output".
M = $(m_1\ m_2\ m_3\ m_4)$ = First address to be filled (in hexadecimal at q = 29).

Examples:

1. Code word 00F30200 means fill 10 (F) words, starting in 0200 with numbers of the form ± XXX.XXXX (stored at q = 10)

2. Code word 0318051J.
N = $(31)_{16} = (49)_{10}$

C = 8

M = $(051J)_{16} = (0507)_{10}$

Fill 0507, 0508, ..., 0555 with the next 49 decimal words on tape. Words are interpreted as ± XXXXXXX0 (stored at q = 27).

Output:

Binary representation of words on tape filled sequentially beginning in location M.

C	q	Decimal words interpreted as:
0	0	± .XXXXXXX
1	4	± X.XXXXXX
2	7	± XX.XXXXX
3	10	± XXX.XXXX
4	14	± XXXX.XXX
5	17	± XXXXX.XX
6	20	± XXXXXX.X
7	24	± XXXXXXX.
8	27	± XXXXXXX0.
9	30	± XXXXXX00.

Exit:

After all words have been scaled and stored, the routine exits to α + 2.

Accuracy:

± 1 at q = 30 for $0 \leq C \leq 6$. Exact conversion for $7 \leq C \leq 9$.

Time:

45 to 55 words per minute.

Note: If the hexadecimal code word is on tape, replace the α-1 instruction of calling sequence by the instructions P0000 and I0000.

II ONE WORD INTEGER CONVERSION:

Input:

One signed decimal integer on tape in the form ± XXXXXXX.

Calling Sequence:

Location	Order	Address
α	R	$(L_o + 7)_{10}$
α + 1	U	L_o
α + 2	etc.	

Output:

Binary integer at q = 30 in accumulator.

Accuracy:

Conversion is exact.

PROGRAM STOP:

Location	Meaning
$(L_o + 0117)_{10}$	Divide check in scaling data word. $\|N\| \geq 2q$

STORAGE: 89 locations of instructions and constants. No temporary storage.

NOTE: When using the one word integer conversion entry (i.e.U L_o) the accumulator is not cleared before executing the P0000 and I0000 instructions.

DATA INPUT NO. 3 SUBROUTINE

(Program 11.2)

FUNCTION:

To input groups of decimal numbers from tape, each group with the same decimal point, convert each number to binary, all at the same q, and store in consecutive memory locations. This differs from Input No. 1 in that each group of numbers, rather than every number, is preceded by an identification word. The identification word contains P, ± q, and the location for the first number to be stored. All following numbers of the group are filled sequentially at the same P and q. A "minus zero" number will terminate the group, and another identification word will be read (See examples for "minus zero" format).

CALLING SEQUENCE: Same as Input No. 1

Location	Order	Address
α	R	$(Lo + 8)_{10}$
α + 1	U	Lo)
α + 2	etc.	

EXIT:

A zero identification word (normally preceded by a minus zero number) will cause the routine to exit to α + 2.

TIME

45 - 55 words per minute.

STORAGE:

192 locations of instructions and constants. (3 tracks) Five locations of temporary storage. (Track 63, sector 00, 01, 02, 03, 04)

PROGRAM STOP:

Location	Meaning
$(Lo + 0135)_{10}$	Divide check in scaling data word $\lvert N \rvert \geq 2^q$

EXAMPLES: (See LGP-30 Data Input 3 Load Sheet)

Group No. 1. Place +96.40236 in drum location 6234 at q = 7
" -30.00000 " " " 6235 " "
" +03.14159 " " " 6236 " "
" -21.50000 " " " 6237 " "

The Minus zero causes identification word No. 2 to be read.

Group No. 2 Place -.000000597 in drum location 2363 at q = -14
" +.000009000 " " " 2400 " "
" +.000060000 " " " 2401 " "

The minus zero causes identification word No. 3 to be read.

LGP-30 DATA LOAD SHEET

PREPARED FOR: | PAGE 3 OF 3

JOB NO. | PROGRAM NO. 11.2 | PROGRAM PREPARED BY: | PROGRAM CHECKED BY: | DATE 6/18/59 Rev.

PROBLEM: EXAMPLES FOR DATA INPUT # 3 | DATA INPUT NO.

NOTES	P	± q	LOCATION	STOP	± NUMBER	STOP	CODE RET
	5	+ 07	6234	/	9641023600	/	
				/	-3000000000	/	X
				/	3141591	/	
				/	-2150000	/	X
				/	-0000000	/	
	9	-14	2363	/	-0000597	/	X
				/	9000	/	
				/	6000000	/	X
				/	-0000000	/	
	0	+30	2100	/	3300000	/	X
				/	-0000000	/	
				/		/	X
				/		/	
				/		/	X
				/		/	
				/		/	X
				/		/	
				/		/	X

Group No. 3. Place +330000 in drum location 2100 at q = 30.

The minus zero word causes identification word No. 4 to be read. Identification word No. 4 is the extra stop code punched after the last number (see bottom of load sheet, "Yes" is checked.)

This enters a zero identification word, so the subroutine exits to (α+ 2).

See Data Input 1 write-up for "Scaling", "Table of P vs q", and "Tape Punching Instructions".

				/		/	
				/		/	X
				/		/	
				/		/	X

$0 \leq p \leq 9$
$-28 \leq q \leq 47$
$0000 \leq Loc \leq 6363$

Royal McBee Corporation
DATA PROCESSING DIV.
PORT CHESTER, NEW YORK

FORM LP-16

PUNCH A STOP CODE AFTER THE LAST NUMBER?
YES [X] NO [ ]

DATA OUTPUT NO. 1 SUBROUTINE
(Program 12.0_A)

FUNCTION:

To convert and print a nine decimal digit number plus decimal point and sign (sign if the number is negative).

INPUT:

The number to be printed in the accumulator and a code number in storage location $\alpha + 2$ to indicate the number of integers before the decimal point.

CALLING SEQUENCE:

Location	Order	Address
$\alpha - 1$	B	L (N)
α	R	$(L_o + 12)_{10}$
$\alpha + 1$	U	L_o
$\alpha + 2$	Z	000C
$\alpha + 3$	etc.	

$\alpha - 1$ need not contain a bring order. Any order which leaves the argument in the accumulator is permissable.

C denotes code number and may be 0 thru 9.

OUTPUT:

Nine decimal digits plus a sign (or space if the number is positive) and a decimal point. Leading integral zeros are printed as spaces.

CODE NUMBER:

C	q of No.	Output
0	0	.XXXXXXXX
1	4	X.XXXXXXX
2	7	XX.XXXXXX
3	10	XXX.XXXXX
4	14	XXXX.XXXX
5	17	XXXXX.XXX
6	20	XXXXXX.XX
7	24	XXXXXXX.X
8	27	XXXXXXXX.
9	30	XXXXXXXX.

MISCELLANEOUS:

Each binary number is scaled and converted as a fraction. The code number is used by the routine as a print stroke counter. The only format control is a tab after printing. It is safe to print immediately on exit from the routine. After printing control is returned to $\alpha + 3$.

TIME:

Printing takes about 1.5 seconds including the tab.

ACCURACY:

Maximum error is one in the ninth printed digit.

STORAGE:

121 locations of instructions and constants.

PROGRAM STOP:

Location	Meaning
$(L_o + 38)_{10}$	Argument $\geq 10^C$

DATA OUTPUT NO. 5 SUBROUTINE
(Program 12.4)

FUNCTION:

To print one or more groups of numbers in decimal form where each group may consist of one or more numbers stored in consecutive memory locations. All numbers in each group are assigned a specified binal point location (q) in the calling sequence. Spacing operations are executed in place of leading integral zeros.

INPUT:

1. One or more groups of numbers stored in memory.
2. A calling sequence consisting of pairs of instructions of the following kind:
   a. The initial location of the group.
   b. The number of numbers (N) and the binal point location (q) of the group.

CALLING SEQUENCE:

Loc.	Inst.	Add.
α	R	Lo + 3
α + 1	U	Lo
α + 2	Z	$Loc._1$
α + 3	Z	N_1 q_1
α + 4	Z	$Loc._2$
α + 5	Z	N_2 q_2
.	.	.
.	.	.
.	.	.
[α + 2 i]	Z	$Loc._i$
[(α + 2 i) + 1]	Z	N_i q_i
[(α + 2 i) + 2] etc.		

$Loc._i$ = the location of the first number of group i.

N_i = number of numbers in group i. N_i is placed in the track position (in decimal). $1 \leq N_i \leq 63$. q_i = binal point location of the i'th group. q_i is placed in the sector position (in decimal). $0 \leq q_i \leq 31$

$Loc.\alpha + 2_i + 2$ must not contain a Z order.

OUTPUT:

Each output number will consist of a decimal point and eight (or more) decimal digits. Each number is followed by the sign if the number is negative. A tab is executed after each number.

Table of q versus Output:

q	Output
0	.XXXXXXX+
0 - 4	X.XXXXXX+
4 - 7*	XX.XXXXX+
*7 - 10	XXX.XXXX+
10 - 14	XXXX.XXX+
14 - 17	XXXXX.XX+
17 - 20	XXXXXX.X+
20 - 24	XXXXXXX.+
24 - 27	XXXXXXXX.+
27 - 30	XXXXXXXXX.+
30 - 31	XXXXXXXXX.+

*See EXAMPLE for the situation requiring the recurrance of upper limit in successive classes of q.

EXIT:

As was mentioned above under INPUT, a pair of "Z" instructions are required for defining the group of numbers to be printed. When the last such group has been printed, the routine will exit to the next instruction of the calling sequence, and this location will contain a "non- Z" instruction.

EXAMPLE:

Loc.	Inst.	Add.		Notes
α	xR	Lo + 3		
α + 1	xU	Lo		
α + 2	xZ	2100	(1)	$Loc._1$ = 2100
α + 3	xZ	0407		N_1 = 4; q_1 = 7
α + 4	xZ	2110	(2)	$Loc._2$ = 2110
α + 5	xZ	1115		N_2 = 11; q_2 = 15
α + 6	xB	β	(3)	

The above calling sequence will cause this subroutine to:

1. Print the contents of locations 2100, 01, 02, and 03 as XX.XXXXXX+ for those numbers numerically smaller than 100.00000 or as XXX.XXXXX+ for those numbers which exceed this number. (See q of 7 under Table of q vs. Output).
2. Print the contents of 2110 thru 2120 as XXXXX.XXX+
3. Exit to α + 6 which is the "non- Z" instruction terminating the calling sequence.

PROGRAM STOPS:

Loc.	Meaning and Remedy
Lo + 0304	N < 1. Depress the start to exit without printing.

ACCURACY:

Output is exact (and rounded) for eight printed digits. When more digits are printed, the ninth printed digit may be high by one or two.

STORAGE:

224 locations of instructions and constants (3 1/2 tracks). Five locations of temporary storage (track 63, sectors 41, 42, 43, 46, 48).

TIME:

30 to 35 words per minute.

NOTES

HEXADECIMAL PUNCH
(Program 13.2)

FUNCTION:

To punch the contents of consecutive memory locations and to compute and punch a check sum. The output of this routine may be punched on tape through the typewriter or through the twenty character per-second unit.

INPUT:

Beginning and final locations (L_o and L_f) in decimal.

OUTPUT:

The output of this routine is in the form required by program 10.4

1. An identification word will be the first word punched on tape. This identification word consists of v N M, where N is the number of words in the record and M is the initial location (L_o) of the record. Both N and M are in hexadecimal. $(001)_{16} \leq N \leq (7ww)_{16}$.

   (a) Example:
   The identification word v08$\frac{1}{2}$218J' denotes a record of 132 words beginning in location 3335.

2. After the identification word has been punched the contents of memory locations L_o through L_f will be punched.* Following every sixth word the routine will punch a carriage return.

3. When the entire record has been punched the routine will compute the check sum, punch a carriage return, and then punch the check sum. The check sum will always be an eight digit word that might contain leading zeros.

PROCEDURE:

A: When the output is through the medium of the typewriter.

1. Depress MANUAL INPUT lever on the typewriter.
2. Transfer to the first location of this routine.
3. After the "manual input" light turns on, type the beginning and final locations in decimal. This will be one eight digit word (L_o and L_f).
4. Put break point switch 32 in the UP position.
5. Depress the PUNCH ON lever on the typewriter.
6. Depress the START COMP. lever on the typewriter.

NOTE: Following step six this program begins punching. After the check sum has been punched control is returned to step 3 of the procedure where a new L_o and L_f may be entered.

* Leading zeros are not punched. For the contents of a memory location which contains a zero only the conditional stop code will be punched.

B. When the output is through the medium of the twenty character per-second punch unit.

1. Make sure the input selector switch is turned to TYPEWRITER.
2. Depress MANUAL INPUT lever on the typewriter.
3. Transfer to the first location of this routine.
4. After the "Manual Input" light turns on, type the beginning and final locations in decimal. This will be one eight digit word (L_o and L_f).
5. Put break point switch 32 in the DOWN position.
6. Turn the output selector switch to PUNCH position.
7. Depress the START COMP. lever on the typewriter.

NOTE: Following step seven this program begins punching. After the check sum has been punched control is returned to step 4 of the procedure where a new L_o and L_f may be entered.

TIME:

A. Approximately 64 words per-minute using the typewriter.

B. Approximately 128 words per-minute using the twenty character per second punch unit.

STORAGE:

282 locations of instructions and constants. Eleven locations of temporary on track 63 (sectors 14, 17, 22, 31, 32,35, 38, 43, 48, 53, 60).

SINE-COSINE SUBROUTINE
(Program 14.0)

FUNCTION:

To compute the sine or cosine of any given angle. A 9th degree polynomial approximation is used. The argument must be in degrees and will be reduced to the first quadrant equivalent before computation of the function.

INPUT:

One word in the accumulator at q = 9.

OUTPUT:

One word in the accumulator at q = 1.

CALLING SEQUENCE:

	SINE			COSINE	
Location	Order	Address	Location	Order	Address
α - 1	B	L(Arg.)	α - 1	B	L(Arg.)
α	R	$(Lo + 49)_{10}$	α	R	$(Lo + 49)_{10}$
α + 1	U	Lo	α + 1	U	$(Lo + 4)_{10}$
α + 2	etc.		α + 2	etc.	

α - 1 need not be a B order. Any order or orders that leaves the argument in the accumulator is permissable.

ACCURACY:

The maximum error is approximately 5×10^{-7}.

TIME:

250 to 275 MS.

STORAGE:

64 locations of instructions and constants. 6 locations of temporary storage (Track 63, sectors 02, 04, 05, 06, 07, 45).

SQUARE ROOT SUBROUTINE
(Program 15.1)

(M. Levy - White Sands Proving Ground, N.Mexico)

FUNCTION:

To compute the square root of any positive number. The argument may be at any even scale q, and the result is at q/2.

INPUT:

One word in the accumulator at any even q.

OUTPUT:

One word in the accumulator at q/2.

CALLING SEQUENCE:

Location	Instruction
α	$R(Lo + 50)_{10}$
α + 1	U Lo
etc.	

METHOD:

Newton's method to solve the equation

$$0 = x^2 - a$$

by the successive approximations

$$x_{i+1} = x_i + [-1/2][-a/x_i + x_i].$$

ACCURACY:

If x is the true square root and x* is the computed root then

$$x - x^* < 2^{-30}$$

Note that if the true square root is digital, then the computed root is exact.

TIME:

Maximum is 510 ms.

STORAGE:

51 locations. No temporary storage.

PROGRAM STOPS:

$(Lo + 24)_{10}$ Argument is negative; restarting sets the accumulator to zero and exits.

ARCTANGENT SUBROUTINE
(Program 16.0)

FUNCTION:

To compute the arctangent of any given number. A 15th degree polynomial approximation is used. The output is in degrees, and the principle value will be given (first or fourth quadrant).

INPUT:

One word in the accumulator at q = 9.

OUTPUT:

One word in the accumulator at q = 9 (degrees).

CALLING SEQUENCE:

Location	Order	Address
$\alpha - 1$	B	L (Arg.)
α	R	$(L_o + 51)_{10}$
$\alpha + 1$	U	L_o
$\alpha + 2$	etc.	

$\alpha - 1$ need not be a B order. Any order or orders that leaves the argument in the accumulator is permissable.

ACCURACY:

Maximum error is 5×10^{-7} degrees. The output will be between 0°and 89,90; because the argument cannot be numerically greater than 512. If the programmer wants his output to come closer to 90° he can modify the routine by changing $(L_o + 56)_{10}$ and $L_o + 59)_{10}$ from 1 and 2, respectively, at q= 9, to 1 and 2 at some greater q_2 Then the argument must be at q_2

TIME:

320 milliseconds.

STORAGE:

64 locations of instructions and constants. 10 locations of temporary storage (track 63, sectors 04, 05, 06, 07, 08, 09, 10, 13, 50, 51).

EXPONENTIAL SUBROUTINE
(Program 17.0)

FUNCTION:

To evaluate the function K^X, where K = 2, e, or 10, and $-1 \leq x \leq 1$. To obtain higher values of the exponential function, multiply the output of the subroutine by K to the integer part of the exponent.

EXAMPLES:

$10^{2.5} = 10^{2.} (10.5)$
$2^{-3.5} = 2^{-3.} (2^{-.5})$
$e^{x.xx} = e^{x.} (e^{.xx})$

INPUT:

One word in the accumulator at q = 1.

OUTPUT:

One word in the accumulator at q = 4.

CALLING SEQUENCE:

Location	Order	Address
$\alpha - 1$	B	L (Arg.)
α	R	$(L_o + 09)_{10}$
$\alpha + 1$	U	L_o for 2^x
$\alpha + 1$	U	$(L_o + 2)_{10}$ for e^x
$\alpha + 1$	U	$L_o + 3)_{10}$ for 10^x
$\alpha + 2$	etc.	

α - 1 need not contain a B order. Any order or orders that leaves the argument in the accumulator is permissible.

ACCURACY:

The maximum error is 5×10^{-8}.

TIME:

255 to 285 MS.

STORAGE:

63 locations of instructions and constants.
No temporary storage.

(Program 18.0)
LOGARITHM SUBROUTINE

FUNCTION:

To compute the logarithm of any given number to the base 2, e, or 10. A 7th degree polynomial approximation is used. The argument must be positive. The base to be used must be specified in the calling sequence.

INPUT:

One word in the accumulator at a positive q.

OUTPUT:

One word in the accumulator at q=6.

CALLING SEQUENCE:

Location	Order	Address
$\alpha - 1$	B	L (Arg.)
α	R	$(Lo + 24)_{10}$
$\alpha + 1$	U	Lo Lo= Initial location of Subroutine.
$\alpha + 2$	Z	q q = No. of places in argument.
$\alpha + 3$	Z	K K= (0 for $\log 2^x$)
$\alpha + 4$	etc.	(1 for $\log e^x$)
		(2 for $\log 10^x$)

$\alpha - 1$ need not be a B order. Any order or orders that leaves the argument in the accumulator is permissible.

NOTES:

The argument must be greater than zero. The q (number of places in the argument) must be in the range $0 \leq q \leq 31$. If K, the type of output, is not equal to 0 or 1, the base 10 will be used.

ACCURACY:

The maximum error is 3×10^{-8}.

PROGRAM STOPS:

Location	Meaning
$(Lo + 8)_{10}$	Argument is zero or negative.

TIME:

Approximately (445 + 30 N) MS, where N is the number of leading zeros.

STORAGE:

122 locations of instructions and constants
No temporary storage.

NOTES

ALPHANUMERIC OUTPUT SUBROUTINE
(Program 19.0)

FUNCTION:

To print (or punch and print) alphabetic and/or numeric information.

INPUT:

A set of code words, where each code word consists of 4 alphanumeric output codes.

CALLING SEQUENCE:

Location	Order	Address
α	R	Lo
$\alpha + 1$	U	Lo
$\alpha + 2$	[c o d e w o r d]	
.		
.		
.		
.		
$(\alpha + 1) + n$	[Code word containing VQ]	
$(\alpha + 1) + (n + 1)$	etc.	

Where n is the number of code words.

EXAMPLE:

PROGRAM INPUT CODES	STOP	LOCATION	INSTRUCTION OPERATION	INSTRUCTION ADDRESS	STOP	CONTENTS OF ADDRESS	NOTES
	/						
	/	☒					
		00	R	Lo	/		Enter Alphanumeric
		01	U	Lo	/		Subroutine
000003		02	2010OJSJ		/	Carr.Ret.,Carr.Ret.,Print	
		03	4208OA1J		/	☒ P,LC,+3,"LGP-30",Tab and	
		04	043OVQOO		/	O,Tab,Exit	Exit
		05	etc		/		Continue with Prog.

This calling sequence will perform a carriage return, print "LGP-30" and execute a tab.

OUTPUT:

Printing (or punching and printing) of alphanumeric characters selected.

ALPHANUMERIC OUTPUT CODES:

See "6 Bit Alphanumeric Output Codes" (next page).

EXIT:

The routine will exit to the location following the location containing the exit code (VQ).

STORAGE:

58 locations of instructions and constants.
No temporary storage.

TIME:

About 400 characters per minute.

NOTE :

An increase in output speed can be obtained by switching the instruction in location 0035 with the one in 0036. This will raise output speed to 475 characters per minute. But this change requires that there not be a long carriage return or tab code as the 4th code of a code word.

6 - BIT ALPHANUMERIC OUTPUT CODES

)0	04	Aa	72
L1	0J	Bb	0F
*2	14	Cc	6F
"3	1J	Dd	2F
Δ4	24	Ee	4F
%5	2J	Ff	54
$6	34	Gg	5J
⅞7	3J	Hh	62
Σ8	44	Ii	22
(9	4J	Jj	64
Space	06	Kk	6J
--	0A	Ll	0J
=+	16	Mm	3F
:;	1A	Nn	32
?/	26	Oo	46
].	2A	Pp	42.
[,	36	Qq	74
TAB	30	Rr	1F
Lower Case	08	Ss	7F
Upper Case	10	Tt	5F
Color Shift	18	Uu	52
Carr. Ret.	20	Vv	3A
Back Space	28	Ww	7J
'	40	Xx	4A
Leave Routine	VQ	Yy	12
		Zz	02

ARCSINE - ARCCOSINE SUBROUTINE
(Program 20.0)

FUNCTION:

To compute the arcsine or arccosine of any given value between $-1 \leq X \leq 1$. A 7th. degree polynomial approximation is used.

INPUT:

One word in accumulator at q = 1.

OUTPUT:

One word in the accumulator at q = 9 in degrees

CALLING SEQUENCE:

	Arcsine			Arccosine	
Location	Order	Address	Location	Order	Address
$\alpha - 1$	B	L (Arg.)	$\alpha - 1$	B	L (Arg.)
α	R	$(L_o + 21)_{10}$	α	R	$(L_o + 21)_{10}$
$\alpha + 1$	U	L_o	$\alpha + 1$	U	$(L_o + 0211)_{10}$
$\alpha + 2$	etc.		$\alpha + 2$	etc.	

$\alpha - 1$ need not be a B order. Any order or orders that leaves the argument in the accumulator is permissable.

Accuracy:

The maximum error is approximately 1.2×10^{-6}

TIME:

850 to 900 ms.

STORAGE:

160 locations of instructions and constants, 9 locations of temporary storage (track 63, sectors 12, 16, 18, 19, 20, 21, 23, 24, 28).

PROGRAM STOPS:

Location	Meaning
$(L_o + 0161)_{10}$	Argument is larger than 1 at q = 1.

Note :

Since the square root subroutine is required for the evaluation of either arc-sine or arc-cosine, the coding for the former (Program 15.0) is included in this program (20.0). In those instances in which the square root subroutine is independently required, the following calling sequence may be used for square root extraction:

Location	Order	Address
α -1	B	L(Arg.)
α	R	$(L_o + 0150)_{10}$
α +1	U	$(L_o + 0100)_{10}$
α +2	etc.	

For further information on the square root subroutine see program 15.0.

DECIMAL MEMORY PRINTOUT
(Program 21.0)

FUNCTION:

To print the contents of consecutive memory locations in decimal form.

INPUT:

Beginning and final locations and the modifier (all in decimal)

OUTPUT FORMAT:

A. Locations:
The printed location is equal to the real location minus the modifier used.

B. Instructions:
1. With modifier subtracted if in the range Modifier $\leq$Address$\leq$Final location.

2. If in the range Modifier >Address>Final location
   a. Instructions are preceded by an "x" if modifier $\neq$ 0.
   b. Instructions are not preceded by an "x" if modifier = 0

C. Data:
1. In decimal (at q=0) if transfer control button is up.
   a. Decimal data preceded by sign and decimal point.

2. In hexadecimal if transfer control button is down.
   a. Hexadecimal data preceded by a comma.

Output is six words per line preceded by initial location of the line. Words are separated by spaces. The sign and decimal point are printed for decimal data words and a comma is printed for hexadecimal words. Two carriage returns are given before and after printing.

Note:

Data printed in this manner can be converted to its real decimal value by multiplying by 29.

PROCEDURE:

1. Depress MANUAL INPUT Lever on the Typewriter.
2. Transfer to the first location of this routine.
3. After the "manual input" light comes on, type the initial and final locations (in decimal) into the keyboard.
4. Depress the START COMP. Lever on the Typewriter.
5. After a space is given and the "manual input" light comes on again, type in the modifier in decimal.
6. Make sure the TRANSFER CONTROL button is in the desired position - up for decimal data - down for hexadecimal.
7. Depress the START COMP. Lever on the Typewriter.
8. The position of the TRANSFER CONTROL button may be changed at any time to change the output format of non instructional words.

TIME:

Approximately 60 words per minute.

STORAGE:

256 locations of instructions and constants (4 tracks).
No temporary storage.

FIXED POINT TRACING SUBROUTINE
(Program 23.1)
(J. Wilkinson - University of Michigan)

FUNCTION:

To facilitate the check-out of fixed point programs by providing a printed record of their sequence of instructions and numerical results obtained at each step.

INPUT:

1. Initial contents of accumulator required by the program being traced or monitored. (A) = _____
2. An indication of address radix in the form "D:H = _____" requiring the typewritten symbol "d" or "h".
3. Lower and upper limits defining the interval over which tracing and printing is to be executed by the program.

4. The address at which computation is to begin, symbolized by a "C".

OUTPUT:

	INFORMATION	SYMBOL
1.	The location of the instruction..	C
2.	The instruction expressed in terms of Order & Address.....	(C)
3.	Contents of the address appearing in the instruction.....	(M_1)
4.	Contents of the accumulator after the execution of instruction.....	(A)

The actual typed output has the following arrangement for a sample program:

INSTRUCTION COUNTER 1 C	INSTRUCTION ORDER & ADDRESS 2 (C)	CONTENTS OF INSTRUCTION ADDRESS (i.e.the operand) 3 (M_1)	CONTENTS OF ACCUMULATOR 4 (A)
1002	b 1019	000q0814	
1003	c 1007		
1004	h 0800		
1005	b 0800	0000 0000	
1006	m 0804	0kw3 g646	0000 0000
1007	a 0805	0189 374j	0189 374j
1008	h 0800		
etc.	etc.	etc.	etc.

PROCEDURE:

(1) Load
   a. Tracing Subroutine.
   b. Program to be traced.
   c. Subroutines required by traced program.

(2) Transfer to the first instruction of the tracing program.

Five items of information are to be typed in next, one at a time, following the printing by the typewriter of a standard program symbol. A single depression of the START COMPUTE lever is to follow each standard symbol.

	STANDARD SYMBOL	TYPE IN
(3)	(A) = _____	Starting contents of accumulator if such are required by program traced.
(4)	D:H = _____	"d" or "h" to designate decimal or hexadecimal addresses.
(5)	A = _____	The address of first traced instruction in the agreeable radix. This may be the address of the first instruction in a subroutine used by the main program being traced.
(6)	B = _____	The address of the last instruction traced. This may be the last instruction in a subroutine used by the main program rather than the last instruction in the latter.
(7)	C = _____	The address at which computation is to begin in the main program being traced.

If the START COMPUTE lever is successively depressed immediately after the typewriter prints the symbols "(A)", "D:H", "A", and "C" the following values will be automatically assigned:

(A)	00000000
D:H	d
A	0000
C	0000

The value of "B" cannot be assigned in this way unless the single instruction 0000 is the only one which the tracing program is to report upon. The value of "B" must be typed and the START COMPUTE lever depressed.

Depression of the START COMPUTE lever following the typing of the "C" value will cause the computer to begin executing the tracing program. Each instruction in the interval A-B will be taken in turn, traced, and a record printed. After the tracing and printing has been completed for each instruction the computer will carriage return and stop as the result of a programmed break-point stop, z3200. This stop will occur after the completion of the tracing-printing for each instruction. If it is desired that these stops be omitted, and that continuous tracing-printing be done, then the break-point 32 switch should be depressed.

When the control counter is set to a value outside of the interval A-B, printing will usually be suspended. However, if the control counter is outside of the interval A-B and the computer attempts, through a C,H,R, or Y instruction, to write information into a memory location within the interval A-B, then that instruction will be traced. A break-point - 16 stop will be executed just preceding the execution of the instruction. Furthermore, in the case of an R or Y instruction, the contents of the operand will be printed as an instruction after completion of the instruction. Finally when the control counter re-enters the interval A-B the current contents of the accumulator will be printed on a separate line before tracing the next instruction.

The purpose of providing a means of supressing monitoring when outside of a specific range is to permit already-checked-out routines to be run at greater speed, and to reduce the volume of output to a minimum consistent with an understanding of the program being monitored.

The large number of possible combinations of printing modes occurring when the program control counter's instruction is inside or outside of the interval A-B and the possibilities of printing before and/or after instruction-execution can best be summarized in tabular form. Accordingly, the following classification or analysis is presented for clarity.

PRINTING DONE BY THE TRACING PROGRAM
WHEN THE INSTRUCTION APPEARING IN THE
CONTROL COUNTER IS IN THE INTERVAL A-B

LGP-30 ORDER STRUCTURE OPERATION	PRINTING DONE BEFORE COMPUTER EXECUTES INSTRUCTION			PRINTING DONE AFTER COMPUTER EXECUTES INSTRUC:
	Instruction Counter	Instruction	Contents of Inst. Address	
A,D,E,M,N,S	C	(C)	(Mn)*	A
B	C	(C)	(Mn)	---
I,P,Z	C	(C)	---	---
C,H	C	(C)	---	---
R	C	(C)	---	(Mi)*
Y	C	(C)	---	(Mi)
T	C	(C)	---	---
U	C	(C)	---	---

*
(Mi) represents the contents of the memory location referenced by the current instruction, printed as an instruction; (Mn) represents the contents of the memory location referenced by the current instruction.

PRINTING DONE BY THE TRACING PROGRAM
WHEN THE INSTRUCTION APPEARING IN THE
CONTROL COUNTER IS OUTSIDE THE INTERVAL A-B

LGP-30 ORDER STRUCTURE OPERATION	PRINTING DONE BEFORE COMPUTER EXECUTES INSTRUCTION			PRINTING DONE AFTER COMPUTER EXECUTES INSTRUCTION
	Instruction Counter	Instruction	Contents of Instr.Address	
C,H	C	(C)	(A)	---
R	C	(C)	---	(Mi)
Y	C	(C)	(A)	(Mi)
T	C	(C)	---	*
U	C	(C)	---	*

* In addition to the above printing, the contents of the accumulator will be printed on a separate line preceded by a "(A)=" if the control counter enters that range after having been outside of the A-B interval.

SPECIAL FEATURES:

In order to permit running through stop instructions following print orders, when the control counter is within the A-B interval, all z0000 instructions are done as though they were written z0400 and all other stop instructions are done as though written z0800. When the control counter is out of the interval A-B all stops are ignored.

When the control counter address is outside the interval A-B, all print instructions are done as written. When inside the limits, format-control instructions (space, lower case, upper case, color chift, carriage return, back-space, and tabulate) will not be done. Instead, an abbreviation of the function will be printed, as "tab" for tabulate, etc.. All monitor printing is done in lower case, but a record is kept of the most recent case-shift, and print instructions are done in either upper or lower case accordingly.

The "TRANSFER CONTROL" switch functions normally with respect to the program being monitored.

STORAGE: Nine tracks of storage, constants, and temporaries.

TIME: About 3.8 seconds per instruction when monitor-printing, and about 0.7 seconds per instruction when not printing.

FLOATING POINT TRACE ROUTINE

(Program 23.4)

FUNCTION:

To trace all floating point instructions defined by program 24.0 except 800t and Rxxxx.

INPUT:

A tape punched with the decimal locations of instructions to be traced, each followed by the conditional stop code. The final location is followed by 80000000.

PROCEDURE:

Enter the trace routine at Lo and read in the locations tape. As each location is read, the instruction in that location is tagged with a 1 at 2 and restored in memory. After the final location has been tagged, a halt is executed in Lo + 106 and pressing the start switch transfers control to the program input routine.

OUTPUT:

Execution of a tagged instruction is followed by a carriage return, print of the decimal location, a tab and a print of the contents of the floating point accumulator.

DE-TAG PROCEDURE:

Enter the trace routine at Lo + 25 and re-read the locations tape. Instructions are detagged and program 24.0 is restored. Tracing may be halted without detagging by entering Lo + 25 and giving zero as an address. Also, tracing may be resumed by entering Lo and giving zero as an address.

TIME:

One extra drum revolution per instruction plus printing time.

STORAGE:

127 locations of instructions and constants. 3 locations of temporary storage. (Track 63, sectors 10, 52, 58).

NOTE:

The symbol L_f used in the coding refers to the initial location of program 24.0. The decimal program tape assumes L_f = 4000. If this is not the case the tape must be repunched with the correct address. See coding sheets for those instructions which refer to program 24.0.

* This routine was suggested by Mr. A. J. Ness of Reaction Motors, Inc., and modified at Lehigh University.

NOTES

FLOATING POINT INTERPRETIVE SYSTEM
(Program 24.0)

PART 1

SECTION I: FUNCTION

The function of this floating point system is the re-interpretation of the LGP-30 fixed point order structure so that it may be programmed as a floating point computer. This re-interpretation is effected by:

(1) The provision of a multiplier register and an address register as well as a floating point accumulator.

(2) The provision of more types of orders including cumulative multiply, shift, sign change, and function generating orders.

(3) A broadening of the scope of certain instructions such as the input instruction and the print instruction.

SECTION II: GENERAL CHARACTERISTICS

Floating point programming has several advantages over fixed point programming in that it is more rapid, does not require an exact knowledge of the range of magnitude of the variables, and does not involve as much truncation of the smaller values when that range is large. Thirty-three orders are provided for in the system. All of these orders except input, output, sine, cosine, arctangent, logarithm, and exponential are included in that section of the system known as the floating point interpretive routine. This routine requires only 10 out of the 64 tracks of LGP-30 memory leaving 3456 words available for problem program and data storage. The input and output orders require 6 tracks and the floating point functions require 7 tracks. The entire system leaves 2624 words of memory left for problem program and data storage.

Generally the execution of a floating point program will take 10 to 20 times as long as the execution of the corresponding fixed point program.

Times for the execution of the individual floating point orders are included in the summary tabulation at the end of Part 2.

SECTION III: REGISTERS

(1) The floating point accumulator occupies 2 words of memory, one for the characteristic of a floating point number and one for the exponent. The floating point accumulator is similar in function to the fixed point accumulator; it holds intermediate results.

(2) The multiplier (M) register occupies 2 words of memory, one for the characteristic of a floating point number and one for the exponent. The multiplier register holds the multiplier for the reset and multiply order and for the cumulative multiply order.

(3) The address accumulator occupies 1 word of memory and holds a single address or tally which is the same in form as for fixed point operations.

(4) The contents of none of these registers is changed unless replaced by a new result. For example, the M register remains unchanged following execution of a square root or multiply instruction. Nor is the contents of any memory location changed except when affected as specifically noted in the order description in the section that follows.

SECTION IV: FLOATING POINT ORDERS

Thirty-three orders are available. The list of these orders and their meaning follows. In the following exposition the term "accumulator" refers to the two memory cells of the floating point accumulator as defined above.

A. Arithmetic Instructions
Memory location XXXX is the address of one floating point number in standard form as defined in Part 2.

1. B XXXX. Bring
The contents of memory location XXXX replace the contents of the accumulator.

2. A XXXX. Add.
The contents of the accumulator plus the contents of memory location XXXX replace the contents of the accumulator.

3. S XXXX. Subtract.
The contents of the accumulator minus the contents of memory location XXXX replace the contents of the accumulator.

4. D XXXX. Divide.
The contents of the accumulator divided by the contents of memory location XXXX replace the contents of the accumulator.

5. P XXXX. Place.
The contents of memory location XXXX replace the contents of the M register.

6. M XXXX. Reset and Multiply
The contents of the M register multiplied by the contents of memory location XXXX replace the contents of the accumulator.

7. N XXXX. Cumulative Multiply
The contents of the M register multiplied by the contents of memory location XXXX and added to the contents of the accumulator replace the contents of the accumulator.

8. D 000y. Right Shift
The contents of the accumulator divided by 2^y replace the contents of the accumulator. The contents of accumulator remain in floating point form.
$0 \leq y \leq 9$

9. M 000y. Left Shift
The contents of the accumulator multiplied by 2^y replace the contents of the accumulator. The contents of accumulator remain in floating point form.
$0 \leq y \leq 9$

10. H XXXX. Hold
Place the contents of the accumulator in memory location XXXX.

11. C XXXX. Clear
Place the contents of the accumulator in memory location XXXX and set the accumulator to zero.

B. Logical or Transfer Instructions

12. U XXXX. Unconditional Transfer
The next instruction to be interpreted is in memory location XXXX. This order cannot be used to exit from the floating point interpretive system.

13. T XXXX. Test
The next instruction to be interpreted is in memory location XXXX if the accumulator is negative. Otherwise the first successive location will be interpreted.

14. 800T XXXX. Transfer Control
The next instruction to be interpreted will be in memory location XXXX if either the accumulator has a negative characteristic or the transfer control switch is down. Otherwise the first successive location will be interpreted.

C. Address Modification Instructions
Location XXXX implies a fixed point address.

15. E XXXX. Enter
The address portion of memory location XXX replaces the contents of the address accumulator.

16. I XXXX. Increment
The address accumulator is incremented by the address XXXX. This order can be used to decrement the address accumulator by complementing the address portion of the I XXXX order.

17. Y XXXX. Store Address
The address portion of the address accumulator replaces the contents of the address portion of memory location XXXX.

SECTION IV: FLOATING POINT ORDERS

18. Z XXXX. Zero Test
The address of the "Z" instruction is subtracted from the contents of the address accumulator. If the result is not zero, the first successive instruction is interpreted. If the result is zero, the first successive instruction is skipped and the second successive instruction is interpreted.

D. Auxiliary Instructions

19. R XXXX. Return Address
The location of this instruction is increased by 2 and is stored in the address portion of memory location XXXX.

20. U 0000. Reverse Registers
The contents of the M register and accumulator are interchanged.

21. B 0000. Set Sign Plus
The sign of the accumulator is made positive if not already so.

22. T 0000. Set Sign Minus
The sign of the accumulator is made negative if not already so.

23. Y 0000 Change Sign
The sign of the accumulator is reversed.

24. Z 0000. Stop
Computation is halted unless break point switch No. 16 is down. Depressing the start button causes the next instruction to be interpreted.

25. E 0000. Exit
Exit from the floating point interpretive system. Control is returned to the location following the location of the E 0000 instruction.

E. Input-Output Instructions

26. I 0000. Input
Control is transferred to a floating point data input subroutine which reads decimally punched numbers on tape, converts them to floating binary, and stores them. The next instruction is interpreted after the proper exit code has been read from tape. See Section V, Part 1 for tape format and input details.

27. P 0000. Print
Print the contents of the accumulator. The contents of the accumulator are not destroyed. See Section VI, Part 1 for Output format.

SECTION IV: FLOATING POINT ORDERS

F. Function Evaluation Instructions

28. R 0000. Square root
The square root of the contents of the accumulator replaces the contents of the accumulator.

29. S 0000. Sine
The sine of the contents of the accumulator replaces the contents of the accumulator. The accumulator must be in radian measure.

30. C 0000. Cosine
The cosine of the contents of the accumulator replaces the contents of the accumulator. The accumulator must be in radian measure.

31. A 0000. Arctangent
The arctangent of the contents of the accumulator replaces the contents of the accumulator. Output is in radian measure.

32. N 0000. Natural Logarithm
The natural logarithm of the contents of the accumulator replaces the contents of the accumulator.

33. H 0000. Exponential
The quantity e^x replaces the contents of the accumulator, where x is initially the contents of the accumulator.

SECTION V: DATA INPUT FORMAT

Data Input is accomplished by reading a prepunched decimal tape. The tape consists of groups of the following:

1. One identification word. This consists of a sign and two decimal digits for P, followed by four decimal digits for initial location to begin storing the converted floating floating point binary numbers.

2. Signed decimal numbers. Each number consists of a sign (if negative) and seven decimal digits.

3. A "minus zero" word. This consists of a minus sign followed by seven zeros. This number is not stored in memory, but is used by the routine to signal the end of the group.

A stop code must follow the last "minus zero" word. This is interpreted as a "zero" identification (I.D.) word since it follows the "minus zero" data word. It causes the system to exit from the subroutine, carriage return, and interpret the instruction following the I 0000 instruction.

P denotes the number of decimal places following the point in the seven digit field. $-6 \leq P \leq 16$. Internally the exponent must be in the range $-32 \leq \text{Exp.} \leq 31$.

SECTION VI: DATA OUTPUT FORMAT

The printed output consists of a decimal point followed by seven decimal digits of the characteristic and its sign. Following the sign there are two spaces followed by the exponent and its sign (if the sign is negative). e.g. .5060000- 02 is -50.60000. A tab is executed after printing.

FLOATING POINT INTERPRETIVE SYSTEM
(Program 24.0)

PART 2

Note: Refer to Part 1 for FUNCTION, REGISTERS, ORDERS, and INPUT and OUTPUT FORMAT.

INPUT:

Floating point numbers on tape or in memory, or numbers in the pseudo registers resulting from previous operations.

CALLING SEQUENCE:

Location	Order	Address	
α	R	Lo	
α + 1	U	Lo	
α + 2	.	.	Floating point operations
α + 3	.	.	
.	.	.	
.	.	.	
.	.	.	
α + n	E	0000	"Exit" instruction
α + n + 1	etc.		Resume fixed point operation.

INTERNAL NUMBER FORMAT:

A standard floating point number as carried in memory consists of sign and 24 bits for characteristic (x) and sign and 5 bits for the exponent (y). However, all intermediate calculations (i.e., numbers appearing only in accumulator and multiplier registers) are carried with 30 bits of characteristic and 30 bits of exponent. Each factor of any calculation must be in standard floating point form. ($N = x \cdot 2^y$; $.5 \leq |x| < 1.$ or $x = 0$; $-32 \leq y < 31$). Numbers appearing in accumulator or M registers are in the range $.25 \leq |x| < .5$ or $x = 0$. N is defined as the original unscaled number.

The standard floating point binary form:

X	.XXX.....XX	X	XXXXX
Sign of Characteristic 0 for plus 1 for minus	Characteristic 24 bits.	Sign of exponent 0 for plus 1 for minus	Exponent 5 bits Power of 2

DATA TAPE PREPARATION:

All characters of the I.D. word should be punched. E.g. -012040' must contain eight characters including the stop code. The stop code (') must be the last character punched.

2. Punch only those I.D. words appearing on the load sheet. Do not punch the stop code if an I.D. word is not present.

3. The sign and any leading zeros of a positive number need not be punched. To enter all zeros merely punch a stop code. The sign and all seven digits of a negative number must be punched.

   Be sure to check each load sheet to see whether an additional stop code should follow the last number punched.

EXIT:

The interpretive routine exists to the first location following the E 0000 instruction.

SUBROUTINE MEMORY RELATIONSHIPS:

The arithmetic, logical, address modification, and auxiliary instructions have been coded as a unified group on a single set of coding sheets ("Floating Point Interpretive Routine"). A single corresponding tape has been punched for this set. In many instances the programmer will wish to use just this part of the floating point system, if so, only this tape need be stored in the memory. This will leave 54 tracks for program instructions and data in contrast to 41 when the entire system is used.

In other cases the Input-Output and/or function evaluation routines may be needed. Only those routines actually used need be stored on the drum. These required routines must be stored on the drum in the following relationship:

Program	Routine	Start Fill	Set Modifier	No. of Tracks
24.0	Interpretive (Includes $\sqrt{}$)	Lo	Lo	10
11.6-12.6	Input-Output	Lo + 1000	Lo + 1000	6
14.1	Sine-Cosine	Lo + 1600	Lo	2 1/2
16.2	Arctangent	Lo + 1832	Lo	1 1/2
18.1	Logarithm	Lo + 2000	Lo	1
17.1	Exponential	Lo + 2100	Lo	2

All track 63 except sectors 10, 15, 16, 23, 27, 29, 34, 36, 40, 47 thru 50, 52, 56 thru 58, 60 and 63 is used for temporary storage by various parts of the system. Therefore Lo should be set such that no part of the floating point system used is stored in track 63.

PROGRAM STOPS:

Location	Order	Meaning and Remedy
Lo + 0654	Z 0000	Programmed stop. Depress "start" to continue.
Lo + 0557	H XXXX or C XXXX	Exponent is too large. Location of instruction being executed is in the real accumulator. Start to continue.

Lo + 0557	R 0000	Accumulator is negative. Location of instruction being executed is in the real accumulator. Start to continue.
Lo + 1152	I 0000	Input data has too large an exponent. A start will store a zero for that word and continue with next word on tape.
Lo + 0612	D XXXX	Division by zero or a non-floated number. Do not continue.
Lo +2005	N 0000	Accumulator is ≤ 0. A start continues with an answer of zero.
Lo + 2028 or Lo + 2030	N 0000	Accumulator exponent is not in range. Do not continue.

TIME:

See summary tabulation.

EXAMPLE:

See the following LGP-30 coding sheet.

NOTES:

1. The floating point system may be left and re-entered without destroying the contents of the registers.

2. The exponent of a number in a register which is to be stored in memory must be less than +32, or a range error will result. If it is less than -32, the number is replaced by zero.

3. It is strongly suggested that the initial location occupied by the system be the 00 sector of a track. If it is not, many of the addresses that refer to track 63 are not optimum.

4. It is also suggested that the entire system be placed in memory and punched out in parts by program 13.2. Then the parts needed may be loaded by program 10.4 and each check sum may be verified.

5. All instructions with zero addresses have special interpretations. None of these zero addresses refer to memory location "zero", (0000), but rather designate a special interpretive instruction. This floating point system employs sixteen special instructions. Furthermore, the two shift instructions (D 000y, M 000y) utilize the next nine addresses (0001 through 0009); hence the divide and reset and multiply instructions cannot use these addresses.

SUMMARY TABULATION

ORDER	RESULT IF ADDRESS (α) $\neq$ 0	TIME	RESULT IF ADDRESS = 0*	TIME
Z	C (Add.Acc) - (α) = 0? No:No Skip Yes: Skip	133 ms	Stop (SW No. 16).Proceed on start	117 ms
B	C(α)$\rightarrow$Acc.	233 ms	Make C (ACC.) positive	150 ms
Y	C (Add.Acc.)$\rightarrow$ Add. of (α)	150 ms	Complement C (Acc.)	150 ms
R	(Loc. of R) + 2$\rightarrow$Add. of (α)	166 ms	$\sqrt{C\ (Acc.)}\rightarrow$ Acc.	500 ms
I	C(Add. Acc.) + (α)$\rightarrow$ Add. Acc.	150 ms	Input floating point data	40/min.
D	C(Acc.) $\div$ C (α)$\rightarrow$Acc.	283 ms	C(Acc.) $\div$ $2\overset{\alpha}{\rightarrow}$Acc.	183 ms
N	C(M) x C (α) + C(Acc.)$\rightarrow$ Acc.	566 ms	ln C(Acc.)$\rightarrow$Acc.	500 ms
M	C(M) x C (α)$\rightarrow$ Acc.	266 ms	C(Acc.) x 2$\rightarrow$Acc.	150 ms
P	C(α) $\rightarrow$ M	217 ms	Print C(Acc.)	1.85 sec.
E	C[Add. (α)]$\rightarrow$ Add. Acc.	150 ms	Exit from interpretive routine	117 ms
U	Next abstract order taken from (α)	117 ms	C(Acc.)$\rightarrow$ M; C (M) $\rightarrow$ Acc.	200 ms
T	Transfer if C(Acc.) is negative	133 ms	Make C(Acc.) negative	150 ms
H	C(Acc.) $\rightarrow$ (α)	200 ms	$_{e}$C (Acc.)$\rightarrow$ Acc.	450 ms
C	C(Acc.)$\rightarrow$(α); 0$\rightarrow$Acc.	233 ms	Cosine C(Acc.)$\rightarrow$ Acc.	517 ms
A	C(Acc.) + C (α)$\rightarrow$Acc.	400 ms	Arctangent C(Acc.)$\rightarrow$ Acc.	450 ms
S	C(Acc.) - C(α)$\rightarrow$Acc.	417 ms	Sine C(Acc.)$\rightarrow$ Acc.	550 ms

Add. Acc. = Address Accumulator register
M = Multiplier register
Acc. = Floating point Accumulator
= any address
C = Contents of

*Address = 0: except for instructions "M 000α" and "D 000α", where $0 < \alpha < 9$.
$\longrightarrow$ = Is stored in

All times are approximate and will vary with the amount of overflow and/or underflow. Actual times should be slightly less than listed. Time will usually be reduced if any factor is zero.

LGP-30 DATA LOAD SHEET — Floating Point Data Input

ILLUSTRATIVE EXAMPLE FOR FLOATING POINT INTERPRETIVE SYSTEM

PREPARED FOR:

PAGE / OF

JOB NO.	PROGRAM NO.	PROGRAM PREPARED BY:	PROGRAM CHECKED BY:	DATE
ONE	24.0	G.L.W.	M.K.	REV:6/26/59

PROBLEM: CODING FOR 4th. DEGREE POLYNOMIAL

DATA INPUT NO.

NOTES	± P	LOCATION	STOP	± NUMBER	STOP	CGE RET
X	+ P 0 1 7	2 0 0 4	/	1 0 9 0 0 0 0 0	/	
A_0			/	1 2 0 0 0 0 0	/	X
A_1			/	3 4 0 0 0 0 0	/	
A_2			/	5 6 0 0 0 0 0	/	X
A_3			/	7 8 0 0 0 0 0	/	
A_4			/	9 0 0 0 0 0 0	/	X
			/	- 0 0 0 0 0 0 0	/	
			/		/	X
			/		/	
			/		/	X
			/		/	
			/		/	X
			/		/	
			/		/	X
			/		/	
			/		/	X
			/		/	
			/		/	X
			/		/	
			/		/	X
			/		/	
			/		/	X
			/		/	
			/		/	X
			/		/	
			/		/	X
			/		/	
			/		/	X
			/		/	
			/		/	X
			/		/	
			/		/	X

-3≤P≤ 15

0000≤Loc≤6363

Royal McBee Corporation
DATA PROCESSING DIV.
PORT CHESTER, NEW YORK

FORM LP-15

PUNCH A STOP CODE AFTER THE LAST NUMBER?
YES [X] NO [ ]

LGP-30 CODING SHEET

PREPARED FOR: ILLUSTRATIVE EXAMPLE FOR FLOATING POINT INTERPRETIVE SYSTEM

PAGE 1 OF 1

JOB NO.	PROGRAM NO.	PROGRAM PREPARED BY:	PROGRAM CHECKED BY:	DATE
	24.0	G.L.W.	M.K	REV:6/26/59

PROBLEM: CODING FOR 4th. DEGREE POLYNOMIAL

TRACK

PROGRAM INPUT CODES	STOP	LOCATION	INSTRUCTION OPERATION	INSTRUCTION ADDRESS	STOP	CONTENTS OF ADDRESS	NOTES
; 0 0 0 2 5 0 0	/						
/ 0 0 0 2 5 0 0	/	X					
		0 0	X R	B 0 0 0	/		Enter Interpretive Routine
		0 1	X U	B 0 0 0	/		
		0 2	X E	0 0 0 0	/		Input Coefficients
		0 3	E	0 0 1 7	/	X 2005	Set Initial Address
		0 4	V	0 0 0 6	/	2005	
		0 5	B	0 0 1 B	/	Zero	
		0 6	X A	[ ]	/	An	Add n'th coefficient
		0 7	X E	0 0 1	/	X a[2005+n]	Increase address accumulator by unity
		0 8	V	0 0 0 6	/	a[2005+n]	Store contents of add.acc in 0006
		0 9	X Z	E 0 1 0	/		Test for finish.
		1 0	U	0 0 1 4	/	Not finished	
		1 1	X P	0 0 0 0	/	X Print result here:Finished	
		1 2	X E	0 0 0 0	/	Exit	
		1 3	X Z	0 0 0 0	/	Stop	Fixed point inst.
		1 4	X U	0 0 0 0	/	Move Accum.to Register	
		1 5	X M	2 0 0 4	/	X Multiply by x	
		1 6	U	0 0 0 16	/	Return for next term.	
		1 7	X Z	2 0 0 15	/	Initial coefficient.	
		1 8			/		
		1 9			/	X	
		2 0			/		
		2 1			/		
		2 2			/		
		2 3			/	X	
		2 4			/		
		2 5			/		
		2 6			/		
		2 7			/	X	
		2 8			/		
		2 9			/		
		3 0			/		
		3 1			/	X	

FORM LP-10

Royal McBee Corporation
DATA PROCESSING DIV.
PORT CHESTER, NEW YORK

X = CARRIAGE RETURN
/ = CONDITIONAL STOP CODE

FLOAT AND UNFLOAT SUBROUTINES
(Program 25.0_R)

FUNCTION:

A. To convert a fixed point binary number to standard floating point form as defined in program 24.0. This is called "floating" a number, or

B. To take a floating point number as used by program 24.0, and convert it to fixed point form. This process is called "unfloating" a number.

INPUT:

The number to be operated upon in the accumulator.

CALLING SEQUENCES:

	A. Float	B. Unfloat
α - 1	B L (No.) (Fixed point number)	α - 1 B L(No.)(Floating point no.)
α	R (Lo + 25)$_{10}$	α R (Lo + 0148)$_{10}$
α + 1	U Lo	α + 1 U (Lo + 0122)$_{10}$
α + 2	Z q (q of fixed point number)	α + 2 Z q (q of unfloated number)
α + 3	etc.	α + 3 etc.

α - 1 need not contain a B order. Any order which leaves the number to be floated, or unfloated, in the accumulator is permissable.

OUTPUT:

A. The number in standard floating point form in the accumulator, or
B. The number in fixed point form at the q specified in calling sequence B in the accumulator.

PROGRAM STOPS:

Loc.	Routine	Meaning
Lo + 0512	Unfloat	Number is too large to unfloat to the specified q.
Lo + 0262	Float	Exponent > 32, so cannot be expressed as a floating point number.

EXIT:

For either subroutine, exit is to α + 3.

TIME: Float -- (150 + 16 n) MS where n is the number of leading zeros.
Unfloat -----150 MS.

STORAGE: 192 locations of instructions and constants, (3 tracks).
No temporary storage.

SEARCH FOR ADDRESS
(Program 26.2)

FUNCTION:

To search the drum and determine if a given address is in the address portion of the words in locations 0000 through 6263. If such a word is found, its location and the operation code corresponding to bits 11 through 15 of that word are printed; the search then continues. Bits outside the operation and address portions of the words are disregarded. The search does not alter the contents of the locations searched.

INPUT:

The decimal track and sector for which the search is to be made, e.g., 0237 for track 2, sector 37.

OUTPUT:

All locations in 0000-6263 containing the given address will be printed in decimal followed by the operation code found in that location. The printing of each location and operation is preceded by a tabulation.

PROCEDURE FOR MANUAL OPERATION:

Transfer to the first location of the routine with the MANUAL INPUT lever of the typewriter down. When the input light glows, type the four character decimal address and then depress the START COMP. lever on the typewriter. When the search is completed, the input light on the typewriter will again go on, and a new address for the search may then be typed.

STORAGE:

64 locations of storage are required. No temporary storage is needed, so the routine may occupy track 63.

TIME:

Each search requires about 2 3/4 minutes exclusive of printing.

NOTES:

This program requires that Program Input Routine (Program 10.4) be in locations 0000-0263. This program may be relocated. However, if locations Lo + 0016 and Lo + 0048 are not in track 63, then they will be printed as two of the locations where the given address occurs.

If it is desired to search for all references to a given track, insert xz6300 in location Lo + 0021. The input must still be the four character decimal track and "sector" desired, e.g., 2300 for track 23.

NOTES