LGP-30 USERS' ORGANIZATION - POOL

LGPSAP

SYMBOLIC ASSEMBLY PROGRAM FOR THE LGP-30 COMPUTER

POOL Program No. H2-120

THIS PROGRAM IS DISTRIBUTED TO
MEMBERS OF POOL ONLY; D STRIBUT O
TO NON MEMBERS OF POOL IS PROHIBIT

By James N. Orton
Royal McBee Corp.
Washington, D.C.

February 5, 1960

LGP-30 USERS' ORGANIZATION - POOL

Program No. H2-120

LGPSAP

SYMBOLIC ASSEMBLY PROGRAM FOR THE LGP-30 COMPUTER

Table of Contents

Introduction ---------- 1.

Purpose ---------- 2.

General Description ---------- 2.

Input
Symbolic Checks

Specific Description: Sample Input & Output ---------- 2.

Sample Assembly 3*
Output: First Pass 4
Output: Second Pass 4
Tape Format 4
Note on the use of Photoreader 4
End-of-program symbol 4
Symbol specifications 4

Explanatory Notes ---------- 4.

1. Starting location and reference address 4
2. Undefined symbols 5
3. Multiply-defined symbols 5
4. Instructions with absolute addresses 5
5. Hexadecimal constants 5
6. Instruction-form constants 5
7. Temporary storage locations, counters, etc. 5

Operation of the Assembler ---------- 5.

Assembly Outside of the "Allowable Range" ---------- 6.

Preparation of Repositionable Punched Tape ---------- 6.

Error Stops ---------- 6.

Symbol Table ---------- 7.

LGPSAP Subroutines ---------- 7.

Time ---------- 7.

Appendix: The Random Address Generator ---------- 8.

Table of Symbols for the LGPSAP Flowchart ---------- 9.

Flow Chart of LGPSAP ---------- 10-13.

Coding for Assembly Routine (LGPSAP and Decimal) --------- 14-20.

Note on Subroutine D3 ---------- 21.

* Arabic symbols following subsection titles are page numbers

Conventional LGP-30 Coding Sheets for LGPSAP ---------- 22-32.

LGP-30 USERS' ORGANIZATION - POOL

L G P S A P

SYMBOLIC ASSEMBLY PROGRAM FOR THE LGP-30 COMPUTER

James N. Orton, Royal McBee Corp.

INTRODUCTION

An essential step in the writing of a program in machine language is the use of a symbolic version of the program for the preparation of the final coding sheets. The arbitrary symbols used for addresses are selected to be appropriate to the problem being solved.

The program described in the following pages relieves the programmer of the work of final translation to numerical addresses; a slight modification of the usual symbolic program with machine language instructions and arbitrary symbolic addresses is entered into the LGP-30 computer. The output is a finished program typed in coding sheet format with numerical addresses (not optimized). As part of the input, absolute addresses or hexadecimal patterns may be entered as required in place of the symbolic program steps. Thus the programmer can readily establish linkages with standard subroutines, and in general, has all the "bells and whistles" of the machine at his disposal. Format control is feasible so that comments on the individual program steps are typed on the final coding sheets. The program as it is assembled is stored in the LGP-30 where it may then be executed or punched out using POOL Program K2-71, the Repositionable Decimal Memory Punch, or 13.2, the Hexadecimal Punch.

Programs in languages similar to the LGP-30 machine language may also be used; for example, a program for 24.0, the Floating Point Interpretive Routine, can be assembled by LGPSAP. Unfortunately, 24.1 cannot be assembled because the program does not handle the 800xmmmm instructions, other than the 800Tmmmm.

PAUL SELIGMANN

Chairman of the POOL Committee
on Publications

February 5, 1960.

Program No. H2-120

LGP-30 USERS' ORGANIZATION - POOL

L G P S A P

SYMBOLIC ASSEMBLY PROGRAM FOR THE LGP-30 COMPUTER

James N. Orton, Royal McBee Corp.

PURPOSE:

To input a symbolic code, translate the symbolic addresses into the numeric form required by the LGP-30, store the translated instructions sequentially starting from a specified memory location, and output the code in both symbolic and standard LGP-30 decimal form.

GENERAL DESCRIPTION:

Input. The input to LGPSAP is a symbolic program with a carriage return after each instruction so that it prints one instruction per line. Each symbolic instruction consists of: (1) a symbolic location, if the instruction is referenced by another, (2) the standard LGP-30 operation code, and (3) the symbolic address for flexowriter input, comments may be written to the right of the instruction if desired.

Preceding the program to be input, two words must be input (either manually or from the tape):

1. The starting location of the program, in decimal. The program is assembled beginning at this location.
2. The "reference address" of the program, in decimal. The "reference address" is equivalent to the "modifier" in the Program Input Routine (10.4). If the program is to be executed after assembly, the"reference address" will generally be the same as the starting location. For preparation of a punched tape, a reference address of 0000 may be desirable. These matters are discussed later.

For assembly, the program tape must be input twice to the assembler. Six-bit input is required. On the first pass, the assembler compiles a table of the symbols used in the program, and another table of the addresses represented by the symbols. On the second pass, the assembler stores and outputs the program, substituting the proper memory addresses for the symbolic addresses. Constants, in either hexadecimal or instruction form, and absolute-address instructions may be input as well as symbolic instructions.

Symbolic Checks During input the assembler makes two symbolic checks as a logical aid to the programmer. It notes (1) the use of "multiply-defined" symbols and (2) the use of "undefined" symbols. These terms will be explained by illustration below.

SPECIFIC DESCRIPTION: SAMPLE INPUT AND OUTPUT

The following sample assembly is illustrative of the principal functions of the assembler. A tape of this assembly is included in the program package. An explanation of these immediately follows:

Note on tape contents: The tape distributed with this program description has information punched in the following order:
1. The basic LGPSAP assembler, "LS"; 2. The modified 21.0,"D3"; 3. "Sample Assembly" (see top page three) 4. Coding for Assembly Routine (LGPSAP and Decimal), see pages 14-20.

SAMPLE ASSEMBLY

Double spacing was used on the second pass.

Flexowriter settings: Margin 6, Tabs 12, 16, 22, 32.

.0000700

p1 (First Pass)

				Notes (see below)
4000'4000'				1
'alpha'	b'	beta'	bring beta	
	h'	gamma'	store into gamma	
	xp'	0000'		
	xi'	0000'		
'beta'	xu'	6363'	exit	3
'lx'	3089'	q172'	mask	
'y'	'	'	at q-29	
	z'	'		
'delta'	u'	delta'	variable connector	
'c2'	xz'	0001'		
'beta'	a'/m	gamma'		3
'n'	'	'	counter	
'4zqx'	'	11q2'		
'end''	'			

p2 (Second Pass)

Symbolic Code Loc.	Op.	Addr.	Decimal Code Loc.,	Op., Addr.	Comments	
'alpha'	b'	beta'	4000	b4004	bring beta	1,3
	h'	gamma'/u	4001	,qqqqqqqq	store into gamma	2
	xp'	0000'	4002	p0000		4
	xi'	0000'	4003	i0000		4
'beta'	xu'	6363'	4004	u6363	exit	3,4
'lx'	3089'	q172'	4005	,3089q172	mask	5
'y'	'	'	4006	z0000	at q-29	7
	z'	'	4007	z0000		
'delta'	u'	delta'	4008	u4008	variable connector	
'c2'	xz'	0001'	4009	z0001		6
'beta'	a'	gamma'/u	4010	,qqqqqqqq		2
'n'	'	'	4011	z0000	counter	7
'4zqx'	'	11q2'	4012	,000011q2		5

Output: First pass. The sample printout following "p1" is the Flexowriter output from the first pass. Output in addition to that from the tape itself will occur only if a multiply-defined symbol or other input error (see below) is detected.

If the Photoreader is used for first-pass input (but first see "Note on use of the Photoreader" below), no printout other than "p1" and the error indications will occur.

Output: Second pass. The printout following "p2" is the Flexowriter output from the second pass. The output, in addition to that from the tape, consists of the standard decimal representation of the program as stored in memory. The symbolic and decimal representations of each instruction appear side by side. In addition identifying characters are printed next to the symbolic address whenever an undefined address symbol is detected (see below). Since format controls are all on the input tape, the Flexowriter, rather than the Photoreader, should be used for second-pass input.

Tape format. Instructions which do not have a symbolic location are input in two words: (a) the command and (b) the address, each of which is followed by a stop code (making 2 stop codes per line of coding). Instructions which have a symbolic location are input in four words: (a) an initial zero-word, (b) the symbolic location, (c) the command, and (d) the address, each of which is followed by a stop code (making 4 stop codes per line of coding). The initial zero-word will thus appear as a stop code; its only function is to indicate that the next word input will be a symbolic location.

Format controls (tabs, carriage returns, etc.) should all be punched on the tape. These are arbitrary, save for the following: if a comment follows the instruction one tab must follow this comment (to clear accumulator bits 26 - 31; only the tab will accomplish this, using 6-bit input on a standard Flexowriter). For the sample assembly above, tabs were included between the location operation address and comments; for the assembly of the program itself (see below) the first two of these were omitted.

Note on use of the Photoreader. The standard reader does not input 000000 (6-bit) on the execution of a tab, as required by the assembler if there are any comments included in the program. A minor modification of the input circuitry or exclusion of program comments is thus required for use of the Photoreader on the first assembling pass.

End-of-program symbol. The symbol "end," preceded by one and followed by two stop codes, must follow the last instruction of the program. This symbol must be used only for terminating an input; it should not be used as a program symbol.

Symbol specifications. Any numeric, alphabetic, or alphanumeric symbol of one to five characters in length is permissible. Certain symbols have special uses which are given below. The typewriter controls such as upper case, lower case carriage return etc. are not considered by the assembler as characters. The symbols "TEMP" and "temp" are indistinguishable as are "()" and "90," or "-" and "+."

Explanatory Notes (Ref. Sample Assembly)

1. Starting location and reference address. The reference address is the

"base number" of the addresses, i.e. the number from which the addresses (as distinguished from the locations) are numbered. In the present example, with reference address = 4000, the address "beta" = 4004. If the reference address were 0000, the address "beta" would be 0004. In either case, the program itself is stored starting at 4000.

2. Undefined symbols. The symbol "gamma" is "undefined" in the sense that it does not appear in the "location" column, and hence no numeric address can be associated with it. In such cases "/u" is printed following the symbol, and the easily recognized hex pattern "qqqqqqqq" is stored in the given memory location.

3. Multiply-defined symbols. The symbol "beta" is used to represent two different locations (lines 5 and 11 of the "p2" code). Each time a location symbol appears after its initial appearance, the characters "/m" are printed following the command of the given instruction. When the code is assembled on the second pass, the stored address corresponding to a multiply-defined symbol is the address assigned to it on its first appearance in the code. Thus "beta" = 4004 in line 1 of the "p2" printout, rather than 4010.

4. Instructions with absolute addresses may be input by prefixing an "x" to the operation, as for PIR input.

5. Hexadecimal constants may be input, as illustrated, by using "x" as the last character of the symbol designating the location of the constant. The eight-character word must be split into two four-character words for input, since they are input as 6-bit characters and converted to 4-bit by LGPSAP. Leading zeros are not required for either half of the word. Thus:

,3089ql72 is input from tape as 3089'ql72'

,0009ql72 can be input as 9'ql72'

,00000002 can be input as '2'

,40000000 can be input as 4000' '.

6. Instruction-form constants may be input, as for PIR, in the form xz'AAAA'.

7. Temporary storage locations, counters, parameters, etc. may be symbolically specified as illustrated; they will then be set to zero during assembly. Symbols ending in "x" may be used here.

OPERATION OF THE ASSEMBLER

For operation the following sequence of steps should be executed:

1. Load input tape in Flexowriter (or Photoreader).

2. Using 4-bit input, type manually .000AAAA, where AAAA is the start fill location of the assembly subroutine LS. Depress START COMP lever on Flexowriter.

3. Depress 6-bit input button.

4. Lift MAN INPUT on Flexowriter (if using reader, switch to reader input instead).

5. Depress START COMP lever again. The Flexowriter will print "p1" and the starting and reference addresses will be input. After about 2 minutes, the rest

of the tape will be input for the first pass. Any multiply-defined symbol error checks will be printed out as noted. Possible error stops may occur (see below). When the symbol "end" is reached, the Flexowriter prints out "p2" and the program stops, indicating that the second pass may be started.

6. Restart the tape in the Flexowriter to input the first word of the program proper (immediately following the reference address). Depress START COMP.

7. After the second pass is completed, a new tape may be loaded and the assembler restarted by simply depressing START COMP.

ASSEMBLY OUTSIDE THE "ALLOWABLE RANGE"

The assembler is self-protecting, and will input a program only into that portion of memory separating the assembler itself from the symbol tables, and not occupied by either. The symbol tables occupy the last 16 tracks in memory permitting the use of up to 512 different symbols in a single program. The output routine is fixed at 0300. If LS is loaded at 0700, the maximum range of locations 1232 to 4763 will be available for program assembly.

If it is desired to locate a program out of this range, for example starting at 5000, this may be done either by the use of K2-71 (see below) or as follows: Assemble the program within the allowable range, specifying 5000 as the reference address. Using program 13.2, output a hexadecimal tape of the stored program and change its "v" load instruction to load the tape at 5000. This illustrates the purpose of the reference address function of the assembler.

Preparation of Repositionable Punched Tape

Repositionable decimal punched tapes are readily prepared for the assembled programs by use of the POOL Program K2-71. Note that if the recommended initial locations of 0300 for the D-3 tape and 0700 for the LS tape are used, the program for punching must be entered after the assembly has been completed. By this technique it is possible always to use a standard initial location for the assembled program and to reposition it as desired later. The "modifier" used will be the same as the "reference address" of the assembly routine.

ERROR STOPS

During the first pass, the Flexowriter types out indications of the following errors:

Flexowriter Output	Error	Remedy
"ob"	Assembly is "out of bounds"; that is, not within allowable range given above	Relocate assembly and restart
"st"	Capacity of symbol table has been exceeded (should occur only rarely; possible only with large programs)	Easiest solution is to divide program into two or more subroutines and assemble separately

SYMBOL TABLE:

Tracks 48-63 inclusive are used for the symbol table and the symbol address table. Time is saved in storing symbols in the table in this program by the use of a random number method for the determination of the final locations of the symbols. The last nine bits of the six-bit pattern for the symbol are treated as a number from which a random address within the table is generated. If this address is not occupied it becomes the location for the symbol. If it is already occupied, the next address is then tested. Uniqueness is assured during the second pass by a comparison of the symbol sought with the symbol already stored. Further details are given in the Appendix.

LGPSAP SUBROUTINES

LGPSAP consists of two subroutines: (1) the assembler proper, designated LS, and (2) the instruction printout subroutine, designated D3. It also uses PIR 10.4 (see below). Subroutine D3 is a modified version of the decimal memory printout routine (21.0) and hence prints out hex words in fractional or hexadecimal form according to the setting of the TRANSFER CONTROL button. A list of the changes required in program 21.0 to obtain D3 is included below. Following this is the assembly of LS in LGPSAP code and decimal code relative to 0000.

The subroutine assembly is as follows:

Subroutine	Load At	Memory Space Required
D3	0300	0300 - 0663
LS	Relocatable, but maximum space is available for program assembly when loaded at 0700	Five and a half tracks, plus locations 4800 - 6363 for symbol and symbol-address tables
PIR 10.4	0000	0000 - 0263 Subroutine LS uses the binarize subroutine of PIR 10.4. (Note: other versions of PIR may not be used here).

TIME

On the first pass, the program will input about 17.5 instructions per minute, and on the second pass, about 9.5 instructions per minute, using the format illustrated for the assembly of LS (see below). These rates would be somewhat slower if the symbolic location, operation and address were separated by tabs, or if more comments were inserted (conversely, a program without any comments could probably be input at the rate of 11 or 12 instructions per minute on the second pass).

These figures are relatively unaffected by the length of the program. More specifically, an increase in the number of unique symbols used in the program will not increase the symbol-table "lookup" time for any given symbol, until the number of unique symbols begins to approach the maximum allowable 512. (The programming logic for accomplishing this was suggested to the author by Mr. George Feeney of the General Electric Company).

The present version of LGPSAP is unoptimized; a further reduction of input time could doubtless be realized by full or even partial optimization of the code.

APPENDIX: The Random Address Generator

The operation is given in the flow chart in the first box following variable connector branch V1A, as follows:

$$\text{Bits } 1 - 9 \ (\text{WD} * \text{MPLR} \bmod 2^{30}) \rightarrow \text{R},$$

where the symbol is to be stored in the Rth line of the table. The LGP-30 instructions corresponding to this (ref. the LGPSAP symbolic code) are:

Loc.	Op.	Address	Comments
V1A	B	WD	
	N	MPLRX	(WD * MPLR) mod 2^{30} at 29
	M	1A21	Bits S - 8→Bits 21 - 29
	E	NMXM1	Extract Bits 21 - 29
LS15	H	R	and store

The locations referred to contain the following:

WD		the current symbol	
1A21	XZ	0400	1 at 21
NMXM1	XZ	0763	mask
MPLRX	,5K2	1KGF	5^{11} at 30

In other words, the equation for the "random number" function could be expressed as

$$R = \text{Integer} \left\{ \left[(S \times 5^{11}) \bmod 2^{30} \right] \times 2^{-21} \right\}$$

Where R is as defined above and S is the value obtained by regarding the symbol as a number at q = 30; R ranges from 0 to 511. The incorporation of this search-saver into the routine reduced average assembly time by about two-thirds.

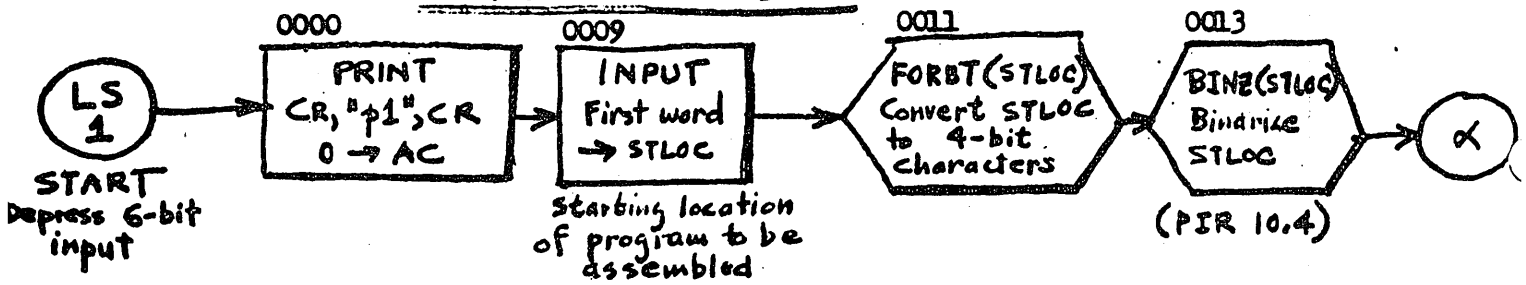
TABLE OF SYMBOLS FOR THE LGPSAP FLOW CHART

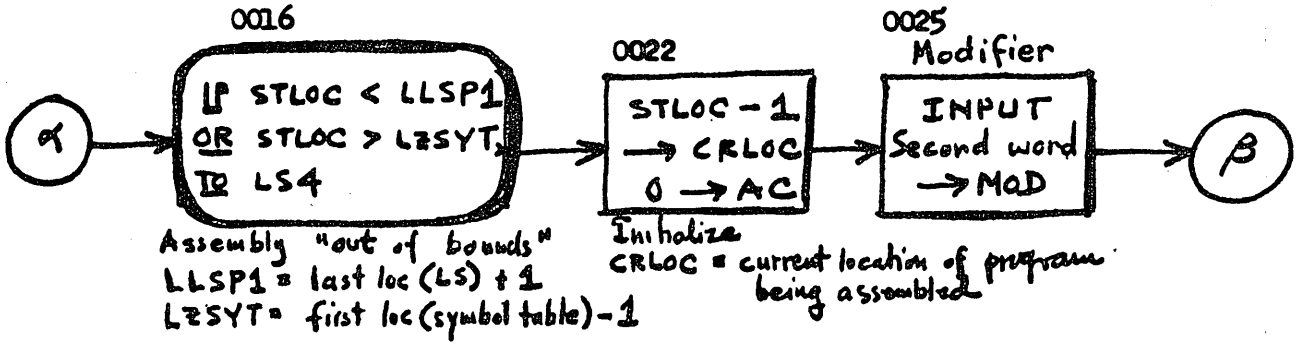
BINZ	Binarize
CR	Carriage Return
CRLOC	Current location of program being assembled
FORBT	Convert from 6-bit to 4-bit characters
LGSYT	Address of a line in the symbol table
LLSPI	Last location +1
LZSYA	First location of the symbol-address table
LZSYA(R)	Rth line of symbol-address table
LZSYT	First location of the symbol table - 1
LZSYT(R)	Rth line of symbol table
MOD	Modifier
MPLR	Multiplier for computing random address ($=5^{11}$)
NEGWD	Negative word (=wwwwwwwq)
NMXMI	Number of lines in the symbol table - 1
R	Random address modifier for symbol table
S	Index for lines of symbol table
STLOC	Starting location of program to be assembled
SYCTR	Symbol address counter
WD	A temporary storage location
WKLOC	A temporary storage location

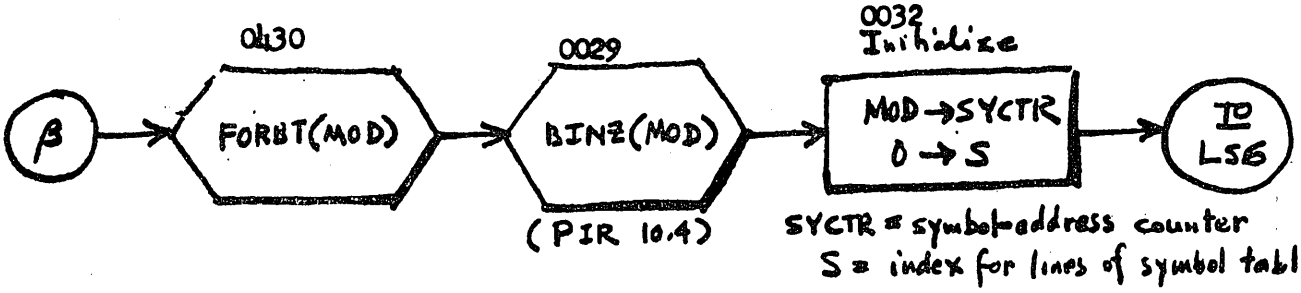
Table of Symbols Used on the Coding Sheet Notes

(	Less than
)	Greater than
*	Multiplied by
e	Exponent

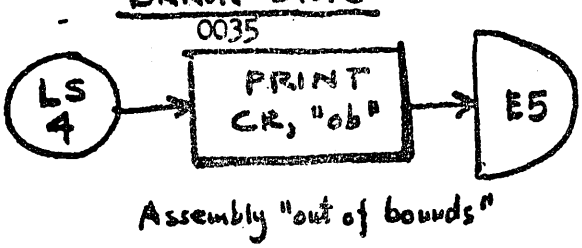
Flow Chart

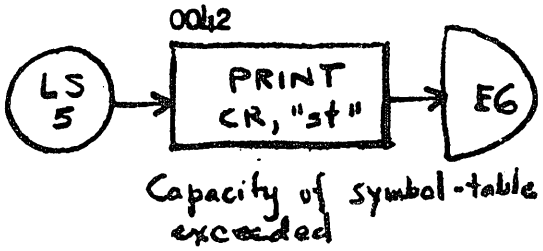


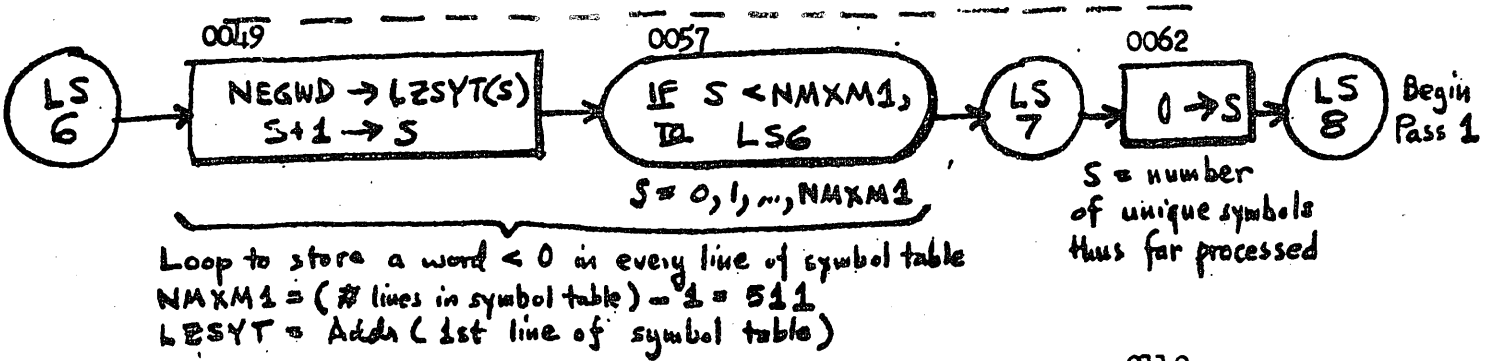


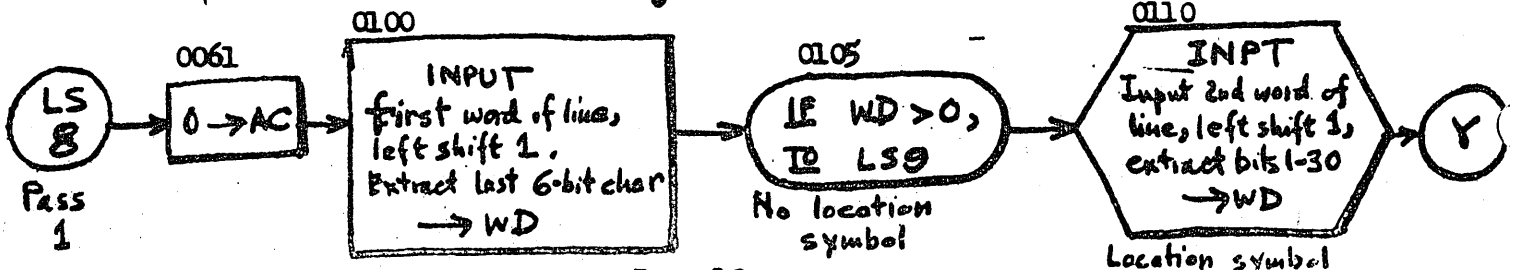


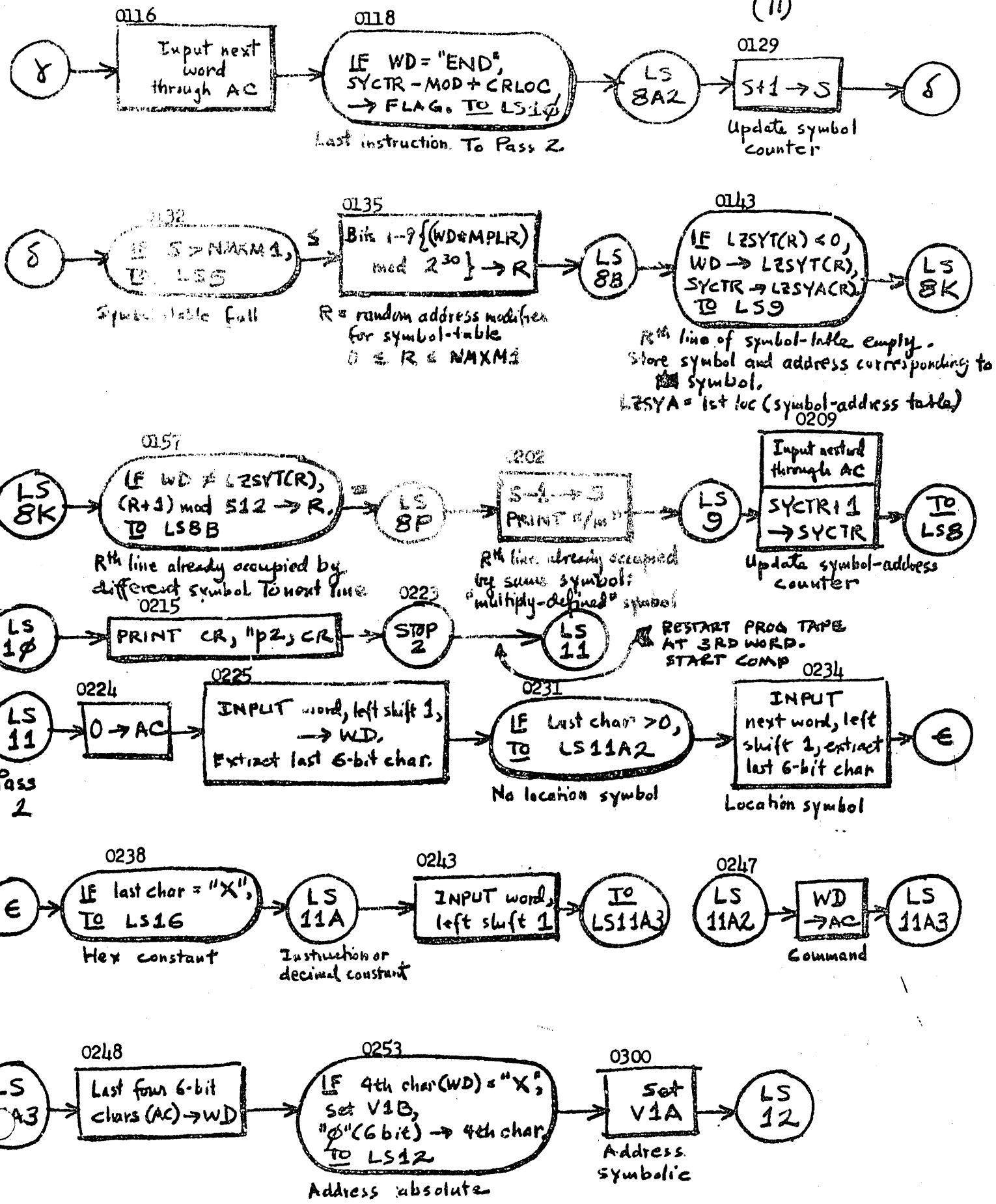
ERROR STOPS

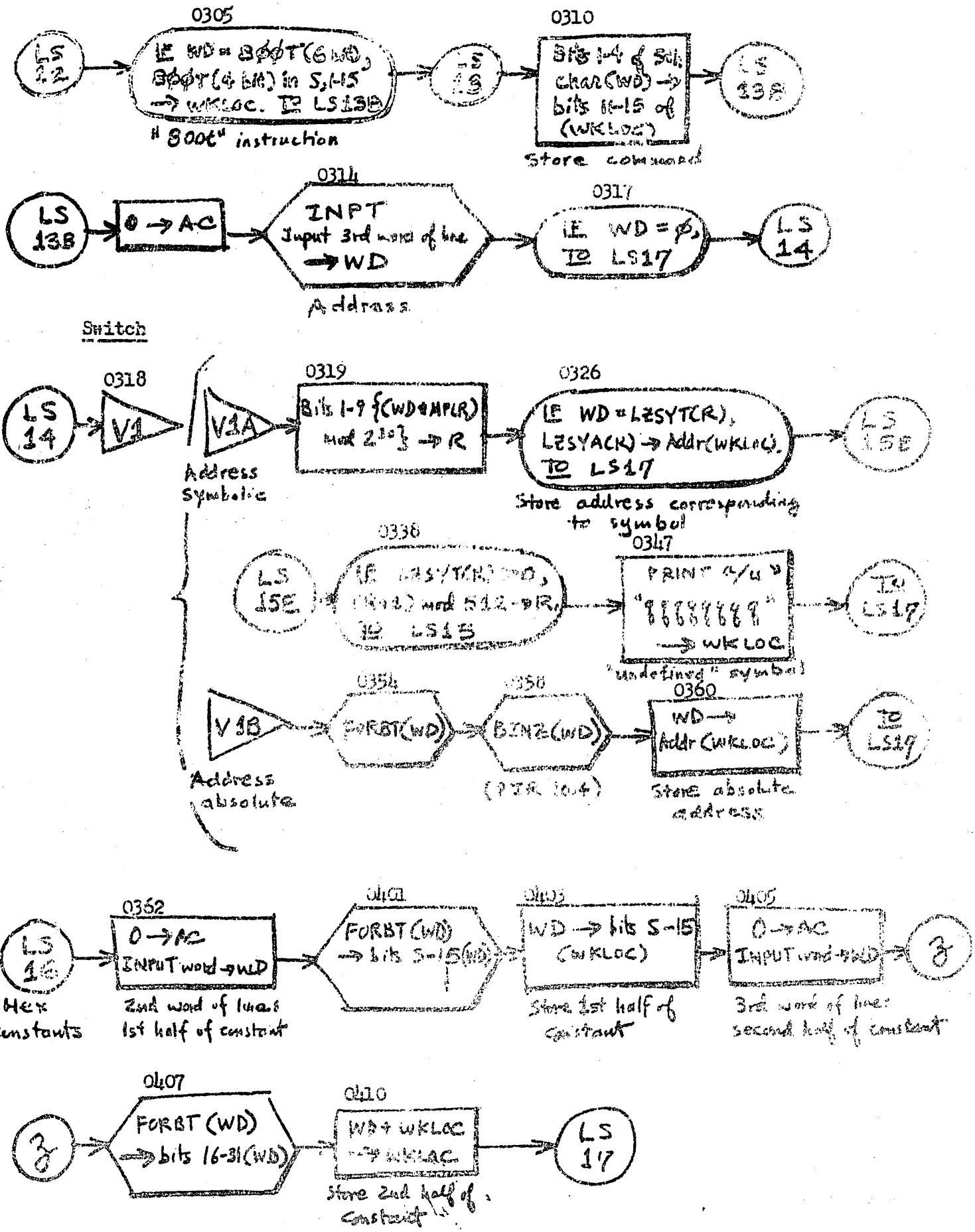


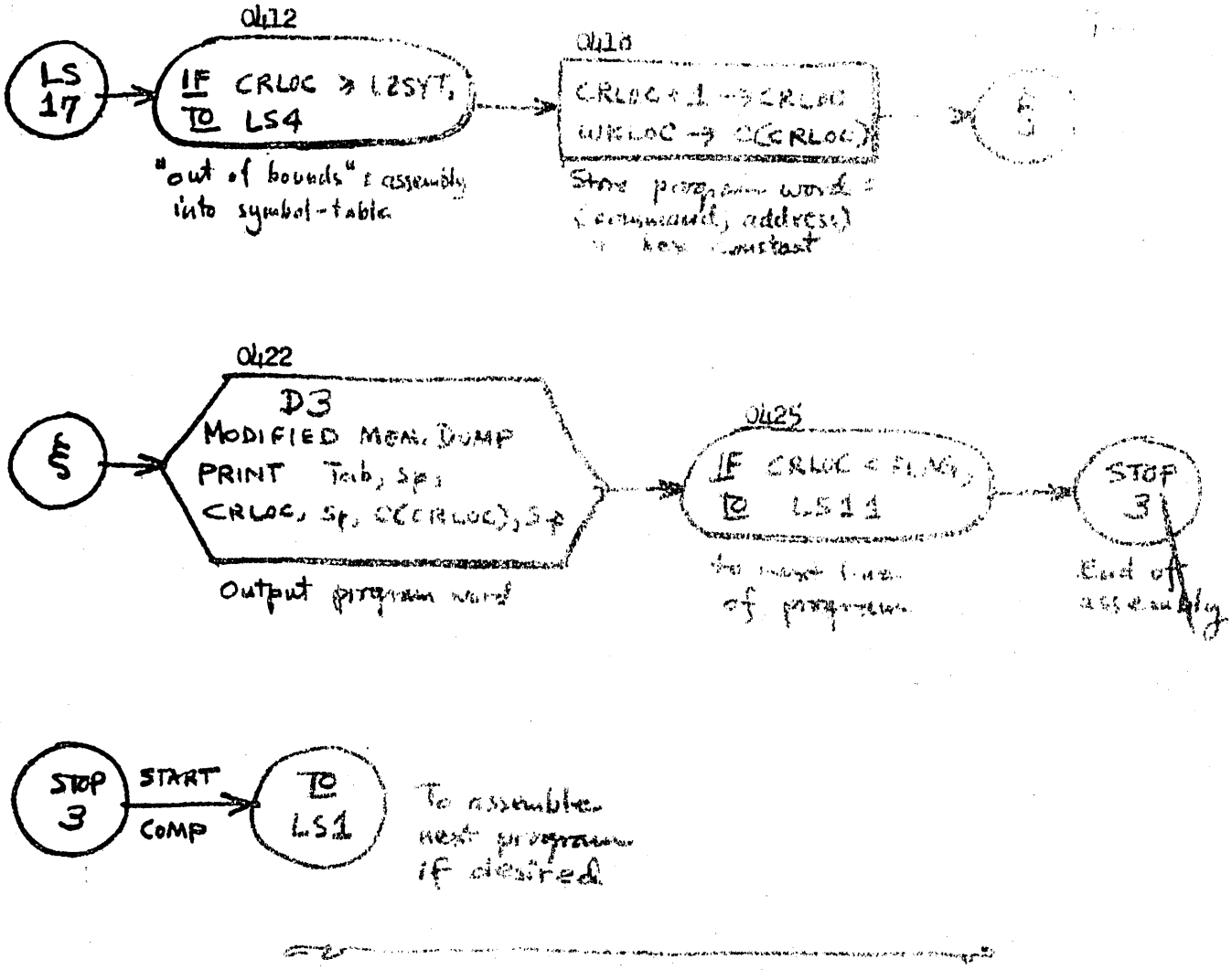







γ
0116
Input next word through AC
0118
IF WD = "END", SYCTR - MOD + CRLOC → FLAG. TO LS1Ø
Last instruction To Pass 2.
LS 8A2
0129
S+1 → S
Update symbol counter
δ
δ
0132
IF S > NMXM1, TO LS5
Symbol-table full
S
0135
Bits 1-9 {(WD*MPLR) mod 2^30} → R
R = random address modifier for symbol-table
0 ≤ R ≤ NMXM1
LS 8B
0143
IF LZSYT(R) < 0, WD → LZSYT(R), SYCTR → LZSYA(R), TO LS9
Rth line of symbol-table empty. Store symbol and address corresponding to symbol.
LZSYA = 1st loc (symbol-address table)
LS 8K
LS 8K
0157
IF WD ≠ LZSYT(R), (R+1) mod 512 → R, TO LS8B
Rth line already occupied by different symbol. To next line.
LS 8P
0202
S-1 → S
PRINT "/M"
Rth line already occupied by same symbol: "multiply-defined" symbol
LS 9
0209
Input nextwd through AC
SYCTR+1 → SYCTR
Update symbol-address counter
TO LS8
LS 1Ø
0215
PRINT CR, "P2", CR
STOP 2
0223
LS 11
RESTART PROG TAPE AT 3RD WORD. START COMP
LS 11
Pass 2
0224
0 → AC
0225
INPUT word, left shift 1, → WD. Extract last 6-bit char.
0231
IF Last char > 0, TO LS11A2
No location symbol
0234
INPUT next word, left shift 1, extract last 6-bit char
Location symbol
ε
ε
0238
IF last char = "X", TO LS16
Hex constant
LS 11A
Instruction or decimal constant
0243
INPUT word, left shift 1
TO LS11A3
0247
LS 11A2
WD → AC
Command
LS 11A3
LS A3
0248
Last four 6-bit chars (AC) → WD
0253
IF 4th char(WD) = "X", set V1B, "Ø"(6 bit) → 4th char, TO LS12
Address absolute
0300
Set V1A
Address symbolic
LS 12


0305
LS 12
IF WD = BØØT(6 bit), BØØT(4 bit) in S,1-15 → WKLOC. TO LS13B
"BOOT" instruction
LS 13
0310
Bits 1-4 of 5th char(WD) → bits 11-15 of (WKLOC)
Store command
LS 13B
LS 13B
0 → AC
0314
INPT
Input 3rd word of line → WD
Address
0317
IF WD = Ø, TO LS17
LS 14
Switch
LS 14
0318
V1
V1A
Address Symbolic
0319
Bits 1-9 {(WD•MPLR) mod 2^10} → R
0326
IF WD = LZSYTCR), LZSYACK) → Addr(WKLOC), TO LS17
Store address corresponding to symbol
LS 15E
LS 15E
0338
IF LZSYTCR) = 0, (R+1) mod 512 → R, TO LS15
0347
PRINT "U"
"ØØØØØØØØ" → WKLOC
"undefined" symbol
TO LS17
V1B
0354
FORBT(WD)
0350
BINZ(WD)
(PTR 104)
0360
WD → Addr(WKLOC)
Store absolute address
TO LS17
Address absolute
LS 16
Hex Constants
0362
0 → AC
INPUT word → WD
2nd word of line: 1st half of constant
0401
FORBT(WD) → bits 5-15(WD)
0403
WD → bits 5-15 (WKLOC)
Store 1st half of constant
0405
0 → AC
INPUT word → WD
3rd word of line: second half of constant
3
3
0407
FORBT(WD) → bits 16-31(WD)
0410
WD + WKLOC → WKLOC
Store 2nd half of constant
LS 17


0412
LS 17
IF CRLOC > L2SYT,
TO LS4
"out of bounds" assembly into symbol-table
0416
CRLOC + 1 → CRLOC
WRLOC → C(CRLOC)
Store program word (command, address) in hex constant
§
0422
§
D3
MODIFIED MEM. DUMP
PRINT Tab, Sp,
CRLOC, Sp, C(CRLOC), S↑
Output program word
0425
IF CRLOC < FLMAX
TO LS11
to next line of program
STOP 3
End of assembly
STOP 3
START
COMP
TO LS1
To assemble next program if desired

Coding for Assembly Routine (LGPSAP and Decimal)

(Application of LGPSAP Assembler to a tape containing the LGPSAP program punched in LGPSAP-type coding)

```
18
assembly relative to 0000, loaded at 3000

'isl'xp'1600'      3000  p1600      first pass.  c.r.
xx'0000'           3001  x0000
xp'3500'           3002  p3500      p
xx'0000'           3003  x0000
xp'0600'           3004  p0600      1
xx'0000'           3005  x0000
xp'1600'           3006  p1600      c.r.
xx'0000'           3007  x0000
c'dmp'             3008  c0519      ac = 0
xp'0000'           3009  p0000
xi'0000'           3010  i0000      input starting loc
r'frxt'            3011  r0451
u'forbt'           3012  u0430      stloc = forbt(stloc)
xr'0063'           3013  x0063
xu'0051'           3014  u0051      binarize rtn (PIR 10.4)
h'stloc'           3015  h0514      stloc = binz(stloc)
s'llspl'           3016  s0510
t'ls4'             3017  t0035      if stloc ( llspl
b'stloc'           3018  b0514
s'lssyt'           3019  s0511
t'ls2'             3020  t0022
u'ls4'             3021  u0035      or stloc  ge  lssyt
'ls2'b'stloc'      3022  b0514
s'ls29'            3023  s0503
c'crloc'           3024  c0515      crloc = stloc - 1, ac = 0
xp'0000'           3025  p0000
xi'0000'           3026  i0000      input modifier
r'frxt'            3027  r0451
u'forbt'           3028  u0430      forbt(mod)
xr'0063'           3029  x0063
xu'0051'           3030  u0051
h'mod'             3031  h0517      binz(mod)
c'syctr'           3032  c0518      syctr = mod
c's'               3033  c0524      s = 0
u'ls6'             3034  u0049
'ls4'xp'1600'      3035  p1600      out of boundr.  cr
xx'0000'           3036  x0000
xp'3500'           3037  p3500      o
xx'0000'           3038  x0000
xp'0500'           3039  p0500      b
xx'0000'           3040  x0000
xx'0005'           3041  x0005      error stop 5
'ls5'xp'1600'      3042  p1600      symbol table full.  cr
xx'0000'           3043  x0000
xp'6100'           3044  p6100      s
xx'0000'           3045  x0000
```

(An application of LGPSAP Assembler to a tape containing the LGPSAP program punched in LGPSAP-type coding)

```
xp'4300'
xa'0000'
xm'0006'
'lc6'b'lzsyt'
a'0'
y'lc6a'
b'negwd'
'lc6a'xh'6363'
b'8'
a'lc89'
h'0'
b'mxmal'
a'8'
t'lc7'
u'lc6'
'lc7'c'dump'
c'8'
'lc8'c'dump'
xp'0000'
xa'0000'
n'lc90x'
c'mlx'
c'wd'
s'wd'
t'lc9'
b'clear'
y'input'
'input'c'dump'
xp'0000'
xa'0000'
n'lc80x'
c'mlx'
h'wd'
'input'xm'0363'
'lc8al'xp'0000'
xa'0000'
b'wd'
s'comix'
t'lc8a2'
s'lc90x'
t'lc8a'
u'lc8a2'
'lc8a'b'syctr'
s'mod'
a'erloc'
h'flag'
u'lc10'
'lc8a2'b'8'
a'lc89'
h'8'
b'mxmal'
s'8'
t'lc5'
```

```
3046 4300    t
3047 0000
3048 0006    error step 6
3049 lc9z1
3050 c0z4
3051 y0025
3052 b0z26
3053 h0z63
3054 b0z4
3055 c0z05
3056 h0z4
3057 b0z15
3058 c0z4
3059 t0061
3060 u0z9    if a lo mxmal
3061 c0z19
3062 c0z4    a = 0
3063 c0z9
3100 p0000
3101 10000   lst wd of line
3102 m0z02
3103 c0z55   lost 6b char
3104 c0z20
3105 c0z20
3106 s0z09   if wd gr 0
3107 b0z22
3108 y0015
3109 c0z19   cc = 0
3110 p0000
3111 10000   input location symbol
3112 m0z02
3113 c0z55
3114 h0z20   ... wd at 30
3115 c4363
3116 p0000
3117 10000   next wd through cc
3118 b0z20
3119 c0z4
3120 t0019
3121 c0z12
3122 t0015   if wd = "end",
3123 c0z19
3124 b0z18
3125 c0z17
3126 c0z19   syctr = mod + erloc
3127 h0z14   ... flag
3128 u0z15
3129 b0z4
3130 c0z03
3131 h0z4    a = a + 1
3132 b0z15
3133 c0z4
3134 t0042   if a gr mxmal
```

Coding for Assembly Routine (LGPSAP and Decimal)

b'wd'	3135	b0520	
n'mplrx'	3136	n0522	(wd*mplr)mod 2e30 at 29
m'1e21'	3137	m0509	bits 0-8 to 21-29
e'nmxml'	3138	e0513	21-29
'1e8b'h'r'	3139	h0523	... r random addr modifier
a'lzsyt'	3140	a0511	
y'1e8c'	3141	y0143	
y'1e8e'	3142	y0147	
'1e8c'xb'6363'	3143	b6363	/lzsyt + r/
t'1e8d'	3144	t0146	if lzsyt.r ls 0,
u'1e8k'	3145	u0154	
'1e8d'b'wd'	3146	b0520	
'1e8e'xh'6363'	3147	h6363	lzsyt.r = wd,
b'lzsya'	3148	b0512	
a'r'	3149	a0523	
y'1e8f'	3150	y0152	
b'syctr'	3151	b0518	
'1e8f'xh'6363'	3152	h6363	lzsya.r = syctr.
u'1e9'	3153	u0209	
'1e8k'b'lzsyt'	3154	b0511	
a'r'	3155	a0523	
y'1e8m'	3156	y0158	
b'wd'	3157	b0520	
'1e8m'xz'6363'	3158	z6363	/lzsyt + r/
t'1e8n'	3159	t0162	if wd ue lzsyt.r,
s'1e30x'	3160	s0502	
t'1e8p'	3161	t0202	
'1e8n'b'r'	3162	b0523	
a'1e29'	3163	a0503	
e'nmxml'	3200	e0513	r = (r + 1)mod nmax.
u'1e8b'	3201	u0139	
'1e8p'b's'	3202	b0524	multiply-defined sym
s'1e29'	3203	s0503	
h's'	3204	h0524	s = s - 1
xp'1900'	3205	p1900	/
xz'0000'	3206	z0000	
xp'2900'	3207	p2900	m
xz'0000'	3208	z0000	
'1e9'xp'0000'	3209	p0000	
xi'0000'	3210	i0000	next wd through ac
b'syctr'	3211	b0518	
a'1e29'	3212	a0503	
h'syctr'	3213	h0518	syctr = syctr + 1
u'1e8'	3214	u0063	
'1e10'xp'1600'	3215	p1600	second pass. cr
xz'0000'	3216	z0000	
xp'3300'	3217	p3300	p
xz'0000'	3218	z0000	
xp'1000'	3219	p1000	2
xz'0000'	3220	z0000	
xp'1600'	3221	p1600	cr
xz'0000'	3222	z0000	
xz'0002'	3223	z0002	stop 2. restart tape

```
'lall'e'[illegible]'
xp'0000'
xl'0000'
n'la3[illegible]'
h'wd'
o'[illegible]'
e'[illegible]'
s'[illegible]'
t'lalla2'
e'[illegible]'
xp'0000'
xl'0000'
n'la3[illegible]'
o'[illegible]'
s'[illegible]'
t'lalla'
s'la3[illegible]'
t'lal6'
'lalla'e'[illegible]'
xp'0000'
xl'0000'
n'la3[illegible]'
u'lalla3'
'lalla2'b'wd'
'lalla3'o'[illegible]'
h'wd'
[illegible]'la[illegible]'
o'[illegible]'
s'[illegible]'
t'lalla'
s'la3[illegible]'
t'lal1b'
u'lalla'
'lalla'b'evlb'
y'lal1'
b'wd'
o'[illegible]'
h'wd'
u'lal2'
'lalla'b'evla'
y'lal1'
'lal2'b'wd'
o'60[illegible]'
t'lal3'
s'la3[illegible]'
t'lal2a'
u'lal3'
'lal2a'b'60[illegible]'
h'[illegible]'
u'lal1b'
'lal3'b'wd'
n'lal8'
o'[illegible]'
h'[illegible]'
```

```
[illegible]
[illegible]
[illegible]
```

```
'lol3'r'input'  3590  c0115
e'input'        3591  w0109
n'lol0='        3592  c0022
t'lol7'         3593  w0122
'lol4'u'lol4'   3594  w0328
'vlo'b'wd'      3595  b0220
n'mplus'        3596  s0222
□'lo21'         3597  □0209
o'mmmul'        3598  c0215
'lol5'h'r'      3599  h0235
a'lmyt'         3600  c0231
y'lol3a'        3601  p0215
'lol3a'xb'6365' 3602  b0263
a'wd'           3603  c0220
t'lol3o'        3604  c0223
a'lo30x'        3605  c0232
t'lol3b'        3606  c0242
u'lol3c'        3607  c0238
'lol3b'b'r'     3608  b0239
a'lmya'         3609  c0228
y'lol3e'        3610  p0225
'lol3e'xb'6365' 3611  b0263
y'wlee'         3612  y0226
u'lol7'         3613  w0242
'lol3o'b'r'     3614  b0235
a'lmyt'         3615  c0231
y'lol3f'        3616  p0215
'lol3f'xb'6365' 3617  b0263
t'lol33'        3618  t0247
b'r'            3619  b0239
a'lo29'         3620  c0203
o'mmmul'        3621  c0215
u'lol5'         3622  c0209
'lol33'xp'1900' 3623  p0200
xa'0000'        3624  c0000
xp'4200'        3625  p4200
xa'0000'        3626  c0000
b'cmmm'         3627  b0217
h'wlee'         3628  h0226
u'lol7'         3629  c0242
'vlb'b'wd'      3630  b0220
□'lolr'         3631  □0201
r'fmmt'         3632  r0231
u'fcnbt'        3633  w0230
xa'0035'        3634  x0053
xa'0031'        3635  x0049
y'wlee'         3636  y0226
u'lol7'         3637  w0242
'lol6'o'dump'   3638  c0239
xp'0000'        3639  p0000
xa'0000'        3640  x0000
r'fmmt'         3641  r0231
u'fcnbt'        3642  w0230
```

```
lmyat address symbol

if = 0
ve 1


bito 1-9[vc'mplr]mod 2o30
... r


/lmyt + r/


if wl = lmyt.r,


/lmya + r/
addr(wlee) = lmya.r


/lmyt + r/

if lmyt.r  gr  0

r = (r + 1)mod mmm

undefined symbol.  /

□

wlee = common


fcnbt(wd)

blnk(wd)   (PAR 10.4)
... addr(wlee)

hex constant   cc = 0

input first half

to 4-bit
```

```
n'lal7x'          3403  n0504     at 15
c'wkloc'          3404  c0516     ... wkloc
xp'0000'          3405  p0000
xd'0000'          3406  10000     input second half
r'rmt'            3407  r0451
u'fcrbt'          3408  u0430     to 4-bit
m'lalx'           3409  m0505     at 31
a'wkloc'          3410  a0516     + wkloc
h'wkloc'          3411  h0516     ... wkloc
'lal7'b'crloc'    3412  b0515
a'lm29'           3413  a0503
h'crloc'          3414  h0515     crloc = crloc + 1
s'lmyt'           3415  s0511
t'lal7a'          3416  t0418
u'lsh'            3417  u0035     if crloc  ge  lmyt
'lal7a'b'crloc'   3418  b0515
y'lal7b'          3419  y0421
b'wkloc'          3420  b0516
'lal7b'xh'6363'   3421  h6363     c(wkloc) ... crloc
b'crloc'          3422  b0515
xr'0600'          3423  r0600     DB.  print tab, sp,
xm'0303'          3424  m0303     crloc, c(crloc), sp
b'crloc'          3425  b0515
s'flag'           3426  s0521
t'lsll'           3427  t0224     if crloc  is  flag
xz'0003'          3428  z0003     prog stop
u'lal'            3429  u0000     start comp for next input
'forbt'c'6bwd'    3430  c0528     4-bit conv rtn.   word at 31
c'4bwd'           3431  c0529     4bwd = 0
b'mlmlx'          3432  b0506
h'dump'           3433  h0319     -4 into counter
b'cl5'            3434  b0507
h'mask'           3435  h0330     initialize mask  bits 26-9
'frl'b'6bwd'      3436  b0528
e'mask'           3437  e0330     kth 4-bit char  k = 4,3,2,1
a'4bwd'           3438  a0529
h'4bwd'           3439  h0329
b'6bwd'           3440  b0528
m'lalx'           3441  m0505     position next 4-bit char
h'6bwd'           3442  h0328
b'mask'           3443  b0330
n'c4'             3444  n0508     left 4. set mask for next char
h'mask'           3445  h0330
b'dump'           3446  b0319
a'lal7x'          3447  a0502
h'dump'           3448  h0319     increase ctr by 1
t'frl'            3449  t0436     if ctr negative
b'4bwd'           3450  b0529
'f'wa'ist'(3bd)'  3451  u6363     exit
```

Coding for Assembly Routine (LGPSAP and Decimal)

'c18a1''1s8a1'	3452	z0116	
'm1x'7www'wwwq'	3453	,7wwwwwwq	
'cendx'4'fjff'	3454	,0004fjff	
'm2x''7q'	3455	,0000007q	
'ngwdx'wwww'wwwq'	3456	,wwwwwwwq	
'charx''4q'	3457	,0000004q	
'm3x'1ww'wwwq'	3458	,01wwwwwq	
'la6x'200'0000'	3459	,00000000	
'm4x'1ww'q17q'	3460	,01wwq17q	
'80t6x'110'415f'	3461	,0110415f	
'80t4x'800g'0000'	3462	,800g0000	
'la18'xz'3200'	3465	z3200	
'm5x'w'0000'	3500	z0000	
'la1x'4000'0000'	3501	,40000000	
'la30x''2'	3502	,00000002	
'la29'xz'0001'	3503	z0001	
'la17x''4000'	3504	,00004000	
'la2x'2000'0000'	3505	,20000000	
'min4x'wwww'www8'	3506	,wwwwwww8	minus 4 at 30
'c15'xz'0015'	3507	z0015	
'c4'xz'0004'	3508	z0004	
'la21'xz'0400'	3509	z0400	
'llspl''finwd'	3510	z0531	final loc plus 1
'lzsyt'xz'4800'	3511	z4800	
'lzsya'xz'5600'	3512	z5600	
'nmxml'xz'0763'	3513	z0763	symtbl mxmol minus 1
'stloc'''	3514	z0000	
'crloc'''	3515	z0000	
'wkloc'''	3516	z0000	
'mod'''	3517	z0000	
'syctr'''	3518	z0000	
'dmp'''	3519	z0000	
'wd'''	3520	z0000	
'flag'''	3521	z0000	
'mplrx'5k2'1kgf'	3522	,05k21kgf	5 exp 11
'r'''	3523	z0000	
's'''	3524	z0000	
'cvla''vla'	3525	z0319	
'cvlb''vlb'	3526	z0354	
'cqqqx'qqqq'qqqq'	3527	,qqqqqqqq	
'6swd'''	3528	z0000	
'4swd'''	3529	z0000	
'mask'''	3530	z0000	
'finwd'''	3531	z0000	

Program No. [illegible]

SYMBOLIC ASSEMBLY PROGRAM FOR THE LGP-30 COMPUTER

NOTE ON SUBROUTINE D3

The following changes must be made in the Decimal Memory Printout Routine (#21.0) to obtain D3, the LGPSAP output routine:

Location	Change to
0002	u0019
0003	y0108
0004	xc0143
0005	y0225
0006	n0204
0007	u0000
0019	xp2438
0020	u0359
0026	xp0305
0149	xp0300
0300	xu6363

PREPARED FOR: LGP-30 USERS' ORGANIZATION - POOL | PAGE 1 OF 11

JOB NO. | PROGRAM NO. H2-120 | PROGRAM PREPARED BY: James N. Orton | PROGRAM CHECKED BY: POOL Review | DATE 2/5/60

PROBLEM: SYMBOLIC ASSEMBLY PROGRAM for the LGP-30 COMPUTER- LGPSAP | TRACK

PROGRAM INPUT CODES	STOP	LOCATION	INSTRUCTION OPERATION	INSTRUCTION ADDRESS	STOP	CONTENTS OF ADDRESS	NOTES
	/						
	/ ☒						
		00	x p	1600	/		first pass c. r.
		01	x z	0000	/		
		02	x p	3300	/		p
		03	x z	0000	/	☒	
		04	x p	0600	/		1
		05	x z	0000	/		
		06	x p	1600	/		c.r.
		07	x z	0000	/	☒	
		08	c	0519	/		ac=0
		09	x p	0000	/		
		10	x i	0000	/		input starting loc
		11	r	0451	/	☒	
		12	u	0430	/		stloc = forbt (stloc)
		13	x r	0063	/		
		14	x u	0051	/		binarize (PIR 10.4)
		15	h	0514	/	☒	stloc = binz (stloc)
		16	s	0510	/		
		17	t	0035	/		if stloc (llspl
		18	b	0514	/		
		19	s	0511	/	☒	
		20	t	0022	/		
		21	u	0035	/		or stloc) lzsyt
		22	b	0514	/		
		23	s	0503	/	☒	
		24	c	0515	/		crloc = stloc + 1.ac=0
		25	x p	0000	/		
		26	x i	0000	/		input modifier
		27	r	0451	/	☒	
		28	u	0430	/		forbt (mod)
		29	x r	0063	/		
		30	x u	0051	/		
		31	h	0517	/	☒	binz(mod)

PREPARED FOR:				PAGE 2 OF 11
LGP-30 USERS' ORGANIZATION - POOL				
JOB NO.	PROGRAM NO. H2-120	PROGRAM PREPARED BY: James N. Orton	PROGRAM CHECKED BY: POOL Review	DATE 2/5/60
PROBLEM: SYMBOLIC ASSEMBLY PROGRAM for the LGP-30 COMPUTER - LGPSAP				TRACK

PROGRAM INPUT CODES	STOP	LOCATION	INSTRUCTION OPERATION	INSTRUCTION ADDRESS	STOP	CONTENTS OF ADDRESS	NOTES
	/						
	/	☒					
		13 12	c	0518	/		syctr = mod
		13 13	c	0524	/		s = 0
		13 14	u	0049	/		
		13 15	xp	1600	/	☒	out of bounds. cr
		13 16	xz	0000	/		
		13 17	xp	3500	/		o
		13 18	xz	0000	/		
		13 19	xp	0500	/	☒	b
		14 10	xz	0000	/		
		14 11	xz	0005	/		error stop 5
		14 12	xp	1600	/		symbol table full cr
		14 13	xz	0000	/	☒	
		14 14	xp	6100	/		s
		14 15	xz	0000	/		
		14 16	xp	4500	/		t
		14 17	xz	0000	/	☒	
		14 18	xz	0006	/		error stop 6
		14 19	b	0511	/		
		15 10	a	0524	/		
		15 11	y	0053	/	☒	
		15 12	b	0456	/		
		15 13	xh	6363	/		lzsyt(s) = negwd
		15 14	b	0524	/		
		15 15	a	0503	/	☒	
		15 16	h	0524	/		s = s + 1
		15 17	b	0513	/		
		15 18	s	0524	/		
		15 19	t	0061	/	☒	
		16 10	u	0049	/		if s (nmxml
		16 11	c	0519	/		
		16 12	c	0524	/		s = 0
		16 13	c	0519	/	☒	

PREPARED FOR: LGP-30 USERS' ORGANIZATION - POOL			PAGE 3 OF 11
JOB NO.	PROGRAM NO. H2-120	PROGRAM PREPARED BY: James N. Orton	PROGRAM CHECKED BY: POOL Review	DATE 2/5/60
PROBLEM: SYMBOLIC ASSEMBLY PROGRAM for the LGP-30 COMPUTER - LGPSAP			TRACK

PROGRAM INPUT CODES	STOP	LOCATION	INSTRUCTION OPERATION	INSTRUCTION ADDRESS	STOP	CONTENTS OF ADDRESS	NOTES
	/						
	/	☒					
		0110 10	x	p 0 0 0 0	/		
		10 11	x	i 0 0 0 0	/		1st wd of line
		10 12	n	0 5 0 2	/		
		10 13	e	0 4 5 5	/	☒	last 6b char
		10 14	c	0 5 2 0	/		
		10 15	s	0 5 2 0	/		
		10 16	t	0 2 0 9	/		if wd) 0
		10 17	b	0 4 5 2	/	☒	
		10 18	y	0 1 1 5	/		
		10 19	c	0 5 1 9	/		ac = 0
		11 10	x	p 0 0 0 0	/		
		11 11	x	i 0 0 0 0	/	☒	input location symbol
		11 12	n	0 5 0 2	/		
		11 13	e	0 4 5 3	/		
		11 14	h	0 5 2 0	/		...wd at 30
		11 15	x	u 6 3 6 3	/	☒	
		11 16	x	p 0 0 0 0	/		
		11 17	x	i 0 0 0 0	/		next wd through ac
		11 18	b	0 5 2 0	/		
		11 19	s	0 4 5 4	/	☒	
		12 10	t	0 1 2 9	/		
		12 11	s	0 5 0 2	/		
		12 12	t	0 1 2 4	/		if wd = "end,"
		12 13	u	0 1 2 9	/	☒	
		12 14	b	0 5 1 8	/		
		12 15	s	0 5 1 7	/		
		12 16	a	q 5 1 5	/		~~syctr - mod + crloc~~
		12 17	h	0 5 2 1	/	☒	...flag
		12 18	u	0 2 1 5	/		
		12 19	b	0 5 2 4	/		
		13 10	a	0 5 0 3	/		
		13 11	h	0 5 2 4	/	☒	s = s + 1

PREPARED FOR: LGP-30 USERS' ORGANIZATION - POOL			PAGE 4 OF 11
JOB NO.	PROGRAM NO. H2-120	PROGRAM PREPARED BY: James N. Orton	PROGRAM CHECKED BY: POOL Review	DATE 2/5/60
PROBLEM: SYMBOLIC ASSEMBLY PROGRAM for the LGP-30 COMPUTER - LGPSAP				TRACK

PROGRAM INPUT CODES	STOP	LOCATION	INSTRUCTION OPERATION	INSTRUCTION ADDRESS	STOP	CONTENTS OF ADDRESS	NOTES
	/						
	/	X					
		0132	b	0513	/		
		133	s	0524	/		
		134	t	0042	/		if s) nmxml
		135	b	0520	/	X	
		136	n	0522	/		(wd*mplr)mod 2e30 at 29
		137	m	0509	/		bits 0-8 to 21-29
		138	e	0513	/		21-29
		139	h	0523	/	X	...r random addr.mod-
		140	a	0511	/		ifier
		141	y	0114	/		
		142	y	0114	7	/	
		143	x	b 6363	/	X	/lzsyt + r/
		144	t	0114	6	/	if lzsyt(r) (ls 0,
		145	u	0154	/		
		146	b	0520	/		
		147	x	h 6363	/	X	lzsyt(r) = wd,
		148	b	0517	2	/	
		149	a	0523	/		
		150	y	0152	/		
		151	b	0518	/	X	
		152	x	h 6363	/		lzsya(r) = syctr.
		153	u	0209	/		
		154	b	0511	/		
		155	a	0523	/	X	
		156	y	0158	/		
		157	b	0520	/		
		158	x	s 6363	/		/lzsyt + r/
		159	t	0162	/	X	if wd / lzsyt(r)
		160	s	0502	/		
		161	t	0202	/		
		162	b	0523	/		
		163	a	0503	/	X	

FORM LP-12

PRINTED IN U.S.A.

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

25

X = CARRIAGE RETURN

/ = CONDITIONAL STOP CODE

LGP-30 CODING SHEET

PREPARED FOR: LGP-30 USERS' ORGANIZATION - POOL | PAGE 5 OF 11

JOB NO. | PROGRAM NO. H2-120 | PROGRAM PREPARED BY: James N. Orton | PROGRAM CHECKED BY: POOL Review | DATE 2/5/60

PROBLEM: SYMBOLIC ASSEMBLY PROGRAM for the LGP-30 COMPUTER - LGPSAP | TRACK

PROGRAM INPUT CODES	STOP	LOCATION	INSTRUCTION OPERATION	INSTRUCTION ADDRESS	STOP	CONTENTS OF ADDRESS	NOTES
	/						
	/	X					
		0210	e	0513	/	r =	(r+1)mod nmax.
		01	u	0139	/		
		02	b	0524	/		multiply-defined sym
		03	s	0503	/	X	
		04	h	0524	/		s = s - 1
		05	xp	1900	/		/
		06	xz	0000	/		
		07	xp	2900	/	X	m
		08	xz	0000	/		
		09	xp	0000	/		
		10	xi	0000	/		next wd through ac
		11	b	0518	/	X	
		12	a	0503	/		
		13	h	0518	/		syctr = syctr + 1
		14	u	0063	/		
		15	xp	1600	/	X	second pass. cr
		16	xz	0000	/		
		17	xp	3300	/		p
		18	xz	0000	/		
		19	xp	1000	/	X	2
		20	xz	0000	/		
		21	xp	1600	/		cr
		22	xz	0000	/		
		23	xz	0002	/	X	stop 2. restart tape
		24	c	0519	/		
		25	xp	0000	/		1st wd of line
		26	xi	0900	/		
		27	n	0502	/	X	
		28	h	0520	/		
		29	e	0455	/		last 6b char
		30	c	0519	/		
		31	s	0519	/	X	

FORM LP-10

PRINTED IN U.S.A.

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

X = CARRIAGE RETURN

/ = CONDITIONAL STOP CODE

PREPARED FOR: LGP-30 USERS' ORGANIZATION - POOL				PAGE 6 OF 11
JOB NO.	PROGRAM NO. H2-120	PROGRAM PREPARED BY: James N. Orton	PROGRAM CHECKED BY: POOL Review	DATE 2/5/60
PROBLEM: SYMBOLIC ASSEMBLY PROGRAM for the LGP-30 COMPUTER LGPSAP				TRACK

PROGRAM INPUT CODES	STOP	LOCATION	INSTRUCTION OPERATION	INSTRUCTION ADDRESS	STOP	CONTENTS OF ADDRESS	NOTES
	/						
	/	☒					
		0123 12	t	02417	/		if char) 0
		13 13	c	05119	/		ac = 0
		13 14	xp	01000	/		
		13 15	xi	01000	/	☒	input loc symbol
		13 16	n	05012	/		
		13 17	e	04515	/		last char
		13 18	s	04517	/		
		13 19	t	02412	/	☒	
		14 10	s	05012	/		
		14 11	t	03612	/		if last char = "x"
		14 12	c	05119	/		ac = 0
		14 13	xp	01000	/	☒	
		14 14	xi	01000	/		input instruction
		14 15	n	05012	/		
		14 16	u	02418	/		
		14 17	b	05210	/	☒	
		14 18	e.	04518	/		last 4 char
		14 19	h	05120	/		
		15 10	m	04519	/		
		15 11	e	04515	/	☒	
		15 12	s	04517	/		
		15 13	t	02613	/		
		15 14	s	05012	/		
		15 15	t	02515	/	☒	if 4th char = "x"
		15 16	u	02613	/		
		15 17	b	05216	/		
		15 18	y	03118	/		set v1b
		15 19	b	05210	/	☒	
		16 10	e	04610	/		
		16 11	h	05120	/		4th char = "0"
		16 12	u	03001	/		
		16 13	b	05215	/	☒	

PREPARED FOR: LGP-30 USERS' ORGANIZATION - POOL | PAGE 7 OF 11

JOB NO. | PROGRAM NO. H2-120 | PROGRAM PREPARED BY: James N. Orton | PROGRAM CHECKED BY: POOL Review | DATE 2/5/60

PROBLEM: SYMBOLIC ASSEMBLY PROGRAM for the LGP-30 COMPUTER LGPSAP | TRACK

PROGRAM INPUT CODES	STOP	LOCATION	INSTRUCTION OPERATION	INSTRUCTION ADDRESS	STOP	CONTENTS OF ADDRESS	NOTES
	/						
	/ ☒						
		0130	y	0318	/		set vla
		01	b	0520	/		
		02	s	0461	/		
		03	t	0310	/	☒	
		04	s	0502	/		
		05	t	0307	/		if wd = 800t(6-bit),
		06	u	0310	/		
		07	b	0462	/	☒	
		08	h	0516	/		wkloc = 800t(4-bit),
		09	u	0314	/		
		10	b	0520	/		store command
		11	n	0463	/	☒	
		12	e	0500	/		[,w0000]
		13	h	0516	/		
		14	r	0115	/		
		15	u	0109	/	☒	input address symbol
		16	s	0502	/		
		17	t	0412	/		if = 0
		18	u	0318	/		vc 1
		19	b	0520	/	☒	
		20	n	0522	/		
		21	m	0509	/		
		22	e	0513	/		bits 1-9 [wd*mplr] 2e30 mod
		23	h	0523	/	☒	...r
		24	a	0511	/		
		25	y	0326	/		
		26	x b	6363	/		/lzsyt + r/
		27	s	0520	/	☒	
		28	t	0338	/		
		29	s	0502	/		
		30	t	0332	/		if wd = lzsyt(r)
		31	u	0338	/	☒	

FORM LP-10

PRINTED IN U.S.A.

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

☒ CARRIAGE RETURN

/ = CONDITIONAL STOP CODE

LGP-30 CODING SHEET

PREPARED FOR: LGP - 30 USERS' ORGANIZATION - POOL				PAGE 8 OF 11
JOB NO.	PROGRAM NO. H2-120	PROGRAM PREPARED BY: James N. Orton	PROGRAM CHECKED BY: POOL Review	DATE 2/5/60
PROBLEM: SYMBOLIC ASSEMBLY PROGRAM for the LGP-30 COMPUTER LGPSAP				TRACK

PROGRAM INPUT CODES	STOP	LOCATION	INSTRUCTION OPERATION	INSTRUCTION ADDRESS	STOP	CONTENTS OF ADDRESS	NOTES
	/						
	/	☒					
		0332	b	0523	/		
		33	a	0512	/		
		34	y	0335	/		
		35	xb	6363	/	☒	/lzsya + r/
		36	y	0516	/		addr(wkloc)=lzsya(r)
		37	u	0412	/		
		38	b	0523	/		
		39	a	0511	/	☒	
		40	y	0341	/		
		41	xb	6363	/		/lzsyt + r/
		42	t	0347	/		
		43	b	0523	/	☒	if lzsyt(r))0
		44	a	0503	/		
		45	e	0513	/		r = (r+1)mod nmax
		46	u	0323	/		
		47	xp	1900	/	☒	undefined symbol. /
		48	xz	0000	/		
		49	xp	4100	/		u
		50	xz	0000	/		
		51	b	0527	/	☒	
		52	h	0516	/		wkloc = qqqqqqq
		53	u	0412	/		
		54	b	0520	/		
		55	m	0501	/	☒	
		56	r	0451	/		
		57	u	0430	/		forbt(wd)
		58	xr	0063	/		
		59	xu	0051	/	☒	binz(wd) (PIR 10.4)
		60	y	0516	/		...addr(wkloc)
		61	u	0412	/		
		62	c	0519	/		hex constant ac = 0
		63	xp	0000	/	☒	

☒ CARRIAGE RETURN

/ = CONDITIONAL STOP CODE

LGP-30 CODING SHEET

PREPARED FOR:				PAGE / OF
LGP-30 USERS' ORGANIZATION - POOL				9 / 11
JOB NO.	PROGRAM NO.	PROGRAM PREPARED BY:	PROGRAM CHECKED BY:	DATE
	H2-120	James N. Orton	POOL Review	2/5/60
PROBLEM: SYMBOLIC ASSEMBLY PROGRAM for the LGP-30 COMPUTER LGPSAP				TRACK

PROGRAM INPUT CODES	STOP	LOCATION	INSTRUCTION OPERATION	INSTRUCTION ADDRESS	STOP	CONTENTS OF ADDRESS	NOTES
	/						
	/	☒					
		04 00	x	i 0 0 0 0	/		input first half
		00 01	r	0 4 5 1	/		
		00 02	u	0 4 3 0	/		to 4-bit
		00 03	n	0 5 0 4	/	☒	at 15
		00 04	c	0 5 1 6	/		...wkloc
		00 05	x	p 0 0 0 0	/		
		00 06	x	i 0 0 0 0	/		input second half
		00 07	r	0 4 5 1	/	☒	
		00 08	u	0 4 3 0	/		to 4-bit
		00 09	m	0 5 0 5	/		at 31
		01 00	a	0 5 1 6	/		+ wkloc
		01 01	h	0 5 1 6	/	☒	...wkloc
		01 02	b	0 5 1 5	/		
		01 03	a	0 5 0 3	/		
		01 04	h	0 5 1 5	/		crloc = crloc + 1
		01 05	s	0 5 1 1	/	☒	
		01 06	t	0 4 1 8	/		
		01 07	u	0 0 3 5	/		if crloc) lzsyt
		01 08	b	0 5 1 5	/		
		01 09	y	0 4 2 1	/	☒	
		02 00	b	0 5 1 6	/		
		02 01	x	h 6 3 6 3	/		c(wkloc)...crloc
		02 02	b	0 5 1 5	/		
		02 03	x	r 0 6 0 0	/	☒	D3. print tab, sp,
		02 04	x	u 0 3 0 3	/		crloc, c(crloc), sp
		02 05	b	0 5 1 5	/		
		02 06	s	0 5 2 1	/		
		02 07	t	0 2 2 4	/	☒	if crloc is flag
		02 08	x	z 0 0 0 3	/		prog stop
		02 09	u	0 0 0 0	/		input start comp for next
		03 00	c	0 5 2 8	/		4-bit conv rtn. word at 31
		03 01	c	0 5 2 9	/	☒	4b wd = 0

FORM LP-10

PRINTED IN U.S.A.

☒ CARRIAGE RETURN

/ = CONDITIONAL STOP CODE

PREPARED FOR: LGP-30 USERS' ORGANIZATION - POOL				PAGE 10 OF 11
JOB NO.	PROGRAM NO. H2-120	PROGRAM PREPARED BY: James N. Orton	PROGRAM CHECKED BY: POOL Review	DATE 2/5/60
PROBLEM: SYMBOLIC ASSEMBLY PROGRAM for the LGP-30 COMPUTER LGPSAP				TRACK

PROGRAM INPUT CODES	STOP	LOCATION	INSTRUCTION OPERATION	INSTRUCTION ADDRESS	STOP	CONTENTS OF ADDRESS	NOTES
	/						
	/	☒					
		0 4 3 2	b	0 5 0 6	/		
		3 3	h	0 5 1 9	/		-4 into counter
		3 4	b	0 5 0 7	/		
		3 5	h	0 5 3 0	/	☒	initialize mask bits 26-9
		3 6	b	0 5 2 8	/		
		3 7	e	0 5 3 0	/		kth 4-bit char k=4,3,2,1
		3 8	a	0 5 2 9	/		
		3 9	h	0 5 2 9	/	☒	
		4 0	b	0 5 2 8	/		
		4 1	m	0 5 0 5	/		position next 4-bit char
		4 2	h	0 5 2 8	/		
		4 3	b	0 5 3 0	/	☒	
		4 4	n	0 5 0 8	/		left 4, set mask for next char
		4 5	h	0 5 3 0	/		
		4 6	b	0 5 1 9	/		
		4 7	a	0 5 0 2	/	☒	
		4 8	h	0 5 1 9	/		increase ctr by 1
		4 9	t	0 4 3 6	/		if ctr negative
		5 0	b	0 5 2 9	/		
		5 1	x	u 6 3 6 3	/	☒	exit
		5 2	z	0 1 1 6	/		
0 0 0 0 1 0		5 3	7	w w w w w w q	/		
		5 4	0 0	0 4 f j f f	/		
		5 5	0 0 0	0 0 0 7 q	/	☒	
		5 6	w w w	w w w w q	/		
		5 7	0 0 0	0 0 0 4 q	/		
		5 8	0 1	w w w w w q	/		
		5 9	0 2 0	0 0 0 0 0 0	/	☒	
		6 0	0 1	w w q 1 7 q	/		
		6 1	0 1	1 0 4 1 5 f	/		
		6 2	8 0 0	g 0 0 0 0	/		
		6 3		x z 3 2 0 0	/	☒	

LGP-30 CODING SHEET

PREPARED FOR: LGP-30 USERS' ORGANIZATION - POOL				PAGE 11 OF 11
JOB NO.	PROGRAM NO. H2-120	PROGRAM PREPARED BY: James N. Orton	PROGRAM CHECKED BY: POOL Review	DATE 2/5/60
PROBLEM: SYMBOLIC ASSEMBLY PROGRAM for the LGP-30 COMPUTER LGPSAP				TRACK

PROGRAM INPUT CODES	STOP	LOCATION	INSTRUCTION OPERATION	INSTRUCTION ADDRESS	STOP	CONTENTS OF ADDRESS	NOTES
	/						
	/	X					
		05 10	s	0 0 0 0	/		
0 0 0 0 0 0 0 0 0 0 1 2		10 11	4 0	0 0 0 0 0 0 0 0	/		
		10 12	0 0	0 0 0 0 0 0 0 2	/		
		10 13	x z	0 0 0 0 1	/	X	
0 0 0 0 0 0 0 0 0 0 1 3		10 14	0 0	0 0 4 0 0 0	/		
		10 15	2 0	0 0 0 0 0 0 0	/		
		10 16	w w	w w w w w 8	/		minus 4 at 30
		10 17	x z	0 0 1 5	/	X	
		10 18	x z	0 0 0 4	/		
		10 19	x z	0 4 0 0	/		
		11 10	z	0 5 3 1	/		final 10¢ plus 1
		11 11	x z	4 8 0 0	/	X	
		11 12	x z	5 6 0 0	/		
		11 13	x z	0 7 6 3	/		
		11 14	x z	0 0 0 0	/		
		11 15	x z	0 0 0 0	/	X	
		11 16	x z	0 0 0 0	/		
		11 17	x z	0 0 0 0	/		
		11 18	x z	0 0 0 0	/		
		11 19	x z	0 0 0 0	/	X	
		12 10	x z	0 0 0 0	/		
		12 11	x z	0 0 0 0	/		
0 0 0 0 0 0 0 0 0 0 1		12 12	0 5 k	2 1 k g f	/		Sell
		12 13	x z	0 0 0 0	/	X	
		12 14	x z	0 0 0 0	/		
		12 15	z	0 3 1 9	/		
		12 16	z	0 3 5 4	/		
q 0 0 0 0 0 0 0 0 0 1		12 17	q q q	q q q q q	/	X	
		12 18	x z	0 0 0 0	/		
		12 19	x z	0 0 0 0	/		
		13 10	x z	0 0 0 0	/		
		13 11	x z	0 0 0 0	/	X	

FORM LP-10 83-26 PRINTED IN U.S.A.

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

X = CARRIAGE RETURN
/ = CONDITIONAL STOP CODE

1026.53779