LGP-30 USERS' ORGANIZATION - POOL

LGPSAP

SYMBOLIC ASSEMBLY PROGRAM FOR THE LGP-30 COMPUTER

POOL Program No. H2-120

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

February 5, 1960

Program No. H2-120

LGPSAP

SYMBOLIC ASSEMBLY PROGRAM FOR THE LGP-30 COMPUTER

Table of Contents

* Arabic symbols following subsection titles are page numbers

LGP-30 USERS' ORGANIZATION - POOL

LGPSAP

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

LGPSAP

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.

```
                                                                  Notes
.0000700                                                          (see below)

p1 (First Pass)
4000'4000'                                                          1
'alpha'        b'        beta'        bring beta
               h'        gamma'       store into gamma
               xp'       0000'
               xi'       0000'
'beta'         xu'       6363'        exit                          3
'1x'           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                Decimal Code
Loc.          Op.       Addr.        Loc., Op., Addr.     Comments

'alpha'       b'        beta'        4000  b4004          bring beta       1,3

              h'        gamma'/u     4001  ,qqqqqqq       store into gamma 2

              xp'       0000'        4002  p0000                           4

              xi'       0000'        4003  i0000                           4

'beta'        xu'       6363'        4004  u6363          exit             3,4

 '1x'         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  ,qqqqqqq                        2

 'n'          '         '            4011  z0000         counter           7

 '4zqx'       '         11q2'        4012  ,000011q2                       5
```

Output: First pass. The sample printout following "pl" 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 "pl" 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:

```
,3089q172 is input from tape as 3089'q172'

,0009q172 can be input as          9'q172'

,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:

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

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.

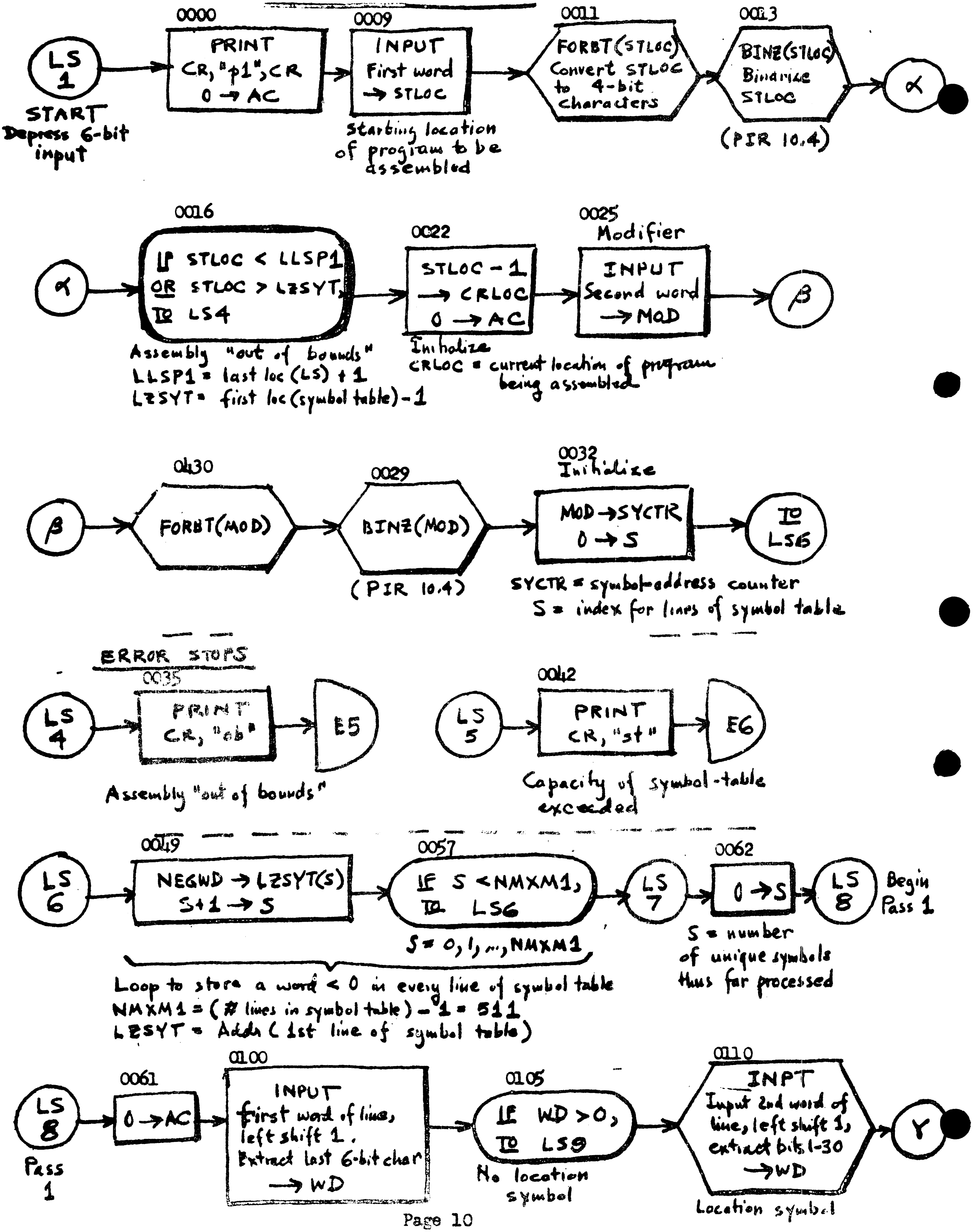
TABLE OF SYMBOLS FOR THE LGPSAP FLOW CHART

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

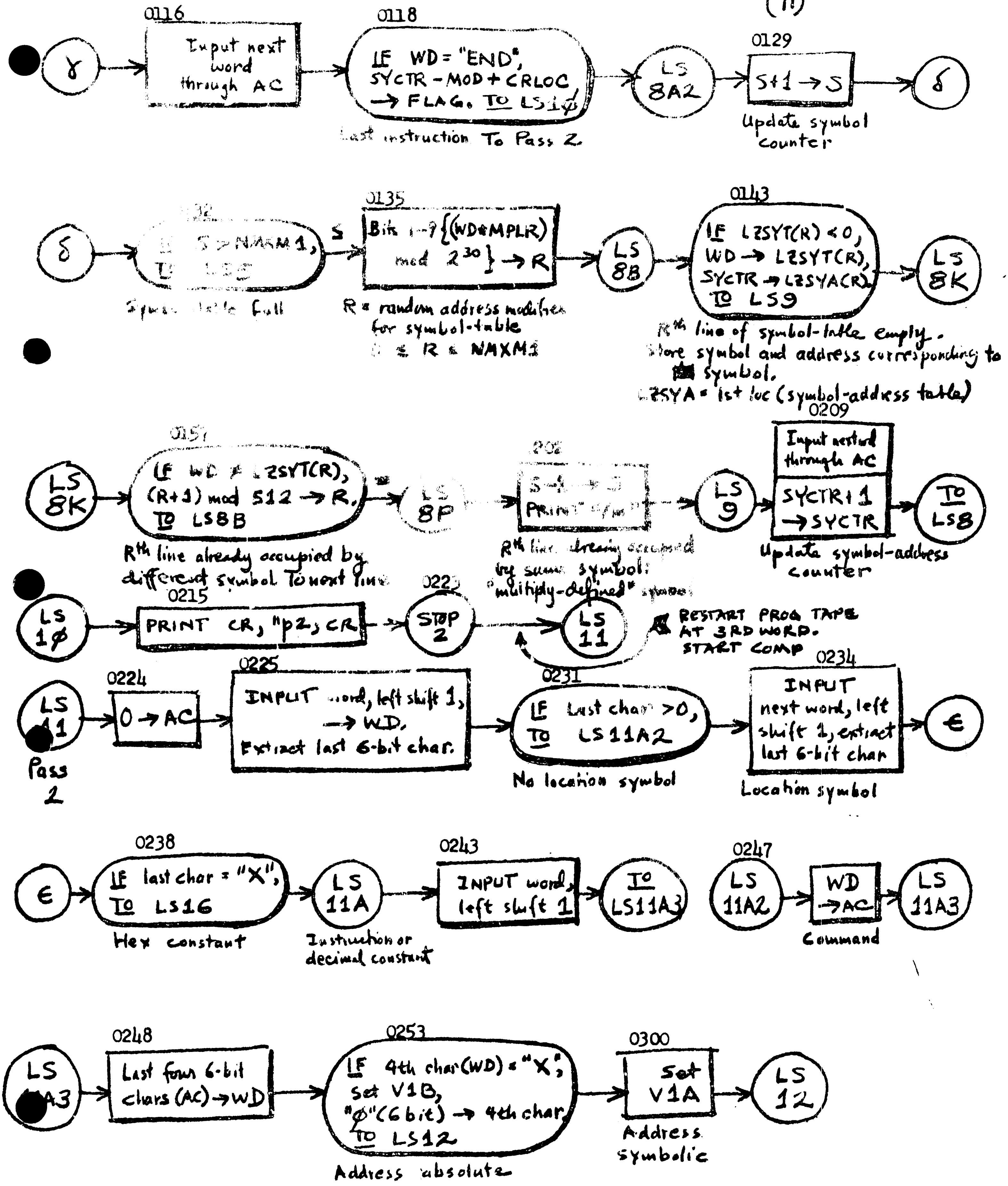
Table of Symbols Used on the Coding Sheet Notes

(	Less than
)	Greater than
*	Multiplied by
e	Exponent

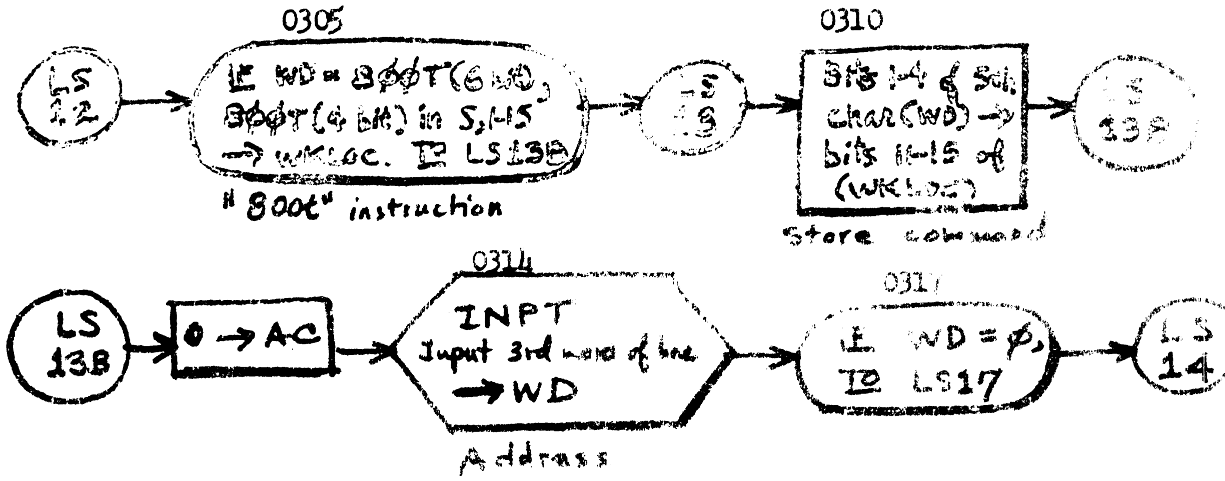
Flow Chart


0000
LS 1
START
Depress 6-bit input
PRINT
CR, "p1", CR
0 → AC
0009
INPUT
First word
→ STLOC
Starting location of program to be assembled
0011
FORBT(STLOC)
Convert STLOC to 4-bit characters
0013
BINZ(STLOC)
Binarize STLOC
(PIR 10.4)
α
α
0016
IF STLOC < LLSP1
OR STLOC > LZSYT,
TO LS4
Assembly "out of bounds"
LLSP1 = last loc (LS) + 1
LZSYT = first loc (symbol table) - 1
0022
STLOC - 1
→ CRLOC
0 → AC
Initialize
CRLOC = current location of program being assembled
0025
Modifier
INPUT
Second word
→ MOD
β
β
0030
FORBT(MOD)
0029
BINZ(MOD)
(PIR 10.4)
0032
Initialize
MOD → SYCTR
0 → S
TO LS6
SYCTR = symbol-address counter
S = index for lines of symbol table
ERROR STOPS
LS 4
0035
PRINT
CR, "ob"
E5
Assembly "out of bounds"
LS 5
0042
PRINT
CR, "st"
E6
Capacity of symbol-table exceeded
LS 6
0049
NEGWD → LZSYT(S)
S+1 → S
0057
IF S < NMXM1,
TO LS6
S = 0, 1, ..., NMXM1
LS 7
0062
0 → S
LS 8
Begin Pass 1
S = number of unique symbols thus far processed
Loop to store a word < 0 in every line of symbol table
NMXM1 = (# lines in symbol table) - 1 = 511
LZSYT = Addr (1st line of symbol table)
LS 8
Pass 1
0061
0 → AC
0100
INPUT
First word of line, left shift 1.
Extract last 6-bit char
→ WD
0105
IF WD > 0,
TO LS9
No location symbol
0110
INPT
Input 2nd word of line, left shift 1, extract bits 1-30
→ WD
Location symbol
γ

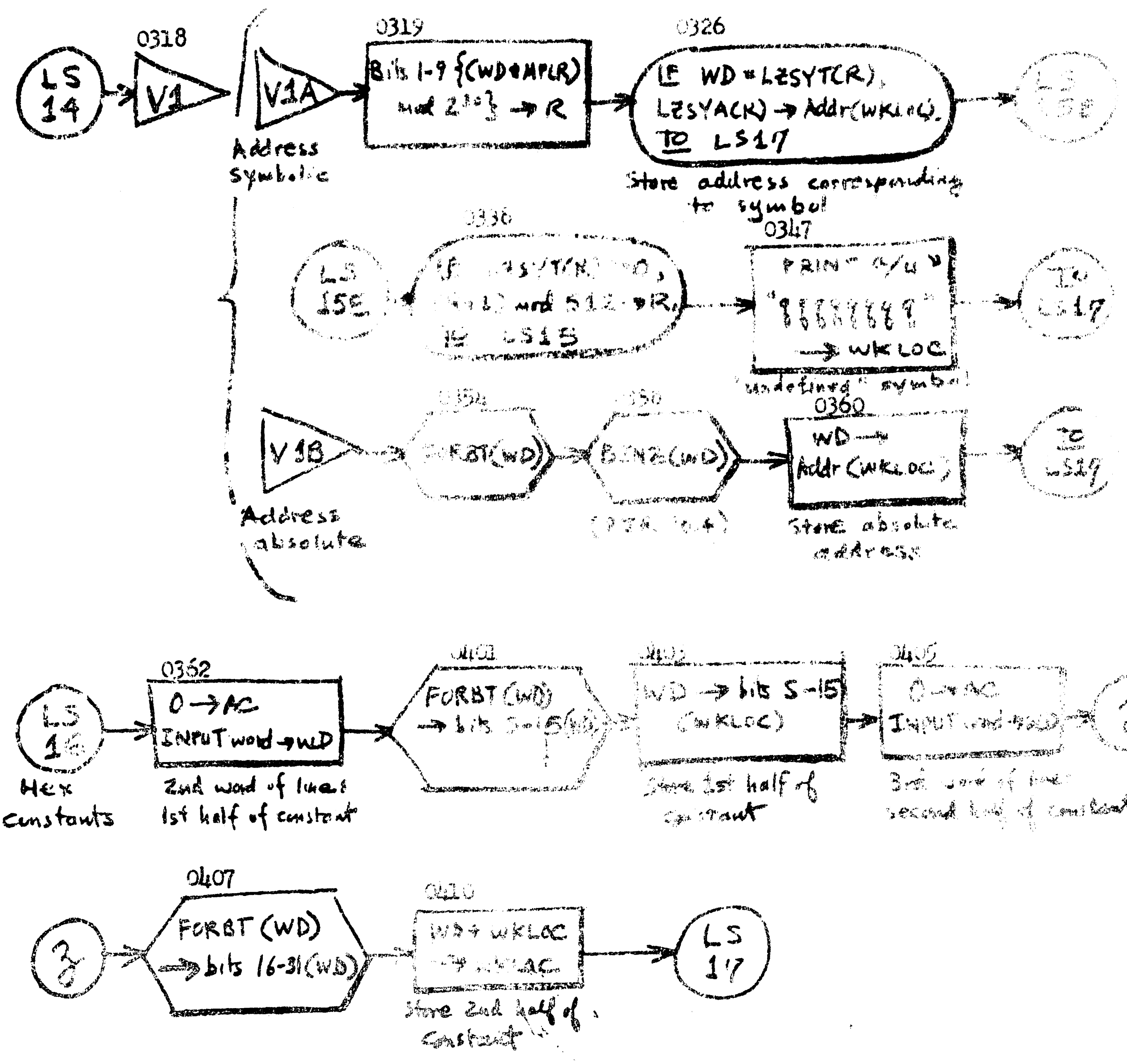
(ii)

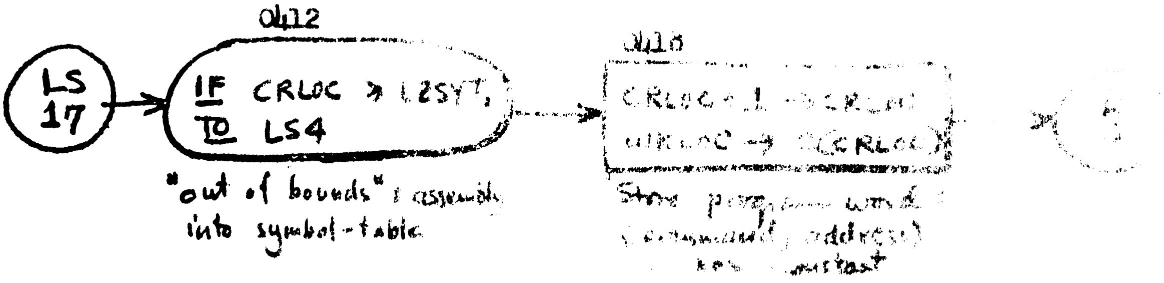


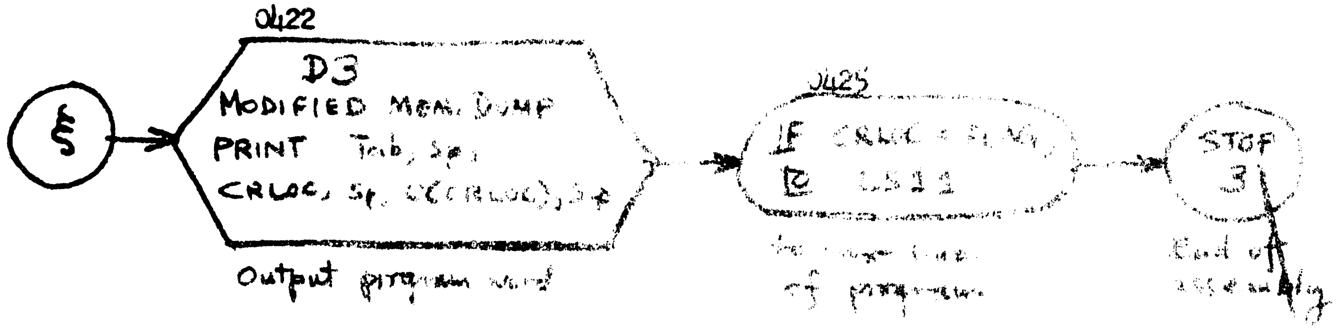
γ
0116
Input next word through AC
0118
IF WD = "END", SYCTR - MOD + CRLOC → FLAG. TO LS1φ
Last instruction To Pass 2
LS 8A2
0129
S+1 → S
Update symbol counter
δ
δ
0132
IF S > NMXM1, TO LS8
S
Symbol table full
0135
Bits 1-9 {(WD*MPLR) mod 2^30} → R
R = random address modifier for symbol-table
0 ≤ R ≤ NMXM1
LS 8B
0143
IF LZSYT(R) < 0, WD → LZSYT(R), SYCTR → LZSYA(R), TO LS9
LS 8K
Rth line of symbol-table empty. Store symbol and address corresponding to symbol.
LZSYA = 1st loc (symbol-address table)
LS 8K
0151
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 "S/m"
Rth line already occupied by same symbol: "multiply-defined" symbol
LS 9
0209
Input next word through AC
SYCTR+1 → SYCTR
Update symbol-address counter
TO LS8
LS 1φ
0215
PRINT CR, "p2, CR
0222
STOP 2
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 11A3
0248
Last four 6-bit chars (AC) → WD
0253
IF 4th char (WD) = "X", Set V1B, "φ" (6 bit) → 4th char, TO LS12
Address absolute
0300
Set V1A
Address symbolic
LS 12

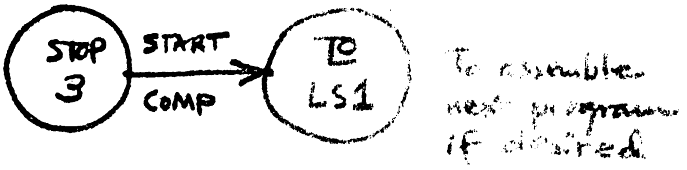

0305
LS 12
IF WD = 800T(6 WD), 800T(4 bit) in 5,1-15 → WKLOC. TO LS13B
"800t" instruction
LS 13
0310
Bits 1-4 of 5th char(WD) → bits 11-15 of (WKLOC)
Store command
LS 13B
LS 13B
0 → AC
0314
INPT Input 3rd word of line → WD
Address
0317
IF WD = ø, TO LS17
LS 14

Switch


LS 14
0318
V1
V1A
Address symbolic
0319
Bits 1-9 {(WD*MPLR) mod 2^10} → R
0326
IF WD = LZSYT(R), LZSYA(R) → Addr(WKLOC). TO LS17
Store address corresponding to symbol
LS 15E
LS 15E
0336
IF LZSYT(R) ≠ 0, (R+1) mod 512 → R, TO LS15
0347
PRINT "U" "99999999" → WKLOC
"undefined" symbol
TO LS17
V1B
Address absolute
0354
FORBT(WD)
0356
BIN2(WD)
(P3R 10.4)
0360
WD → Addr(WKLOC)
Store absolute address
TO LS19
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
Z
Z
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
0418
CRLOC + 1 → CRLOC
WRLOC → C(CRLOC)
Store program word (command, address) or constant
ξ


0422
ξ
D3
MODIFIED MEM. DUMP
PRINT Tab, Sp,
CRLOC, Sp, C(CRLOC), Sp
Output program word
0425
IF CRLOC ≠ FLWA,
TO LS11
for range 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)

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


'isl'xp'1600'        3000  p1600        first pass.  c.r.
xz'0000'             3001  z0000
xp'3300'             3002  p3300        p
xz'0000'             3003  z0000
xp'0600'             3004  p0600        l
xz'0000'             3005  z0000
xp'1600'             3006  p1600        c.r.
xz'0000'             3007  z0000
c'dump'              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  r0063
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'lzsyt'             3019  s0511
t'ls2'               3020  t0022
u'ls4'               3021  u0035        or stloc  ge  lzsyt
'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  r0063
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 bounds.    cr
xs'0000'             3036  z0000
xp'3500'             3037  p3500        o
xs'0000'             3038  z0000
xp'0500'             3039  p0500        b
xs'0000'             3040  z0000
xs'0005'             3041  z0005        error stop 5
'ls5'xp'1600'        3042  p1600        symbol table full    cr
xz'0000'             3043  z0000
xp'6100'             3044  p6100        s
xz'0000'             3045  z0000
```

LGP-30 USERS' ORGANIZATION - POOL

Coding for Assembly Routine (LGPSAP and Decimal)

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

```
xp'4500'              3046  p4500      t
xz'0000'              3047  z0000
xz'0006'              3048  z0006      error stop 6
'ls6'b'lssyt'         3049  b0511
s's'                  3050  s0524
y'ls6a'               3051  y0053
b'negwdx'             3052  b0456
'ls6a'xh'6363'        3053  h6363      lssyt.s = negwd
b's'                  3054  b0524
a'ls29'               3055  a0503
h's'                  3056  h0524      s = s + 1
b'nnnnl'              3057  b0513
s's'                  3058  s0524
t'ls7'                3059  t0061
u'ls6'                3060  u0049      if s  le  nnnnl
'ls7'c'dump'          3061  c0519
c's'                  3062  c0524      s = 0
'ls8'c'dump'          3063  c0519
xp'0000'              3100  p0000
xi'0000'              3101  i0000      1st wd of line
n'ls30x'              3102  n0502
e'mfx'                3103  e0455      last 6b char
c'wd'                 3104  c0520
s'wd'                 3105  s0520
t'ls9'                3106  t0209      if wd  gr  0
b'cl8al'              3107  b0452
y'input'              3108  y3115
'input'c'dump'        3109  c0519      ac = 0
xp'0000'              3110  p0000
xi'0000'              3111  i0000      input location symbol
n'ls30x'              3112  n0502
e'mlx'                3113  e0453
h'wd'                 3114  h0520      ... wd at 30
'input'xu'6363'       3115  u6363
'ls8al'xp'0000'       3116  p0000
xi'0000'              3117  i0000      next wd through ac
b'wd'                 3118  b0520
s'oendx'              3119  s0454
t'ls8a2'              3120  t0129
s'ls30x'              3121  s0502
t'ls8a'               3122  t0124      if wd = "end",
u'ls8a2'              3123  u0129
'ls8a'b'syctr'        3124  b0513
s'mod'                3125  s0517
a'erloc'              3126  a0515
h'flag'               3127  h0521      syctr - mod + erloc
u'ls10'               3128  u0213      ... flag
'ls8a2'b's'           3129  b0524
a'ls29'               3130  a0503
h's'                  3131  h0524      s = s + 1
b'nnnnl'              3132  b0513
s's'                  3133  s0524
t'ls5'                3134  t0042      if s  gr  nnnnl
```

Coding for Assembly Routine (LGPSAP and Decimal)

```
b'wd'                  3135   b0520
n'mplrx'               3136   n0522      (wd*mplr)mod 2e30  at 29
m'1s21'                3137   m0509      bits 0-8 to 21-29
e'mmmal'               3138   e0513      21-29
'1s8b'h'r'             3139   h0523      ... r   random addr modifier
a'lsayt'               3140   a0511
y'1s8c'                3141   y0143
y'1s8e'                3142   y0147
'1s8c'xb'6363'         3143   b6363      /lsayt + r/
t'1s8d'                3144   t0146      if lsayt.r  is  0,
u'1s8k'                3145   u0154
'1s8d'b'wd'            3146   b0520
'1s8e'xh'6363'         3147   h6363      lsayt.r = wd,
b'lsaya'               3148   b0512
a'r'                   3149   a0523
y'1s8r'                3150   y0152
b'syctr'               3151   b0518
'1s8r'xh'6363'         3152   h6363      lsaya.r = syctr.
u'1s9'                 3153   u0209
'1s8k'b'lsayt'         3154   b0511
a'r'                   3155   a0523
y'1s8m'                3156   y0158
b'wd'                  3157   b0520
'1s8m'xs'6363'         3158   s6363      /lsayt + r/
t'1s8h'                3159   t0162      if wd  ne  lsayt.r,
s'1s90x'               3160   s0502
t'1s8y'                3161   t0202
'1s8h'b'r'             3162   b0523
a'1s29'                3163   a0503
e'mmmal'               3200   e0513      r = (r + 1)mod rmax.
u'1s8o'                3201   u0159
'1s8y'b's'             3202   b0524      multiply-defined sym
s'1s29'                3203   s0503
h's'                   3204   h0524      s = s - 1
xp'1900'               3205   p1900      /
xz'0000'               3206   z0000
xp'2500'               3207   p2500      m
xz'0000'               3208   z0000
'1s9'xp'0000'          3209   p0000
xi'0000'               3210   i0000      next wd through ac
b'syctr'               3211   b0518
a'1s29'                3212   a0503
h'syctr'               3213   h0518      syctr = syctr + 1
u'[illegible]'         3214   u0063
'1s10'xp'1600'         3215   p1600      second pass.   cr
xz'0000'               3216   z0000
xp'3500'               3217   p3500      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
```

Coding for Assembly Routine (LGPSAP and Decimal)

```
'1a11'c'dump'        3224  c0519
xp'0000'             3225  p0000     1st wd of line
xi'0000'             3226  i0000
n'1a30x'             3227  n0502
h'wd'                3228  h0520
e'm2x'               3229  e0455     last 6b char
c'dump'              3230  c0519
s'dump'              3231  s0519
t'1a11a2'            3232  t0247     if char  gr  0
c'dump'              3233  c0519     ac = 0
xp'0000'             3234  p0000
xi'0000'             3235  i0000     input loc symbol
n'1a30x'             3236  n0502
e'm2x'               3237  e0455     last char
s'charx'             3238  s0457
t'1a11a'             3239  t0242
s'1a30x'             3240  s0502
t'1a16'              3241  t0362     if last char = "x"
'1a11a'c'dump'       3242  c0519     ac = 0
xp'0000'             3243  p0000
xi'0000'             3244  i0000     input instruction
n'1a30x'             3245  n0502
u'1a11a3'            3246  u0248
'1a11a2'b'wd'        3247  b0520
'1a11a3'e'm3x'       3248  e0458     last 4 char
h'wd'                3249  h0520
m'1c6x'              3250  m0459
e'm2x'               3251  e0455
s'charx'             3252  s0457
t'1a11e'             3253  t0263
s'1a40x'             3254  s0502
t'1a11b'             3255  t0257     if 4th char = "x",
u'1a11e'             3256  u0263
'1a11b'b'cvlb'       3257  b0526
y'1a14'              3258  y0318     set vlb
b'wd'                3259  b0520
e'm4x'               3260  e0460
h'wd'                3261  h0520     4th char = "0"
u'1a12'              3262  u0301
'1a11e'b'cvla'       3263  b0525
y'1a14'              3300  y0318     set vla
'1a12'b'wd'          3301  b0520
s'80t6x'             3302  s0461
t'1a13'              3303  t0310
s'1a30x'             3304  s0502
t'1a12a'             3305  t0307     if wd = 800t(6-bit),
u'1a13'              3306  u0310
'1a12a'b'80t4x'      3307  b0462
h'wkloc'             3308  h0516     wkloc = 800t(4-bit).
u'1a13b'             3309  u0314
'1a13'b'wd'          3310  b0520     store command
n'1a18'              3311  n0463
e'm5x'               3312  e0500     1, s0000;
h'wkloc'             3313  h0516
```

```
'lal3b'r'input'      3314  r0115
u'input'             3315  u0109     input address symbol
a'la30x'             3316  a0502
t'lal7'              3317  t0412     if = 0
'lal4'u'lal4'        3318  u0318     vc 1
'vla'b'wd'           3319  b0520
n'mplrx'             3320  n0522
m'la21'              3321  m0509
e'rmaxl'             3322  e0513     bits 1-9[wd*mplr]mod 2e30
'lal5'h'r'           3323  h0523     ... r
a'lzsyt'             3324  a0511
y'lal5a'             3325  y0326
'lal5a'xb'6363'      3326  b6363     /lzsyt + r/
s'wd'                3327  s0520
t'lal5e'             3328  t0338
a'la30x'             3329  a0502
t'lal5b'             3330  t0332     if wd = lzsyt.r,
u'lal5e'             3331  u0338
'lal5b'b'r'          3332  b0523
a'lzsya'             3333  a0512
y'lal5c'             3334  y0335
'lal5c'xb'6363'      3335  b6363     /lzsya + r/
y'wkloc'             3336  y0516     addr(wkloc) = lzsya.r
u'lal7'              3337  u0412
'lal5e'b'r'          3338  b0523
a'lzsyt'             3339  a0511
y'lal5f'             3340  y0341
'lal5f'xb'6363'      3341  b6363     /lzsyt + r/
t'lal5g'             3342  t0347
b'r'                 3343  b0523     if lzsyt.r  gr  0
a'la29'              3344  a0503
e'rmaxl'             3345  e0513     r = (r + 1)mod rmax
u'lal5'              3346  u0323
'lal5g'xp'1900'      3347  p1900     undefined symbol.   /
xz'0000'             3348  z0000
xp'4100'             3349  p4100     u
xz'0000'             3350  z0000
b'cqqqx'             3351  b0527
h'wkloc'             3352  h0516     wkloc = qqqqqqqq
u'lal7'              3353  u0412
'vlb'b'wd'           3354  b0520
m'lalx'              3355  m0501
r'frxt'              3356  r0451
u'forbt'             3357  u0430     forbt(wd)
xr'0063'             3358  r0063
xu'0051'             3359  u0051     binz(wd)   (PIR 10.4)
y'wkloc'             3360  y0516     ... addr(wkloc)
u'lal7'              3361  u0412
'lal6'c'dump'        3362  c0519     hex constant   ac = 0
xp'0000'             3363  p0000
xi'0000'             3400  i0000     input first half
r'frxt'              3401  r0451
u'forbt'             3402  u0430     to 4-bit
```

Coding for Assembly Routine (LGPSAP and Decimal)

```
n'1al7x'              3403  n0504     at 15
c'wkloc'              3404  c0516     ... wkloc
xp'0000'              3405  p0000
xi'0000'              3406  i0000     input second half
r'frmt'               3407  r0451
u'forbt'              3408  u0430     to 4-bit
m'1a2x'               3409  m0505     at 31
a'wkloc'              3410  a0516     + wkloc
h'wkloc'              3411  h0516     ... wkloc
'1al7'b'crloc'        3412  b0515
a'1a09'               3413  a0503
h'crloc'              3414  h0515     crloc = crloc + 1
s'1zsyt'              3415  s0511
t'1al7a'              3416  t0418
u'1s4'                3417  u0035     if crloc  ge  lzsyt
'1al7a'b'crloc'       3418  b0515
y'1al7b'              3419  y0421
b'wkloc'              3420  b0516
'1al7b'xh'6363'       3421  h6363     c(wkloc) ... crloc
b'crloc'              3422  b0515
xr'0600'              3423  r0600     DS.   print tab, sp,
xu'0303'              3424  u0303     crloc, c(crloc), sp
b'crloc'              3425  b0515
s'flag'               3426  s0521
t'1sl1'               3427  t0224     if crloc  ls  flag
xz'0003'              3428  z0003     prog stop
u'1sl'                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'min4x'              3432  b0506
h'dump'               3433  h0519     -4 into counter
b'c15'                3434  b0507
h'mask'               3435  h0530     initialize mask  bits 26-9
'fr1'b'6bwd'          3436  b0528
e'mask'               3437  e0530     kth 4-bit char  k = 4,3,2,1
a'4bwd'               3438  a0529
h'4bwd'               3439  h0529
b'6bwd'               3440  b0528
m'1a2x'               3441  m0505     position next 4-bit char
h'6bwd'               3442  h0528
b'mask'               3443  b0530
n'c4'                 3444  n0508     left 4. set mask for next char
h'mask'               3445  h0530
b'dump'               3446  b0519
a'1c1x'               3447  a0509
h'dump'               3448  h0519     increase ctr by 1
t'fr1'                3449  t0436     if ctr negative
b'4bwd'               3450  b0529
'fr4x'xu'6363'        3451  u6363     exit
```

Coding for Assembly Routine (LGPSAP and Decimal)

```
'cl8al''ls8al'            3452  z0116
'mlx'7www'wwwq'           3453  ,7wwwwwwq
'cendx'4'fjff'            3454  ,0004fjff
'm2x''7q'                 3455  ,0000007q
'ngwdx'wwww'wwwq'         3456  ,wwwwwwwq
'charm''4q'               3457  ,0000004q
'm3x'1ww'wwwq'            3458  ,01wwwwwq
'1n6x'200'0000'           3459  ,02000000
'mkx'1ww'q17q'            3460  ,01wwq17q
'80t6x'110'415f'          3461  ,0110415f
'80t4x'8002'0000'         3462  ,08020000
'1s18'xs'3200'            3463  z3200
'm5x'w'0000'              3500  z0000
'1s1x'4000'0000'          3501  ,40000000
'1s30x''2'                3502  ,00000002
'1s29'xs'0001'            3503  z0001
'1s17x''4000'             3504  ,00004000
'1s2x'2000'0000'          3505  ,20000000
'min4x'wwww'www8'         3506  ,wwwwwww8         minus 4 at 30
'c15'xs'0015'             3507  z0015
'c4'xs'0004'              3508  z0004
'1s21'xs'0400'            3509  z0400
'llspl''finwd'            3510  z0531             final loc plus 1
'lzsyt'xs'4800'           3511  z4800
'lzsya'xs'5600'           3512  z5600
'mxcol'xs'0763'           3513  z0763             symtbl mxcol minus 1
'stloc'''                 3514  z0000
'crloc'''                 3515  z0000
'wtloc'''                 3516  z0000
'mod'''                   3517  z0000
'syctr'''                 3518  z0000
'dump'''                  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
'6wd'''                   3528  z0000
'4wd'''                   3529  z0000
'wmsk'''                  3530  z0000
'finwd'''                 3531  z0000
```

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	xu0363

LGP-30 CODING SHEET

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		STOP	CONTENTS OF ADDRESS	NOTES
			OPERATION	ADDRESS			
	/						
	/	☒					
		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 (11spl
		18	b	0514	/		
		19	s	0511	/	☒	
		20	t	0022	/		
		21	u	0035	/		or stloc) 1zsyt
		22	b	0514	/		
		23	s	0503	/	☒	
		24	c	0515	/		crloc = stloc - 1.ac=0
		25	x p	00q0	/		
		26	x i	00q0	/		input modifier
		27	r	0451	/	☒	
		28	u	0430	/		forbt (mod)
		29	x r	0063	/		
		30	x u	0051	/		
		31	h	0517	/	☒	binz(mod)

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 2 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
	/						
	/	☒					
		32	c	0518	/		syctr = mod
		33	c	0524	/		s = 0
		34	u	0049	/		
		35	x p	1600	/	☒	out of bounds. cr
		36	x z	0000	/		
		37	x p	3500	/		o
		38	x z	0000	/		
		39	x p	0500	/	☒	b
		40	x z	00q0	/		
		41	x z	0005	/		error stop 5
		42	x p	1600	/		symbol table full cr
		43	x z	00q0	/	☒	
		44	x p	6100	/		s
		45	x z	0000	/		
		46	x p	4500	/		t
		47	x z	00q0	/	☒	
		48	x z	0006	/		error stop 6
		49	b	0511	/		
		50	a	0524	/		
		51	y	0053	/	☒	
		52	b	0456	/		
		53	x h	6363	/		lzsyt(s) = negwd
		54	b	0524	/		
		55	a	0503	/	☒	
		56	h	0524	/		s = s + 1
		57	b	0513	/		
		58	s	0524	/		
		59	t	0061	/	☒	
		60	u	0049	/		if s (nmxml
		61	c	0519	/		
		62	c	0524	/		s = 0
		63	c	0519	/	☒	

FORM LP-12

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 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		STOP	CONTENTS OF ADDRESS	NOTES
			OPERATION	ADDRESS			
	/						
	/	☒					
		0100	x	p0000	/		
		01	x	i0000	/		1st wd of line
		02		n0502	/		
		03		e0455	/	☒	last 6b char
		04		c0520	/		
		05		s0520	/		
		06		t0209	/		if wd) 0
		07		b0452	/	☒	
		08		y0115	/		
		09		c0519	/		ac = 0
		10	x	p0000	/		
		11	x	i0000	/	☒	input location symbol
		12		n0502	/		
		13		e0453	/		
		14		h0520	/		...wd at 30
		15	x	u6363	/	☒	
		16	x	p0000	/		
		17	x	i0000	/		next wd through ac
		18		b0520	/		
		19		s0454	/	☒	
		20		t0129	/		
		21		s0502	/		
		22		t0124	/		if wd = "end,"
		23		u0129	/	☒	
		24		b0518	/		
		25		s0517	/		
		26		a0515	/		syctr - mod + crloc
		27		h0521	/	☒	...flag
		28		u0215	/		
		29		b0524	/		
		30		a0503	/		
		31		h0524	/	☒	s = s + 1

☒ CARRIAGE RETURN

/ = CONDITIONAL STOP CODE

LGP-30 CODING SHEET

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

TRACK

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

PROGRAM INPUT CODES	STOP	LOCATION	INSTRUCTION OPERATION	INSTRUCTION ADDRESS	STOP	CONTENTS OF ADDRESS	NOTES
	/						
	/	☒					
		0132	b	0513	/		
		33	s	0524	/		
		34	t	0042	/		if s) nmxml
		35	b	0520	/	☒	
		36	n	0522	/		(wd*mplr)mod 2e30 at 29
		37	m	0509	/		bits 0-8 to 21-29
		38	e	0513	/		21-29
		39	h	0523	/	☒	...r random addr.mod-
		40	a	0511	/		ifier
		41	y	0143	/		
		42	y	0147	/		
		43	x b	6363	/	☒	/lzsyt + r/
		44	t	0146	/		if lzsyt(r) (ls 0,
		45	u	0154	/		
		46	b	0520	/		
		47	x h	6363	/	☒	lzsyt(r) = wd,
		48	b	0512	/		
		49	a	0523	/		
		50	y	0152	/		
		51	b	0518	/	☒	
		52	x h	6363	/		lzsya(r) = syctr.
		53	u	0209	/		
		54	b	0511	/		
		55	a	0523	/	☒	
		56	y	0158	/		
		57	b	0520	/		
		58	x s	6363	/		/lzsyt + r/
		59	t	0162	/	☒	if wd / lzsyt(r)
		60	s	0502	/		
		61	t	0202	/		
		62	b	0523	/		
		63	a	0503	/	☒	

FORM LP-12

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 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
	/						
	/	☒					
		0200	e	0513	/		r = (r+1)mod nmax.
		01	u	0139	/		
		02	b	0524	/		multiply-defined sym
		03	s	0503	/	☒	
		04	h	0524	/		s = s - 1
		05	xp	1900	/		/
		06	xz	0000	/		
		07	xp	2900	/	☒	m
		08	xz	0000	/		
		09	xp	0000	/		
		10	xi	0000	/		next wd through ac
		11	b	0518	/	☒	
		12	a	0503	/		
		13	h	0518	/		syctr = syctr + 1
		14	u	0063	/		
		15	xp	1600	/	☒	second pass. cr
		16	xz	0000	/		
		17	xp	3300	/		p
		18	xz	0000	/		
		19	xp	1000	/	☒	2
		20	xz	0000	/		
		21	xp	1600	/		cr
		22	xz	0000	/		
		23	xz	0002	/	☒	stop 2. restart tape
		24	c	0519	/		
		25	xp	0000	/		1st wd of line
		26	xi	0900	/		
		27	n	0502	/	☒	
		28	h	0520	/		
		29	e	0455	/		last 6b char
		30	c	0519	/		
		31	s	0519	/	☒	

FORM LP-10

PRINTED IN U.S.A.

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

☒ CARRIAGE RETURN

/ = CONDITIONAL STOP CODE

LGP-30 CODING SHEET

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		STOP	CONTENTS OF ADDRESS	NOTES
			OPERATION	ADDRESS			
	/						
	/	☒					
		0232	t	0247	/		if char) 0
		33	c	0519	/		ac = 0
		34	xp	0000	/		
		35	xi	0010	/	☒	input loc symbol
		36	n	0502	/		
		37	e	0455	/		last char
		38	s	0457	/		
		39	t	0242	/	☒	
		40	s	0502	/		
		41	t	0362	/		if last char = "x"
		42	c	0519	/		ac = 0
		43	xp	0010	/	☒	
		44	xi	0000	/		input instruction
		45	n	0502	/		
		46	u	0248	/		
		47	b	0520	/	☒	
		48	e	0458	/		last 4 char
		49	h	0520	/		
		50	m	0459	/		
		51	e	0455	/	☒	
		52	s	0457	/		
		53	t	0263	/		
		54	s	0502	/		
		55	t	0257	/	☒	if 4th char = "x"
		56	u	0263	/		
		57	b	0526	/		
		58	y	0318	/		set vlb
		59	b	0520	/	☒	
		60	e	0460	/		
		61	h	0520	/		4th char = "0"
		62	u	0301	/		
		63	b	0525	/	☒	

FORM LP-12

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 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		STOP	CONTENTS OF ADDRESS	NOTES
			OPERATION	ADDRESS			
	/						
	/	☒					
		0300		y0318	/		set vla
		01		b0520	/		
		02		s0461	/		
		03		t0310	/	☒	
		04		s0502	/		
		05		t0307	/		if wd = 800t(6-bit).
		06		u0310	/		
		07		b0462	/	☒	
		08		h0516	/		wkloc = 800t(4-bit).
		09		u0314	/		
		10		b0520	/		~~store command~~
		11		n0463	/	☒	
		12		e0500	/		[,w0000]
		13		h0516	/		
		14		r0115	/		
		15		u0109	/	☒	input address symbol
		16		s0502	/		
		17		t0412	/		if = 0
		18		u0318	/		vc 1
		19		b0520	/	☒	
		20		n0522	/		
		21		m0509	/		
		22		e0513	/		bits 1-9 [wd*mplr] 2e30 mod
		23		h0523	/	☒	...r
		24		a0511	/		
		25		y0326	/		
		26	x	b6363	/		/lzsyt + r/
		27		s0520	/	☒	
		28		t0318	/		
		29		s0502	/		
		30		t0312	/		if wd = lzsyt(r)
		31		u0318	/	☒	

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	x b	6363	/	☒	/lzsya + r/
		36	y	0516	/		addr(wkloc)=lzsya(r)
		37	u	0412	/		
		38	b	0523	/		
		39	a	0511	/	☒	
		40	y	0341	/		
		41	x b	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	x p	1900	/	☒	undefined symbol. /
		48	x z	0000	/		
		49	x p	4100	/		u
		50	x z	0000	/		
		51	b	0527	/	☒	
		52	h	0516	/		wkloc = qqqqqqqq
		53	u	0412	/		
		54	b	0520	/		
		55	m	0501	/	☒	
		56	r	0451	/		
		57	u	0430	/		forbt(wd)
		58	x r	0063	/		
		59	x u	0051	/	☒	binz(wd) (PIR 10.4)
		60	y	0516	/		...addr(wkloc)
		61	u	0412	/		
		62	c	0519	/		hex constant ac = 0
		63	x p	0000	/	☒	

FORM LP-12

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 9 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		STOP	CONTENTS OF ADDRESS	NOTES
			OPERATION	ADDRESS			
	/						
	/	☒					
		0400	xi	0000	/		input first half
		01	r	0451	/		
		02	u	0430	/		to 4-bit
		03	n	0504	/	☒	at 15
		04	c	0516	/		...wkloc
		05	xp	0000	/		
		06	xi	0000	/		input second half
		07	r	0451	/	☒	
		08	u	0430	/		to 4-bit
		09	m	0505	/		at 31
		10	a	0516	/		+ wkloc
		11	h	0516	/	☒	...wkloc
		12	b	0515	/		
		13	a	0503	/		
		14	h	0515	/		crloc = crloc + 1
		15	s	0511	/	☒	
		16	t	0418	/		
		17	u	0035	/		if crloc) lzsyt
		18	b	0515	/		
		19	y	0421	/	☒	
		20	b	0516	/		
		21	xh	6363	/		c(wkloc)...crloc
		22	b	0515	/		
		23	xr	0600	/	☒	D3. print tab, sp,
		24	xu	0303	/		crloc, c(crloc), sp
		25	b	0515	/		
		26	s	0521	/		
		27	t	0224	/	☒	if crloc is flag
		28	xz	0003	/		prog stop
		29	u	0000	/		start comp for next input
		30	c	0528	/		4-bit conv rtn. word at 31
		31	c	0529	/	☒	4b wd = 0

☒ CARRIAGE RETURN

/ = CONDITIONAL STOP CODE

LGP-30 CODING SHEET

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

TRACK

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

PROGRAM INPUT CODES	STOP	LOCATION	INSTRUCTION	STOP	CONTENTS OF ADDRESS	NOTES
	/					
	/	☒				
		0432	b0506	/		
		33	h0519	/		-4 into counter
		34	b0507	/		
		35	h0530	/	☒	initialize mask bits 26-9
		36	b0528	/		
		37	e0530	/		kth 4-bit char k=4,3,2,1
		38	a0529	/		
		39	h0529	/	☒	
		40	b0528	/		
		41	m0505	/		position next 4-bit char
		42	h0528	/		
		43	b0530	/	☒	
		44	n0508	/		left 4. set mask for next char
		45	h0530	/		
		46	b0519	/		
		47	a0502	/	☒	
		48	h0519	/		increase ctr by 1
		49	t0436	/		if ctr negative
		50	b0529	/		
		51	x u6363	/	☒	exit
		52	z0116	/		
,00000010		53	7wwwwwwq	/		
		54	0004fjff	/		
		55	0000007q	/	☒	
		56	wwwwwwwq	/		
		57	0000004q	/		
		58	01wwwwwq	/		
		59	02000000	/	☒	
		60	01wwq17q	/		
		61	0110415f	/		
		62	800g0000	/		
		63	x z3200	/	☒	

☒ CARRIAGE RETURN

/ = CONDITIONAL STOP CODE

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		STOP	CONTENTS OF ADDRESS	NOTES
			OPERATION	ADDRESS			
	/						
	/	☒					
		0500	s	0010	/		
,00000012		01	4000	0010	/		
		02	0000	0002	/		
		03	xz	0011	/	☒	
,00000003		04	0000	4010	/		
		05	2000	0000	/		
		06	wwww	www8	/		minus 4 at 30
		07	xz	0015	/	☒	
		08	xz	0004	/		
		09	xz	0400	/		
		10	z	0531	/		final 10¢ plus 1
		11	xz	4810	/	☒	
		12	xz	5600	/		
		13	xz	0763	/		
		14	xz	0000	/		
		15	xz	0000	/	☒	
		16	xz	0000	/		
		17	xz	0000	/		
		18	xz	0000	/		
		19	xz	0000	/	☒	
		20	xz	0000	/		
		21	xz	0000	/		
,00000001		22	05k2	1kgf	/		5ell
		23	xz	0000	/	☒	
		24	xz	0000	/		
		25	z	0319	/		
		26	z	0354	/		
,q000001		27	gggg	gggg	/	☒	
		28	xz	0000	/		
		29	xz	0000	/		
		30	xz	0000	/		
		31	xz	0000	/	☒	

FORM LP-10

PRINTED IN U.S.A.

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

☒ CARRIAGE RETURN

/ = CONDITIONAL STOP CODE