CODING FORM 1960-1961

Don Faust

12
11
10
9
8
7
6
5
4
3
2
1

(user will have to fix up the ending conditions)

6/card and 12/card loader:


```
7000:   36 3980
7001:   72 7017
7002:  -81 44 0000
7003:  -64 0000
7004:  -02 7016
7005:  -73 0000
7006:   28 7008
7007:  -12 0001
7008:  -12 0000
7009:   72 7017
7010:  -26 0001
7011:   66 7016
7012:   12 7017
7013:   63 7018
7014:   12 7016
7015:   20 7000
7016:   60nn
7017:   nnnn
7018:   6011
```

6/w	12/w
-1	-1
1	2
2	3
3	4
4	5
5	6
6	7
=6=	8
	9
	10
	11
	12
	=1=

7007 7005 7004 7003 7002 7001 7000

7018 7017 7016

=6= last card: 20 7011 | 6011 | | |

punches:

```
5     ADD BASE
      STA BASE
      LDB BASE
3  -  CAD 1
      SRT 10
      CIRA 0
      CNZ 1F
      SLT 10
      CNZ 1F
4     IB 12
      BUN 3B
1     BA 11
      LDB BLASE
      STA BASE
      EXT =64 4011=
      STA BLASE  STA TRY
      CSU BASE
   -  STA 0          CSU BASE
      SUB =999999 6000=
      BOF           2-7
WRITE -0f20 CHR 0
   -  LDB 0
   -  BT4 0 ——— CAD =1=
                -STA 13
      LDB 1B
1     CAD TRY
      ADD =1=
      STA TRY
TRY   CAD 0
16    SRT 6
      CIRA
      ADD =9 —— 98=
      BOF 1F
34    DBB 1B
      CAD (f=  )
      STA WRITE
      CAD 4B
      BUN 5B
```

```
1 0   LDB BLASE
      CAD 6E)
      STA WRITE
      CAD 6B
      -STA 7
      BUN 5B
6     [illegible]
7
8     =64 4011=
9     =9 —— 96000=
10    =9 —— 98=
11       =1=
12    (     )
13    (     )
14    BASE = 0=
15    BLASE=4000=
```

20 6 = 120

quasifields with 32 elements.

Sign
2 add in proper temp storage
4 relocatable address
6 pseudo ops in 42-field
3 special use

Pseudo ops

PSI 70 value of I
PSJ 71 value of J
ADD 72 put out addition macro
73 terminate macro
NEXT 74 beginning of program is next line;
75 STP E(n-1)
76 BUN E(n-2)
? 78 exit line E n.
BOOK 79 the bookkeeping and checking macro, exit line

```
      FBGR PRINT, 20 (T5Z5NB)
      CUF *-1, 22.
ASMBL CLL LOCN
      LDR +BEGIN
      STR LINE
      CLL VNO
ST    LDB LINE
      -CAD 0
      BSA A+,4
      BSA -0,6
      BSA V+,2
      BSA SPEC,3
RE    LDB LOCN
      -STA 0
      IFL LOCN,00,1
RE1   IFL LINE,00,1
      BUN ST
*A    SUB LINE
      ADD LOCN
      BUN RE
PSIJ  STA J,01    SRT 6
      BUN RE1     SRT 1
                  STA I,01
ADD   LDR LINE
      STR LINE1
      LDR +MAC
      STR LINE
      DFL VNO,04,1
      BUN ST
MAC   CAD 2 0
      BFA 4 A+,01,0
      BFA 4 B+,91,0
      BFA 4 C+,81,0
      BFA 4 D+,71,0
      BFA 3 0,61,1
      CAD +10000
      BUN 4 E+
*D    ADD +889
      BUN 4 E+

*C    ADD +89
      BUN4 E+
*B    ADD +9
      BUN4 E+
*A    ADD +1
*E    STA 2 0
      BUN6 0,73
PS3   LDR LINE1
      STR LINE
      BUN RE1
*V    ADD VNO
      BUN RE
PS4   CAD LOCN
      SUB EXIT1
      SUB +1
      BFA A+

      CAD LOCN
      LDB EXIT1
      -STA 0,04
      BUN RE1
*A    DFL LOCN,00,1
      BUN RE1

PS5   CAD EXIT1
      ADD +440000
      BUN RE
SPEC  LSA 0
      ADD EXIT2
      BUN RE
PS6   CAD EXIT2
      ADD +30 0000
      BUN RE
PS8   LDR EXIT1
      STR EXIT2
      CAD LOCN
      STA EXIT1   STA EXIT3,04
      ADD +30 0000
      BUN RE
PS7   CAD I
      MUL +5
      SLT 10
      ADD +WIWJ
      STA TEMP1
      ADD J
      ADD +40 0000
      LDB LOCN
      -STA 0
      IFL LOCN,00,1
```

BOOKKEEPING MACRO

```
STA WIWJ+5I+J
ADD WIWJ+5I+tk
STA T
EXT T
BZA E(n-1) or Prog(n).
STA WIB+5b+I
```

$b_0 = w_j + 1$

```
CAD WIWJ+5I+J
ADD WIWJ+5tk+J
STA T
EXT T
BZA End or Prog n
STA BWJ+5b+J
```

$b_0 = W_i + 1$

checking macro

```
CAD BWJ+5b+j
ADD BWJ+5b+tk
STA T
EXT T
BZA End or Prog n
```

$b_0 = w_j + 1$

retest for all t_k, and first line for all b.

```
CAD WIB+5b+i
ADD WIB+5b+tk
STA T
EXT T
BZA (En-1) or Prog n
```

En BUN Prog (n+1).

720 μs on 220
10.9 μs on 7090

```
LOB J
-LDR BST
STR B
STP TINTX
BUN TINIT, M+
```

```
*L  STP GETX
    BUN GETT
    ADD TEMP1
    STA FILL,04
    LOB LOCN
    RTF FILL,4
    CAD +40 0000 + WIB
    ADD B
    ADD I
    -STA 0
    IFL LOCN,00,5
    IFL B,00,5
    BUN L-
```

```
*M  CAD TEMP1
    ADD J
    ADD +10 0000
    LOB LOCN
    -STA 0
    IFL LOCN,00,1
    LOB I
    -LDR BST
    STR B
    STP TINTX
    BUN TINIT, CHECK
```

CAD +WIWJ
ADD J
STA TEMP1

```
*L  STP GETX
    BUN GETT
    MUL +5
    SLT 10
    ADD TEMP1
    STA FILL,04
    LOB LOCN
    RTF FILL,4
    CAD +40 0000 +BWJ
    ADD B
    ADD J
    -STA 0
    IFL LOCN,00,5
    IFL B,00,5
    BUN L-
```

```
TINT  LDR +6969696969
      STR-T
      DBB *+1,1
      CLL -T
      DBB *-1,1
      LDB TINTX
      DLB -9999,44,0
      STB LOOPX,04
TINTX BUN *
```

```
GETT  LOB +1
      LDR -T
      BZR A+
LOOPX BFR *,00,69
      CLL -T
      IBB GETT+1,1
*A    IFL -T,00,1
      STB T
      CAD T
GETX  BUN *
```

```
BST   CNST 0,10,15,25,45,85,165[52^n+1]
FILL  ADD *
      STA T
      EXT T
EXIT3 BZA *
```

I
J

rB = 1 we want 1
rB = 2 1
rB = 3 1,2,1
rB = 4 1,2,1,3,2,1

0000
0001
0011
0010
0110
0111
0101
0100

HANO STOCK FORM 14111

LDB J
STP TIMTX
RUN TIMIT, M+

CHECK LDB I
-CAD BST
STA B
ADD +100000+BWJ
STA TEMP1
*R ADD J
LDB LOCN
-STA 0
IFL LOCN,00,1
*L STP GETX
BUN GETT
ADD TEMP1
STA FILL,04
LDB LOCN
RTF FILL,4
STB LOCN
BUN L-
*M IFL B,00,5 IFL TEMP1,00,5
LDR BSTOP
CFR B
BCU R-
LDB J
-CAD BST+1
STA BSTOP
-CAD BST
STA B
ADD +100000+WIB
STA TEMP1
*R ADD I
LDB LOCN
-STA 0
IFL LOCN,00,1
LDB I
STP TIMTX
BUN TIMIT, M+
*L, *M exactly as before.

BCU R-
BUN PS 82.

```
                BUN 6 NEXT
PRG1    CAD +1          ——  BUN 6 PSI, 1
        BUN 6 BOOK          BUN 6 PSJ, 1
        BUN 6 NEXT
PRG2    CAD +10
        BUN 6 IJ 12
        ~~BUN 6 PSJ 2~~
        BUN 6 BOOK
        BUN 6 NEXT
PRG3    CAD +100
                1,3 etc
        +1000
                1,4 etc
        +10000
                1,5 etc
PRG6    CAD +10
                2,1 etc
PRG7    CAD +100
                3,1 etc
PRG8    CAD +1000
                4,1 etc
PRG9    CAD + 10000
                5,1 etc
PRG10   CAD +100
                2,2
PRG11   CAD +11
                2,3
PRG12   CAD +10000
        BUN6 IJ,24
        BUN6 BOOK  <-- BUN 6 STP, BUN 6 BUN
        BUN6 NEXT
PRG13   CAD +11
        BUN 6 STP
        BUN 4 A+
        CAD +101
        BUN 6 STP
        BUN 4 A+
        HLT 1212    END OF ROUTINE
*A      BUN 6 IJ,32
        BUN 6 BOOK
```

```
                BUN 6 STP
                BUN 6 BUN
                BUN 6 NEXT
PRG14   CAD +1000
                BUN 6 STP
                BUN 4 A+
                BUN 4 PRG14-2
*A              BUN 6 IJ,33
                BUN 6 BOOK
                BUN 6 STP
                BUN 6 BUN
                BUN 6 NEXT
PRG15   CAD +101
                BUN 6 STP
                BUN 4 A+
                CAD +1001
                BUN 6 STP
                BUN 4 A+
                BUN 4 PRG15-2
*A              BUN 6 IJ,34
                BUN 6 NEXT
PRG16   BUN 6 ADD
                BUN 6 IJ,25
                BUN 6 NEXT
PRG17   BUN 6 ADD
                BUN 6 IJ,35
                BUN 6 NEXT
PRG18   BUN 6 ADD
                BUN 6 IJ,42
                BUN 6 STP
                BUN 6 BUN
                BUN 6 NEXT
PRG19   CAD +101
                BUN 6 STP
                BUN 4 A+
                CAD +1001
                BUN 6 STP
                BUN 4 A+
                BUN 4 PRG19-2
*A              BUN 6 IJ,43
                BUN 6 NEXT
PRG20   BUN 6 ADD etc etc
```

PROG	WI	WJ	
1	1	1	00001
2	1	2	00010
3	1	3	00100
4	1	4	01000
5	1	5	10000
6	2	1	00010
7	3	1	00100
8	4	1	01000
9	5	1	10000
10	2	2	00100
11	2	3	00011
12	2	4	10000
13	2	5	
14	3	3	
15	3	4	
16	2	5	
17	3	5	
18	4	2	
19	4	3	
20	5	2	
21	5	3	
22	4	4	
23	5	4	
24	4	5	
25	5	5	

```
PRG26   CWR WIWJ+30, 22
        BUN 6 BUN
```

2515

```
        BUN 6 NXT
PRG5    CAD +1
1       BUN 6 IJ, 11

        BUN 6 NXT
PRG9    CAD +1000
        BUN 6 IJ, 23
        BUN 6 STP
        BUN 6 BUN
        BUN 6 NXT
PRG10   CAD +11
        BUN 6 STP
        BUN 4 E+
        CAD +101
        BUN 6 STP
        BUN 4 E+
        CAD +1001
        BUN 6 STP
        BUN 4 E+
        CAD +1111
        BUN 6 STP
        BUN 4 E+
        HLT 1212
4E      BUN 6 IJ, 24
        BUN 6 NXT
PRG11   BUN 6 ADD
        BUN 6 IJ, 32

PRG16   BUN 6 ADD
        BUN 6 IJ, 44
        BUN 6 NEXT
PRG26   CMR  WINJ+30, 22
        BUN 6 BUN
```

PRG 6	I	J	ans
1	1	1	1
2	1	2	10
3	1	3	100
4	1	4	1000
5	2	1	10
6	3	1	100
7	4	1	1000
8	2	2	100
9	2	3	1000
10	2	4	
11	3	3	
12	3	4	
13	4	4	
14	2	3	
15	3	4	
16	4	4	

0011
0101
1001
1111

1	1	1
2	1	2
3	1	3
4	1	4
5	1	5
6	2	1
7	3	1
8	4	1
9	5	1
10	2	2
11	2	3
12	2	4
13	2	5
14	3	2
15	3	3
16	3	4
17	3	5
18	4	2
19	4	3
20	5	2
21	5	3
22		
23		
24		
25		

01000
10000

etc

WIB + 16 b=3 i=1

WIB 8500
BWJ 8700

BWJ + 12 b=2 j=2

8900 WIWJ

1	10	100	1000
10	100	1000	
100	001		
1000			

8516 8526

1 10 11 101

x^5+x+1
x^5+x^2+1
x^5+x^3+1
x^5+x^4+1

$(x+1)^2 = x^2 + 1$
$(x+1)^3 = x^3+x^2+x+1$
$(x+1)^4 = x^4 + 1$
$(x+1)^5 = x^5 + x^4 + x + 1$

$x^5+x+1=0 \Rightarrow (x+1)^5+(x+1)^4+1=0$
$x^5+x^2+1=0 \Rightarrow (x+1)^5+(x+1)^4+(x+1)^2+(x+1)+1$
$x^5+x^3+1=0 \Rightarrow (x+1)^5+(x+1)^4+(x+1)^3+(x+1)^2+1$
$x^5+x^4+1=0 \Rightarrow (x+1)^5+(x+1)^3+1$

~~x^4~~ $+x^3+x+1$
x^2
~~x^5~~ less x $(x+1)^5+(x+1)^3+(x+1)^2+(x+1)+1$
~~x^4~~ x^3+x^2+x $(x+1)^5+(x+1)^3+1$
less x^2 x^3 $(x+1)^5+(x+1)^2+1$
less x^3 x^2 $(x+1)^4+(x+1)^3+(x+1)+1$
less x^4 $x^4+x^3+x^2$

$x^5+x+1 \longleftrightarrow x^5+x^4+1$
$x^5+x^2+1 \longleftrightarrow x^5+x^4+x^2+x+1$
$x^5+x^3+1 \longleftrightarrow x^5+x^4+x^3+x^2+1$
$x^5+x^4+x^3+x+1 \longleftrightarrow x^5+x^3+x^2+x+1$

8716 3,2
8716 3,1

8722 4,2

HANO STOCK FORM 1

if $x^2x = x+1$,

$(x^2+1)(x+x^2+x^2x^2) = x+1 + x^2x^2 + x^2(x^2x^2) + x + x^2 + x^2x^2$

$= 1 + x^2 + x^2(x^2x^2)$.

$(x^2+1)(x^2+x^2x^2) = x^2x^2 + x^2(x^2x^2) + x^2 + x^2x^2$

x^2+1

$(x^2+1)(x+x^2x^2) = x+1 + x^2(x^2x^2) + x + x^2x^2$. contradiction.

if $x^2x = x^2+1$

$x^2(x+x^2x^2) = x^2+1 + x^2(x^2x^2)$.

$x^2(1+x+x^2+x^2x^2) = x^2+1+x^2x^2 + x^2(x^2x^2)$

sloppy!!

$$(\Sigma a_i x^i)(\Sigma b_j x^j) = \Sigma a_i b_j c_{ijk} x^k$$

```
SUB   LDB +3
      CLL -B
      DRB *-1,1
      LDB +4
      CLL -A
      DRB *-1,1      ~~LDR~~ CLL SUM
      ~~LDR +4~~
      STR I
```

~~*A LDB =0~~
~~STR J~~
~~LDB I~~
~~CAD A~~
~~LDR J~~
~~MUL B~~

```
*D    LDB I
      CAD -A
      BZA A+
      LDR +4
      STR J
*C    LDB J
      CAD -B
      BZA B+
      CAD I
      MUL +5
      $LT 16
      STA *+1,44
      IRB *+1,0
      CAD -TABLE
      ADD SUM
      STA SUM
      EXT SUM
      STA SUM
*B    DFL J,00,1
      BRP C-
*A    DFL I,00,1
      BRP D-
      LDB L
      LDR SUM
      STR BUFFR
      LDR B
      BZR A+
      LDR B+1
      BZR B+
      CLL B+1
      LDR B+2
      BZR C+
      CLL B+2
      LDR B+3
      BZR D+
```

```
      CWF *-1,22
      CLL B+4
      CLL L
      STP SUBX
      BUN SUB
      HLT 1
      $FL B+4,00,1
      STP SUBX
      BUN SUB
      HLT 2
TABLE CNST 1,10,100,1000,10000, etc.
```

```
      CLL B+3
      CWR BUFFR+15,22
      LDR A
      BZR AA+
      CLL A
      LDR A+1
      BZR BB+
      CLL A+1
      LDR A+2
      BZR CC+
      CLL A+2
      LDR A+3
      BZR DD+
      CLL A+3
      LDR A+4
      BZR EE+
      CLL A+4
SUBX  BUN *
*A    IFL B,00,1
      BUN LOOP
*B    IFL B+1,00,1
      BUN LOOP
*C    IFL B+2,00,1
      BUN LOOP
```

Breakpoint digits in the compiler program
Breakpoint 8 on CWR commands for unit 3.
Breakpoint 4 not used – reserved for insertion in special places when debugging
Breakpoint 2 stops on 12: rA contains a compiled instruction
rB contains "LOOPL"
64: rA contains AØ – beginning of SCAN routine
28: Beginning of subgenerator specified in Control Counter
14: Forward reference entry in rA

Breakpoint 1 – same as 2 plus 66: next character read in
30: end of GET, ACALL, BCALL, MASTR, CONOT

NXTCH subroutine. This routine is the only communication link between the compiler and the input pseudo-code. Two versions are necessary, one for Cardatron input and one for ~~Cardatron output~~. The routine also is part of the Error routine as it translates the input to modified Flexowriter code when that character is not on the Flexowriter.

Entry: rA = opl instruction. rB = number; this number will be stored into Q and the next character will go into A(2+Q)

Exit: rA[LSTCH] and A(2+Q) contain next character. Special characters are changed to special code. rP = Q

Temp storage used: NEXIT, WORD, Loop6

Stop 888 if 101st record over 100 words in length is read.
000
000
000 if invalid alphanumeric character is sensed.

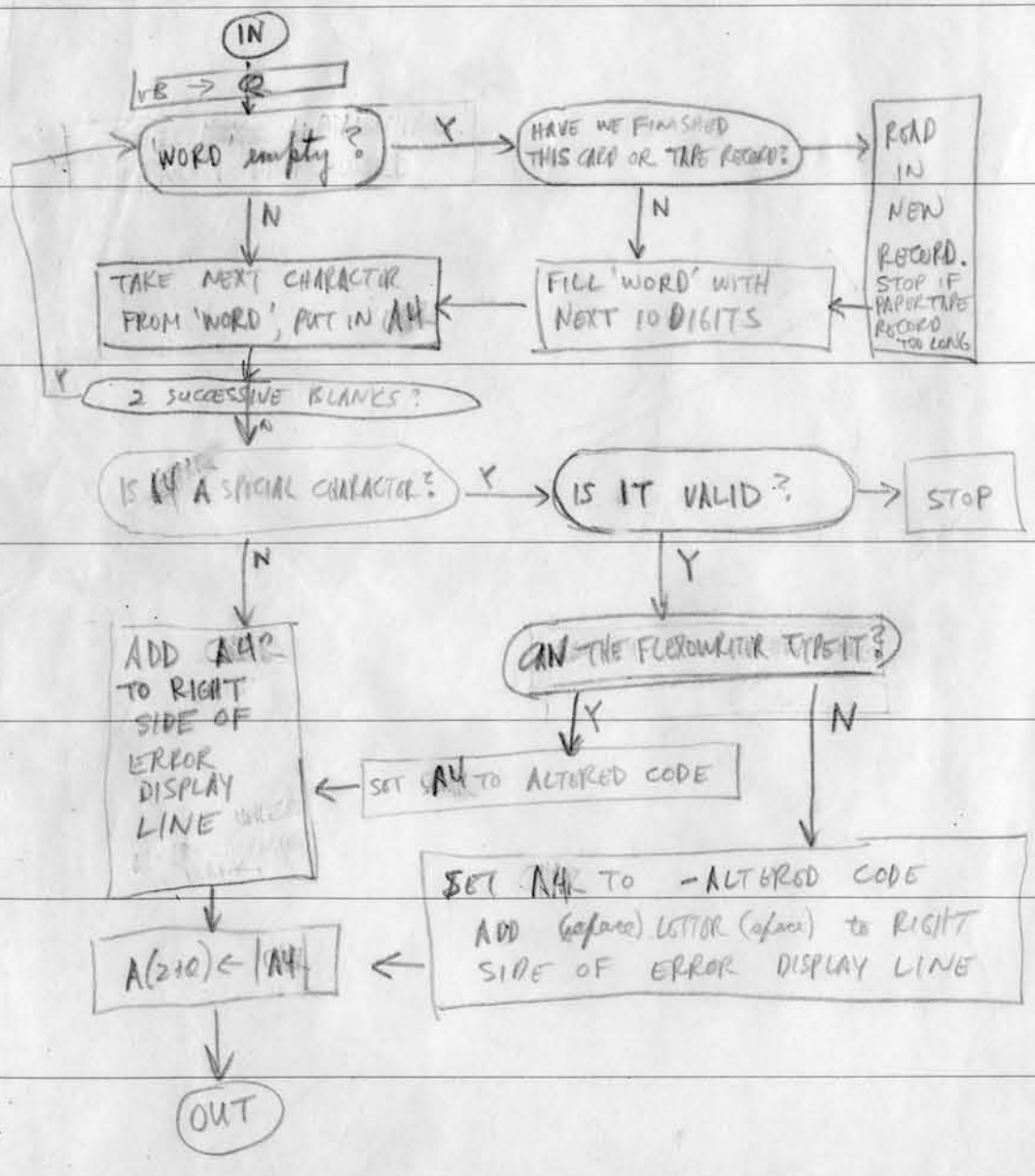
```
NXTCH  STA  NEXIT
       BA
       STA  Q
       CAD  WORD
4      SRT  8
       STA  14
7      ADD  3F 7
       CCB  1F
       SLT  10
       STA  WORD
       SLT  10
       ADD  5F 7
       CCB  8F
       CUB  9F
3 = 9  901 =
5 = 9  958 =
1      LDB  Z          CARDS
       DBB  1F 7
       LDB  2F 7
10     CRD  6000, BF6 0
       BF6  0
1      CAD  =99= 3F 7
       SRT  2
2      BA   15
       STA  Z
       -CAD 0000 = LDB Q
       CUB  4B
3      =99=
       Z initially zero
5      LDB  6F 7       PTAPE
1      LDB  Z          CAD 101
2      -IB  19         OSD 6F 7
3      BA   5B         BOF 3F 7
       STA  Z          HLT 20
       -CAD 0
       LDB  Q
       OSD  2B 7
       CCB  4B
       LDB  2B 7
       -STA 6000
       DBB  1- 7
6      BF6  0
       BF6  40
       BF6  20
       LDB  3B 7
       PTR  0
       Z initially 100, 101 initially negative
```

```
       CNZ  2+ 7
8      CUB  6F
7
       ADD LSTCH
       CNZ 4+ 7
6      SLT  2
       CNZ 3+ 7
       BUN 2- 7
       STC 6B 7
       CUB 7B
```

```
9      LDB  A4
       -CAD TRNS
       CNZ  2+ 7
       HLT  104 — SRT 2
       STA  14
       CAD  14          BUN 2+ 7
       SFT  2
       -CAD A+          BT6 CURNT
       OSD  8+ 7        LDB CURNT 6
       CCB  8F
       SFT  2
       -CAD 6000 (CURNT+2) 6
       SLT  2
       -STA 6000 (CURNT+1) 6
       SLT  8
       DBB  4- 7
       CUB  9F
       LDA  A4
       STA  A2
       -1 = -3
8      SRT  6
       STA  A4
       -CAD 6000 (CURNT+3) 6
       SLT  6
       -STA 6000 (CURNT+2) 6
       SLT  4
       DBB  4- 7
       CUB  9B
9      BF6  CURNT
       LDB  Q
       CAA  A4
       -STA A2 — STA LSTCH
NEXIT  HLT  8888
```

TRANS + n (n = 0, ..., 40)
is zero if n is illegal
is 0 0 on card if legal and
and typable. — on card 00 00 if
legal and untypable.

CURNT = constant of 3



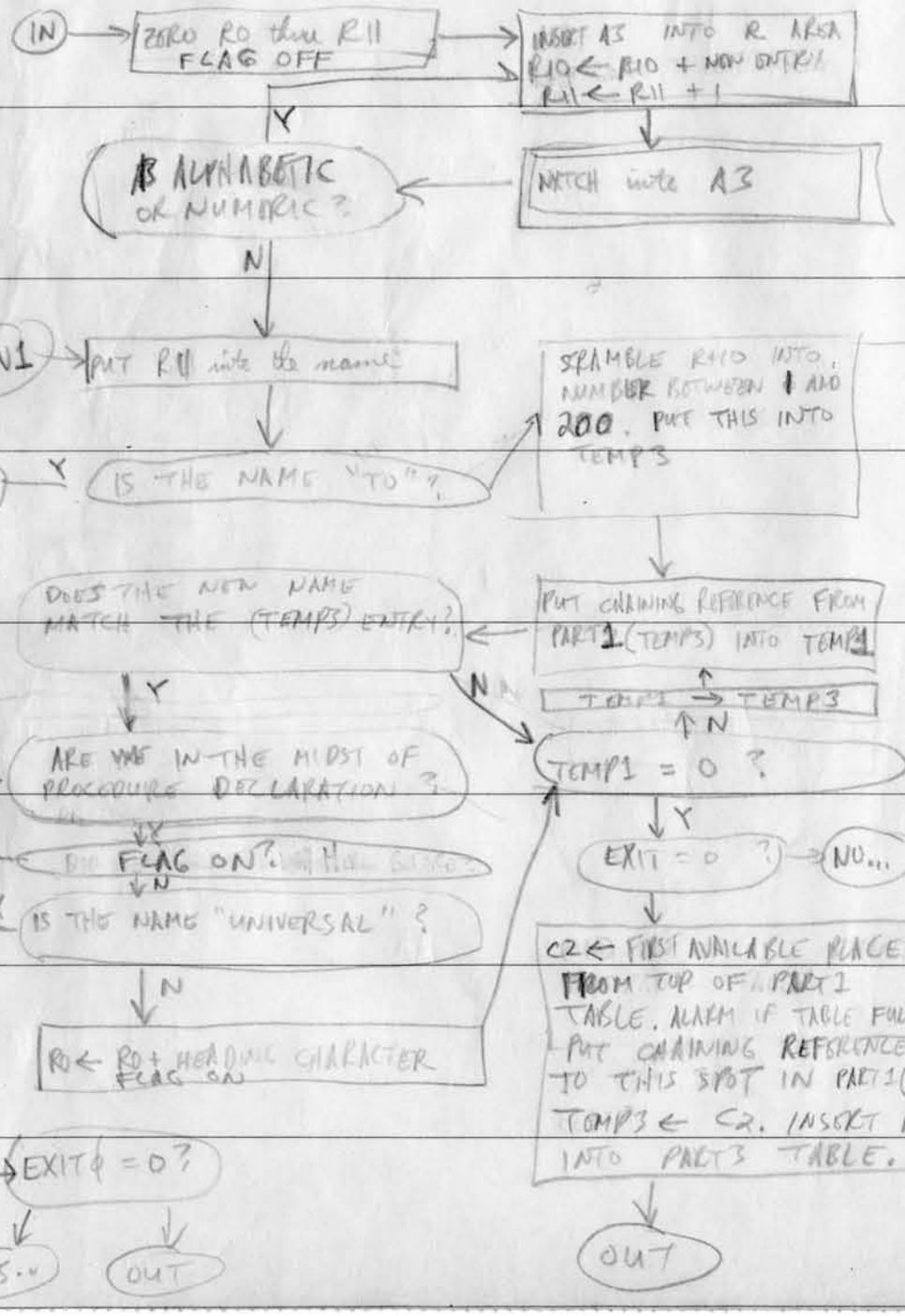
BUILD This subroutine builds a name and converts it into a number between 0 and 299. An n-character name is stored in $\left[\frac{n+6}{5}\right]$ locations in the table PART3 which has room for 500 names.

Entry: A3 contains the first character. rA contains the exit. A special entrance is used for looking up a prefix, and EXIT = zero.

Exit: If exit is zero, it determines whether or not the name as a prefix was in the table, and exits to NO... or YES... Otherwise the name is inserted into the table if not already there and the equivalent is put into TEMP3

Temp storage used EXITϕ TEMP1 TEMP2 TEMP3 TEMP4 TEMP5 R0 thru R11 C2 C1

Subroutines used NXTCH




```
BUILD  STC EXIT$\phi$
       LDB  1F  7
      -STA  R
       DBB  1-  7
       CAD  3F  7
6      STA  TEMP1
       SRT  3
       STA  TEMP2
       LDB  TEMP1
       CAD  A3
1      CLR  11
      -SLT  98
       LDB  TEMP2
      -ADD  R
      -STA  R
       ADD  R+10
       CCB  2F
       CUB  2F
3      =606=
2      STA  R+10
       CAD  R+11
       ADD  ONE
       STA  R+11
       CAD  2F  7
       LDB  ONE
       CUB  NXTCH
3      ADD  4F  7
       BOF  4+  7
       CUB  1F
       CAD  TEMP1
       EXT  SF
       CUB  6B
2      CUB  3B
4      9    780
5      =1303=
1      CAD  R+11
       ADD  R
BLD1   STA  R
       SUB  2F  7
       CNZ  2+  7
       CUB  PRSC1
       LDB  TEMP2
      -CSA  R
      -STA  R
       CAD  3F  7
       MUL  R+10
       STC  1B  7
       SLT  5
       MUL  4F  7
       CUB  1F
```

ADD ONE

STC EXITϕ / LDB 2F 7 / STA R+1 / DBB 1-7 / LDB R+12 / CAD 3F 7 / SRT 10 / CAD A3 / SLT 8 / STA R / IB / ADD R+10 / CCB 2F / CUB 2F / NG. / STA R+10 / CAD R+11 / ADD ONE / STC R+11 / SLT 2 / NOR 1F 7 / IB / BA / STA R+12 / CAA PROCT / STA TEMP5 / LDB R+2 / CAD R / CUB 6B

```
1      STA  TEMP3
       LDB  TEMP3
      -CSA  PART1 -1
       STA  TEMP1
2      SRT  7
       ADD  2F  7
       STA  2F  7
       LDB  2F  7
3     -CAA  R-7
2     -SUA  PART3 -7-500
       CNZ  4F  7
      -OSD  R-7
       CCB  2F
5      IB
       BUN  3B  7
4      LDB  TEMP1
       DBB  2+  7
       CUB  8F
       CAD  TEMP1
       CUB  1B
2      CAA  TEMP5
       CNZ  4+7
3      CAD  EXIT$\phi$
       NOR  YES..
       BUN  EXIT$\phi$
       ADA  R
       STA  R
       LDB  TEMP5
      -OSD  PART1
       BOF  3B  7
       LDB  TEMP2
      -CSA  R
      -STC  R
       CUB  4B
8      LDB  C2  NOR NO...
2      DBB  2+  7
3      CUBR ALARM
      -CAD  PART1
       CNZ  2B  7
       BA
       STA  C2
       LDB  TEMP3
       ADD  PART1
      -STA  PART1
       CAD  C1
       LDB  C2
       SLT  7
      -STA  PART1
       CUB  1+
```

STA TEMP5 / CNZ 2+7 / BUN 3B 7

CUB 8B

```
1      CAD  C1  7
       ADD  TEMP2
       ADD  3F  7
       CCB  ALARM
       [redacted]
       LDB  TEMP2
       CAA  R
      -STA  PART3-500
       DBB  2-  7
       BUN  EXIT$\phi$
3      =9   9001=
```

Layout of part 1 table:

i XXX ... XXX

part 3 ref chaining ref

C1 initially 500

C2 initially 300 goes down

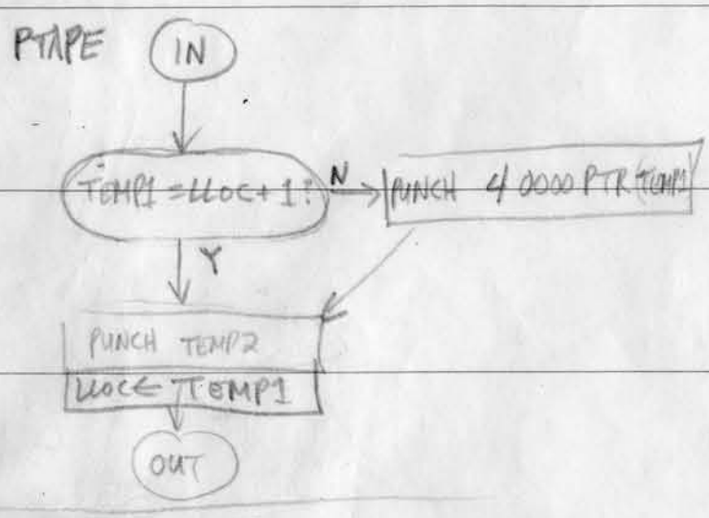
All of the part 3 table is available for initialization

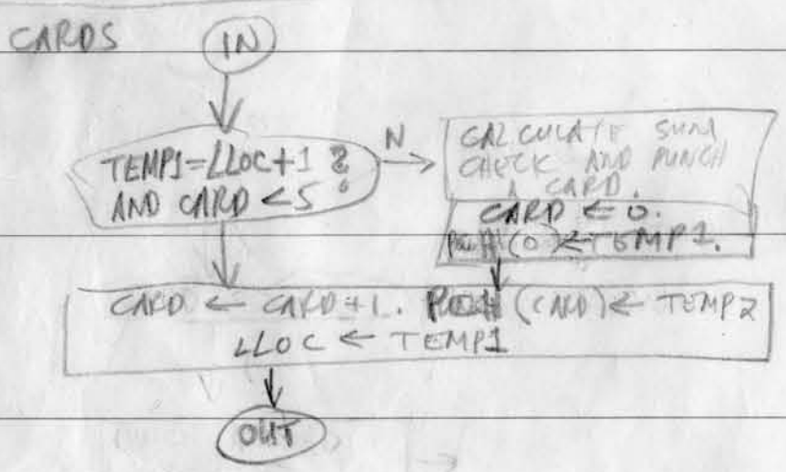
```
       CAD  11
       SUB  procedure to
       CNZ  EXIT$\phi$
       CUB  1F
1      CAD  PROCC
       ADD  100=
       STA  PROCC
       STA  PROCT
       ADA  R
       SRT  10
       ADD  R
       SLT  10
       IB
5      -STA  PART3-500
       LDB  TEMP3
       CAD  =3451
       STA  PART2
```

2 = 5663 0002; 3 = 1001 0110 01=, 4 = 0020 0 =

PUNCH subroutine. This is the only communication between the compiler and the punching of the output. Two routines are necessary.

Entrance: EXIT in rA. TEMP1 = location. TEMP2 = instruction.
Temp storage used: PEXIT, LLOC, TEMP1, TEMP2, CARD, PCH (0)-(4) Comp 6.






```
PNCH1   STA  9+  7
        CAD  PROFL
        SUB  ONE
        STA  PROFL
        STA  TEMP1
        CAD  4J
        SUB  PUNCH
```

```
        CAD  OKUML
        ADD  LOOPL
        SLT
        STA  TEMP2
```

```
PTAPE:  PUNCH  STA  1ST  7
               CAD  TEMP1
               SUB  LLOC
               SUB  ONE
               NOR  1F   7
               CAD  TEMP1
               SRT  4
               CAD  2+   7
               CIRA 9
               SLT  4
               PTW  10
          1    CAD  TEMP2
               PTW  10
               CAD  TEMP1
               STA  LLOC
               HLT  (exit)  not needed
```

```
CARDS   PUNCH  STA  SF   7
               CAD  TEMP1
               SUB  LLOC
               SUB  ONE
               CNZ  7F   6     BT6 7F
               CAD  CARD
               ADD  4F   7
               BOF  7F   6
          3    LDB  CARD
          6    IB   5
               BA
               STA  CARD
               CAD  TEMP2
               STA  PCH
               CAD  TEMP1
               STA  LLOC       38CWR TEMP1
          5    HLT
          1    BT6  7F
               BUN  7F   6
          4    9    95
```

```
          7    CSU  CARD
               SAB  1F   6
               STA  (PCH+7) 6
               LDB  6B   7
               SUB  PCH  6
               BOF  1+   6
               DBB  2-   6
               STC  (PCH+6) 6
               STA  CARD
         10    CWR  PCH  6
               CAD  TEMP1
               STA  PCH
               BUN  3B   7
```

```
               CAD  8F   6
               CLR
               SLT  8
               ADD  9F   6
               STA  8F   6
         10    CWR  PCH  6
               CAD  9F   6
               EXT  = 111113000 0
               ADD  TEMP1
               STC  9F
               LDB  ONE
               BU
```

PCH initially set for loading routine

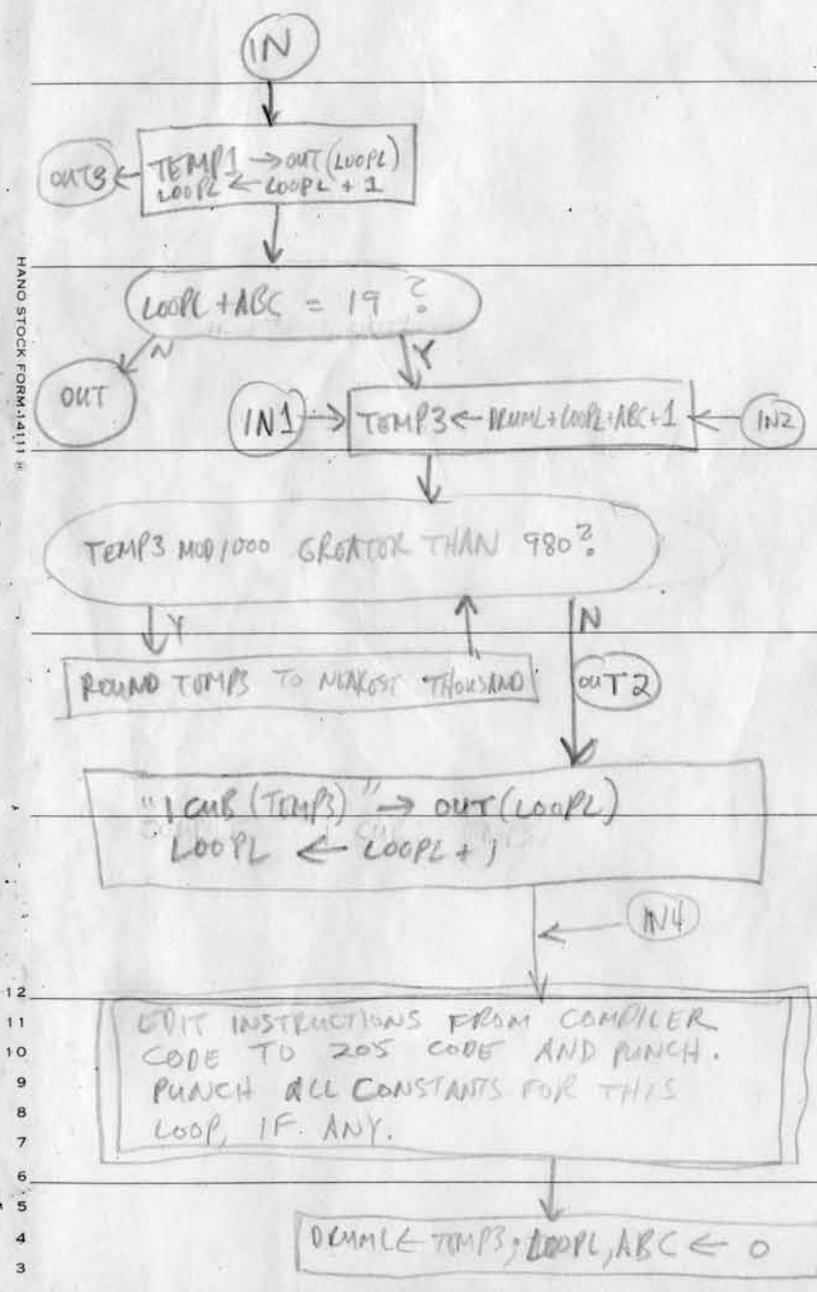
OUTPT subroutine. This is the normal outlet for compiled instructions, but it has several special entrances which are used for unusual effects. It stores instructions away until Augt=read-loop 7 is filled with output program; then it punches them out.

Entry: normal entry — TEMP1 = instruction, rA = exit.
entry OUTP1 — rA = exit. Makes it think loop 7 is full.
entry OUTP2 — rA = exit. Makes it calculate address of next loopfull only
entry OUTP3 — Same as normal except it omits check for full loop.
entry OUTP4 — rA = exit. Punches the loop only.

entry OUTP5 — same as OUTPT.
OUTP7 combines OUTP3 and OUTP2 and OUTP4

Temp Storage: EXITϕ, LOOPL, DRUML, ABC, LLOC, OUT(0)-(19), TEMP1, TEMP2, TEMP3, TEMP4

Subroutines used: PUNCH.




```
OUTP7  STA EXIT$
       CAD 3F 7
       STA 3F 7
OUTP3  STA 3F 7
       BUN 2+ 7
OUTPT  STA EXIT0
OUTP5  LDB LOOPL
       CAA TEMP1
       STA OUT
4      IB
       BA
       STA LOOPL
1      ADD ABC
       ADD 2F 7
       CCB 1F
       BUN EXIT$
```

```
OUTP4  STC EXIT$
3      STA TEMP4
4      LDB TEMP4
       CAD OUT
       SRT 4
       STC TEMP1
       SLT 1
       ADD 3F 7
       CCB 1F
       SUB 3F 7
5      SRT 1
       CAD TEMP1
       CUB 2F
3      =9 93[illegible]
1      =999990-0=
       CNZ 5F 7
       SLT 3
       ADD TEMP4
6      ADD DRUML
       SRT 3
       CAD =7=
       CUB 5B
5      SLT 3
       ADD LOOPL
       BUN 6B 7
8      -CSU OPTBL
9      CAD OPTBL
       SLT 4
       STA TEMP2
       CAD DRUML
       ADD TEMP4
       STA TEMP1
       CAD 1F 7
       CUB PUNCH
8      CAD TEMP4
       ADD ONE
       STA TEMP4
       SUB LOOPL
       CNZ 2+ 7
       CUB 8F
       CUB 4B
1      CUB 8B
```

```
       LDB TEMP1
       ADD 4F 7
       CCB 8F
       -CAD 8PTR1
       CUB 9F
7      CAD TEMP4
       ADD ONE
8      STA TEMP4
       SUB ABC
       CNZ 1F 7
       CAA TEMP3
       STC DRUML
       STA LOOPL
       STA ABC
       BUN EXIT$
1      LDB TEMP4
       -CAD CON
       STA TEMP2
       CAD LLOC
       ADD ONE
       STA TEMP1
       CAD 2+ 7
       CUB PUNCH
       CUB 7B
```

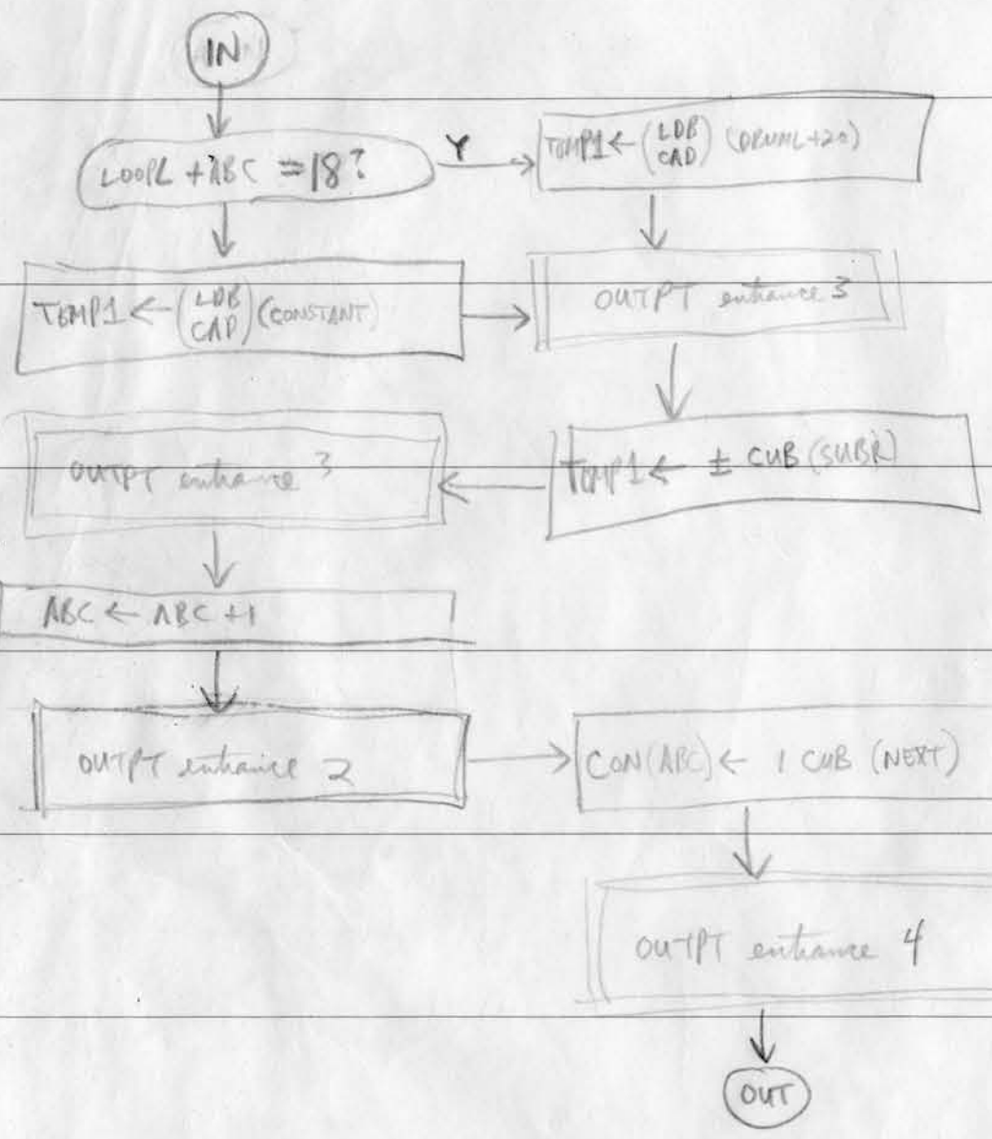
```
       =9 9019=
       19F
6      CUB 3F 8 CUB 7F
2      LDB LOOPL
       CAA TEMP3
       ADD 2F 7
       -STA OUT
7      IB
       BA
       STC LOOPL
       CUB 3F
2      (1 CUB)
```

ACALL, BCALL subroutines. Compile the instructions to call out a subroutine with exit in A or B register, respectively.

Entry: rB = exit. rA = address of subroutine. ~~rA is put in rB to modify the CUB instruction~~

Temp. Storage: EXIT1 TEMP1 – TEMP4 RB

Subroutines Used: OUTPT




```
ACALL STA  TEMP4
      CAD  4F  7
      STC  5F  7
      BUN  2+  7
BCALL STC  TEMP4
      CAD  LOOPL
      ADD  ABC
      ADD  6F  7
      BOF  1F  7
      CAD  ABC
      ...
3     ADD  5F  7
      CUB  2F
1     CAD  DRUML
      SUB  8F  7
      BUN  3B  7
4       (CAD) 8000
5       (LDB) 8000
6     = 9 ——— 982 =
7     = 7980 =
-------------------
2     STA  TEMP1
      BA
      STA  EXIT1      ADD 6F 7
      CAD  2F  7
      CUB  OUTP3
3     CAD  ABC
      ADD  ONE
      STA  ABC
      CAD  TEMP4
      ~~CF~~
      ADD  5F  7
      STA  TEMP1
      CAD  3F  7
      CUB  OUTP3
2     CUB  3B      5 = (CUB 6 =
1     ~~10~~ =     5 = (1 CUB) =
3     CUB  1+
      CAD  2F  7
      CUB  OUTP2
3     CAA  TEMP3
      ADD  (1 CUB)
      LDB  ABC
      STC  CON-1   STA RB
      CAD  EXIT1   ADD ~~6F~~
      CUB  OUTP4   + ~~CUB~~
```

GET subroutine. Given a table entry from the Operand Stack, this subroutine compiles the coding to bring this operand into the A register, storing the previous A register contents if necessary. This routine is unusual in this compiler in the fact that it uses itself, and occasionally uses MASTR which uses GET. This difficulty is overcome by saving the entry exit information in HOLD. The thing is set up so that GET will only use itself once, and generalized stacking of exits is unnecessary.

Entry: normal rA: code information to get. rB: exit location.

GET1 rA: instruction to substitute for CAD in 'get output'. rB: exit location. TOGET = code information to get.
'Almost dead' Same as GET2 but suppresses the compilation of the final instruction.

Temp. Storage used: CND, CMTR, COLUM, TEMP2, HOLD0 - HOLD6, PVAR, SPEC STOR T, COUNT, TEMP6

sequential: MEXIT, RIGHT, LEFT, OP, GEXIT, TOGET, TODO, MFL

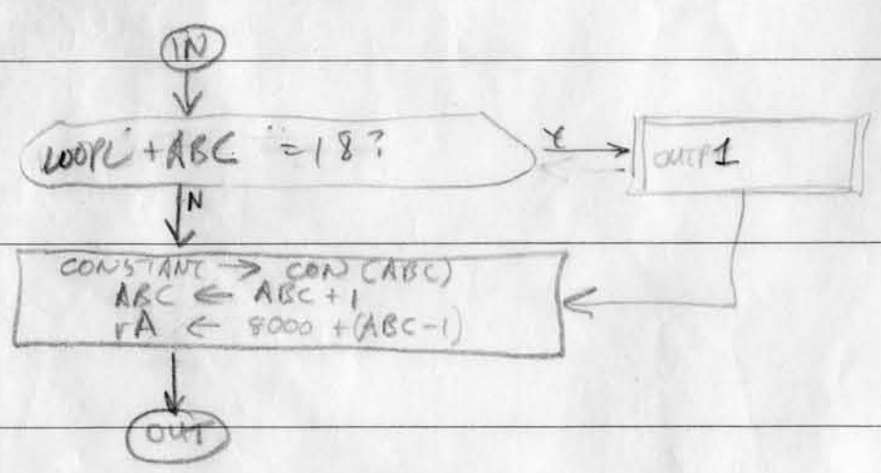
Subroutines used. CONOT, OUTPT, GET, MASTR, STORE, RSULT

continued on next page

CONOT subroutine. Prepares a constant for compilation.

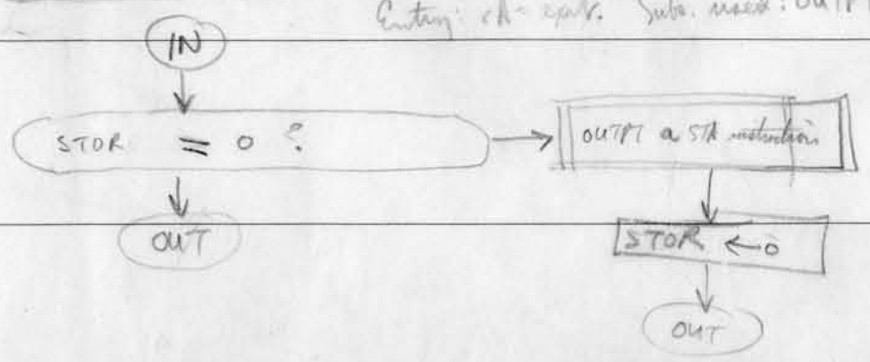
Entry: rA = the constant rB = exit location
Exit: rA = code which will be used to call out this constant
Temp. Storage used EXIT1 TEMP5

Subroutines used. OUTPT

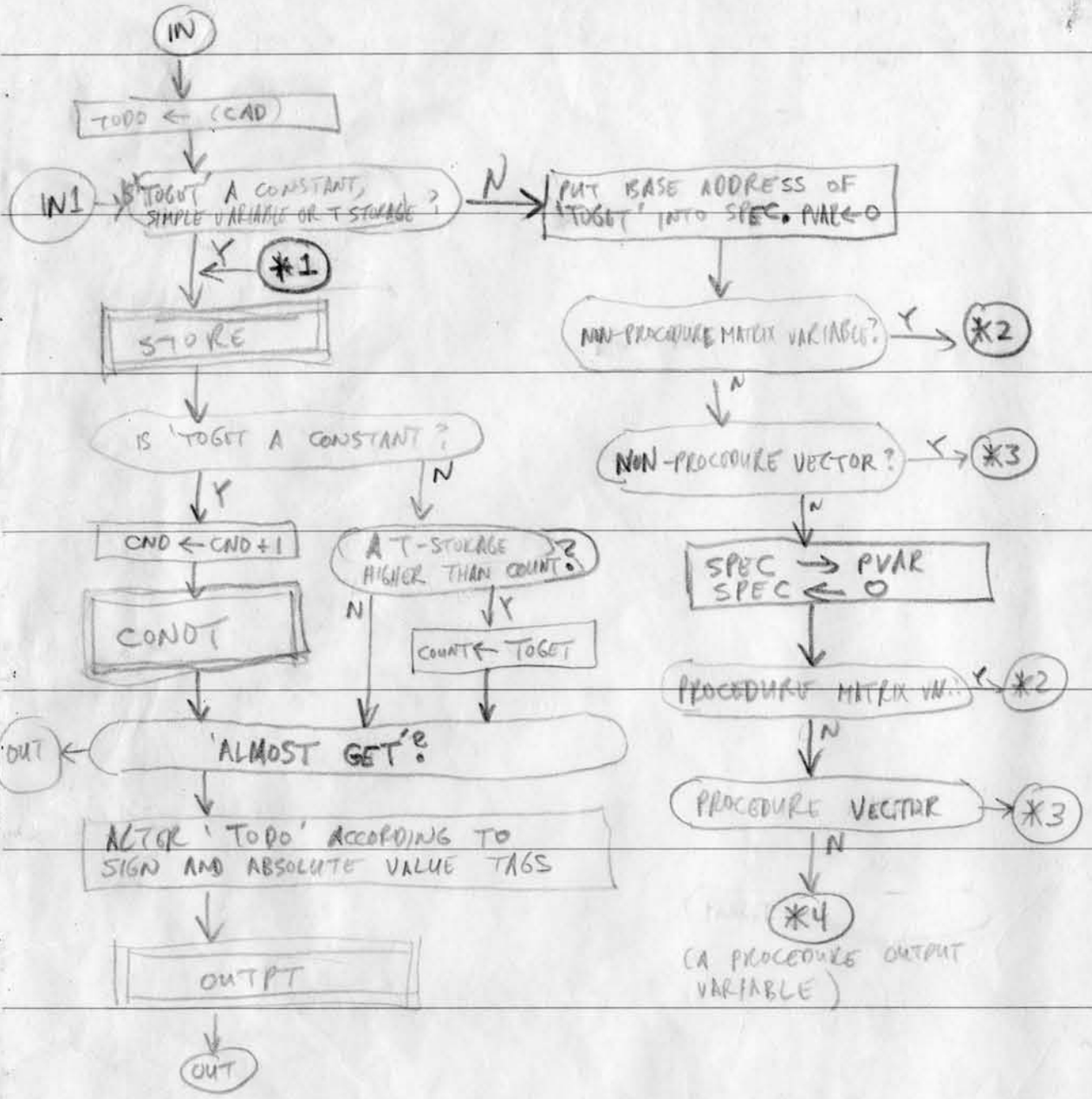



```
CONOT   STA  TEMP5
CNOUT   BA
        STA  EXIT1
        CAD  LOOPL
        ADD  ABC
        ADD  1F 7
        BOF  2+ 7
        CUB  7F
        CAD  1- 7
        CUB  OUTP1
        =9      982=
1/7     LDB  ABC
        CAD  TEMP5
        STA  CON
        IB
        BA
        STA  ABC
        ADD  1F 7
        LDB  EXIT1
        CUB  0
1       =7999=
```

STORE subroutine. Stores rA into location specified by STOR if STOR is positive.
Entry: rA = cplr. Subs. used: OUTPT. Temp storage: EXIT4, STOR.



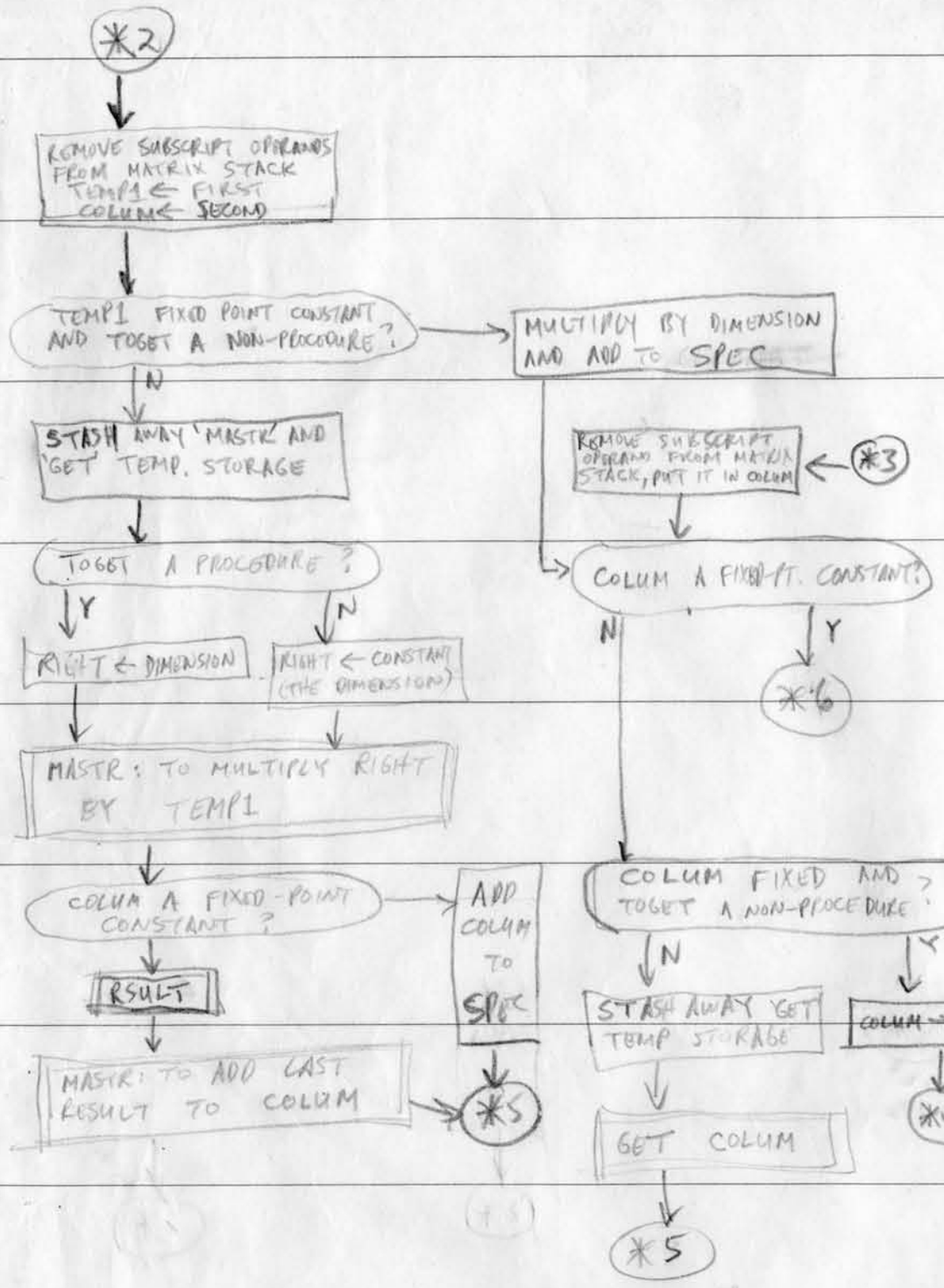
coding is part of GET.




```
GET   STA  TOGET
      CAD   2F  7
GET1  STA  TODO
      BA
      STC  GEXIT
      CAA  TOGET
      ADD   3F  7
      CCB   9F
GET2  CAD   1F  7
STORE STA  EXIT0
      CAD  STOR
3 7000 NOR  EXIT0
      SUB   4F  7
      CUB   2F
      STC  TEMP1
      STA  STOR
      CUB  OUTPS
2     =CAD0=
4   [280 0 - STA]
1     CUB   2F
2     CAA  TOGET
3 7600 ADD   1F  7
      BOF   2F  7
      CAD   CNO
      SUB   ONE
      STA   CNO
      LDB  TOGET
     -CAD  NUMST
      LDB   5F  7
      CUB  CONDT
2     ADD   3B  7
1 9600 CCB   4F
      CUB   2F
5     CUB   3F

4     CAD  TOGET
      EXT   1F  7
      SUB  COUNT
      OSD   1F  7
      CCB   2F
      ADD  COUNT
      STA  COUNT
      CUB   2F
1     =1100001111=
```

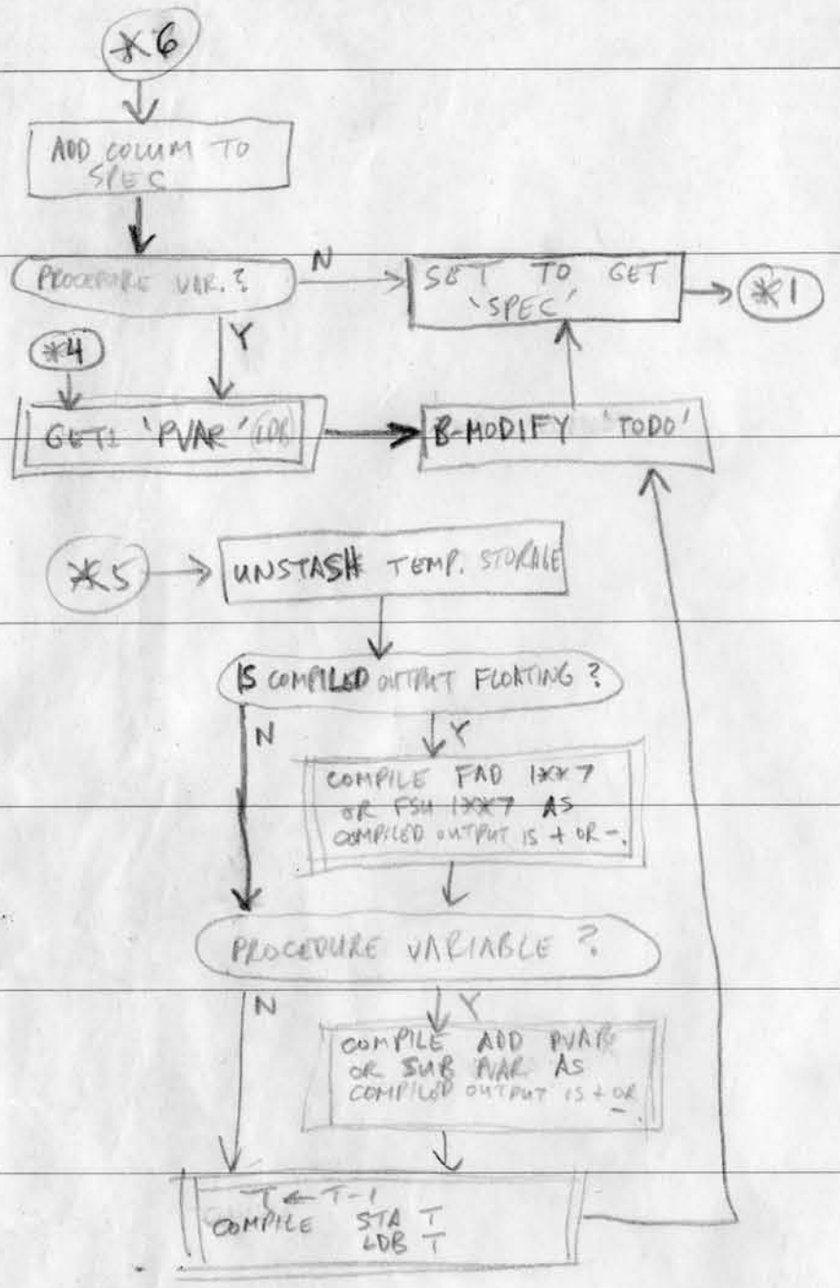
```
2     CAD  TOGET
3     SRT   4
      CAD  TODO
      LDB  GEXIT
      OSD   7+  7
4    -CCB   0
      CAD  TOGET
      EXT   1F  7
      NOR   4+  7
      CAD  TODO
      ADD   2F  7
      STA  TODO
      CAD  TOGET
      EXT   6F  7
      CIRA  0
      CUB   5F

1     =110000=
2       =2=
6     =-0---0=

5     ADD  TODO
      SLT   4
      STC  TEMP1
      CAD   4F  7
      ADD  GEXIT
      CUB  OUTPT
9     SRT   8
      STA  TEMP1
      LDB  TOGET
     -CAD  PART2
      STC   SPEC
      STA   PVAR
      LDB  TEMP1
     -BUN   3-  7   3 CUB 0
      CUB   1F
      CUB  SINGL
      CUB   2F
      CUB   2F
      CUB   2F
2     CAD  SPEC
      EXT   2F  7
      STC   PVAR
      STA   SPEC
      LDB  TEMP1
     -BUN   6-  7
      CUB   1F
      CUB  SINGL
      CUB  LDBR1
```

continued on next page.




```
1    CAD  CMTX
     SUB  2F 7
     STA  CMTX
     LDB  CMTX
-    CAD  MATRX +2
     STA  TEMP1
     SRT  7
     SUB  3F 7      ADD PVAR
     CNZ  2+ 7
     CUB  4F.
-    CAD  MATRX + 1
     STA  COLUM
     CUB  1F
2    =2=
3    =31=
4    CAD  CNO.
     SUB  ONE
     STA  CNO
     CAD  SPEC
     EXT  2F 7
     LDB  TEMP1
-    MUL  NUMST
     SLT  6
     ADD  SPEC
     STA  SPEC
     LDB  CMTX
     ~~MATRX~~
     CUB  SING1
     =1110000=
2    LDB  2F 7
1    ~~ADD NEXT~~
-    ~~STA HOLD~~   BFS HOLD
     DBB  2- 7
     CAD  PVAR
     CNZ  3F 7
     LDB  CNO
     UB
     BA
     STA  CNO
     CUB  1F
2    ADD  ONE
3    STA  RIGHT
     CUB  3F
     ADD  2F 7
1    STA  RIGHT
     CAD  SPEC
     EXT  5F 7
     SRT  4
-    STA  NUMST + 3
     CAD  TEMP1
3    STA  LEFT
     CAD  4F 7
     STA  OP
     CAD  1F 7
     CUB  MASTR
2    =0310 03=
5    =1110000=
4    save for
1    CUB  1+
```

```
     CAD  COLUM
     SRT  7
     SUB  1F 7
     CNZ  2F 7
     CAD  CNO
     SUB  ONE
     STA  CNO
     LDB  COLUM
-    CAD  NUMST
     ADD  SPEC
     STA  SPEC
     CUB  QUER
2    CAD  3F 7
     CUB  RSULT
1    =3=
3    CUB  1+
     STA  LEFT
     CAD  COLUM
     STA  RIGHT
     CAD  3F 7
     STA  OP
     CAD  4F 7
     CUB  MASTR
3    + code
4    CUB  QUER

SINGL  LDB  CMTX
       DBB  1+ 7
       BA
       STA  CMTX
SING1  - CAD  MATRX+1
       STA  COLUM      BFS HOLD
       SRT  7
       SUB  1F 7
       CNZ  2+ 7
       CUB  7F
       CAD  COLUM
       EXT  2F 7
       CNZ  2+ 7
       CUB  3F
       CAD  PVAR
       CNZ  2- 7
       CUB  LDBR
1      =31=
2      =0010 0=
3      ~~LDB 1F 7~~
       ~~CAD EXIT~~
       ~~STA HOLD~~
       DBB  2- 7
       CAD  COLUM
       LDB  2F 7
       CUB  GET
1      ~~2F=~~
2      CUB  COMBR
```




```
7       CAD  CNO
        SUB  ONE
        STA  CNO
        LDB  COLUM
        CAD  NUMST
        ADD  SPEC
        STA  SPEC
        CAD  PVAR
        CNZ  2+ 7
        CUB  CLOSE
        CUB  LDBR2
        STS  HOLD
CLOAS   CAD  TODO
        EXT  1F 7
        STC  TODO      STA RB
CLOSE   CAD  SPEC
        SRT  4
        CAD  TOGET
        CIRA 6
        EXT  1F 7
        SRT  6
        CAD  TOGET
        SLT  10
        STA  TOGET
        CUB  GET2
1       = -0000 22 1111 =

LDBR    CAD  COLUM
        STA  PVAR
LDBR1   LDB  2F 7
        CAD  GEXIT
        STA  HOLD      BTS HOLD
        DBB  2- 7
LDBR2   CAD  PVAR
        STA  TOGET
        CAD  3F 7
        LDB  4F 7
        CUB  GET1
        LDB  2F 7
        CAD  HOLD
        STA  GEXIT
        DBB  2- 7
        CUB  CLOAS
2
3       code for LDB
4       CUB  SB
```

```
QUEER   LDB  6F 7
        CAD  OP
        STA  1F 7
        CAD  LEFT
        STA  TEMP6
        CAD  1F 7
        CUB  3F 7
QUEER   CAA  COLUM
        STA  TEMP6
        EXT  5F 7
        CNZ  2F 7
        CAD  1F 7
        LDB  3F 7
        CUB  CONOT
0
1       =5810=0=
2       CUB  7F
3       CUB  1F
5       =0010=0=
1       ADD  2F 7
5       STA  2B 7
        CAD  TEMP6
        SRT  10
        CIRA 4
        ADD  2B 7
        STA  TEMP1
        CAD  3F 7
        CUB  OUTPT
2       =(FAD 0-0)=
3       CUB  7F
7       CAD  PVAR
        NOR  1F 7
        SRT  4
        STC  1F 7
        STA  PVAR
        CAD  4F 7
        SLT  4
        CUB  SB
2       CUB  3F
1       CAD  T
        SUB  ONE
        STA  T
        SUB  5F 7
        STA  TEMP1
        CAD  2B 7
        CUB  OUTPT
4       =(0 ——ADD)=
5       =(2x 00000 000 — STA 0000)=
3       CAD  T
        SUB  3F 7
        STA  TEMP1
        CAD  4F 7
        CUB  OUTPUT
3       (280 — 0 — LDB 0000)
4.      CUB  CLOAS
```

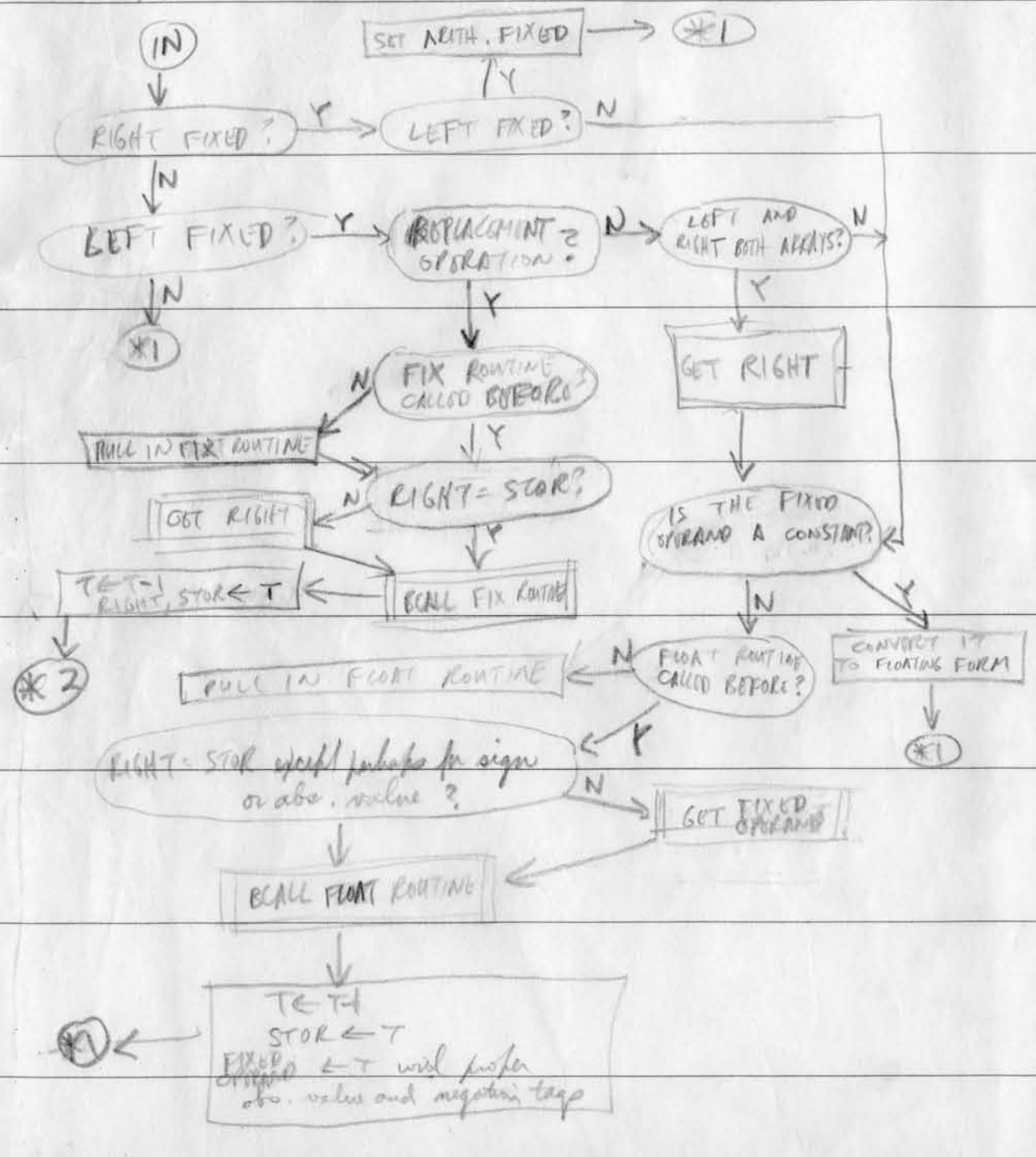
```
QUEER   LDB 6F 7
        CAD OP
        STA 1F 7
        CAD LEFT
        STA TEMP6
      - CAD HOLD
      - STA MEXIT
        DBB 2- 7
        CAD 1F 7
        CUB 1F
    6   =6=
QUEER   LDB 6F 7
      - CAD HOLD
      - STA GEXIT
        DBB 2- 7
        CAA COLUM
        STA TEMP6
    1   EXT 5F 7
```

MASTR subroutine. This routine compiles the output for addition, multiplication, division, exponentiation and the replacement operation. Operand code information are in LEFT and RIGHT. It handles negation, absolute value, and mixed arithmetic

Entry: LEFT, RIGHT, and OP are set up. rA = exit

Temp. storage need: MEXIT HIGHL
RIGHT
LEFT
OP
etc.

Subroutines need GET BCALL OUTPT

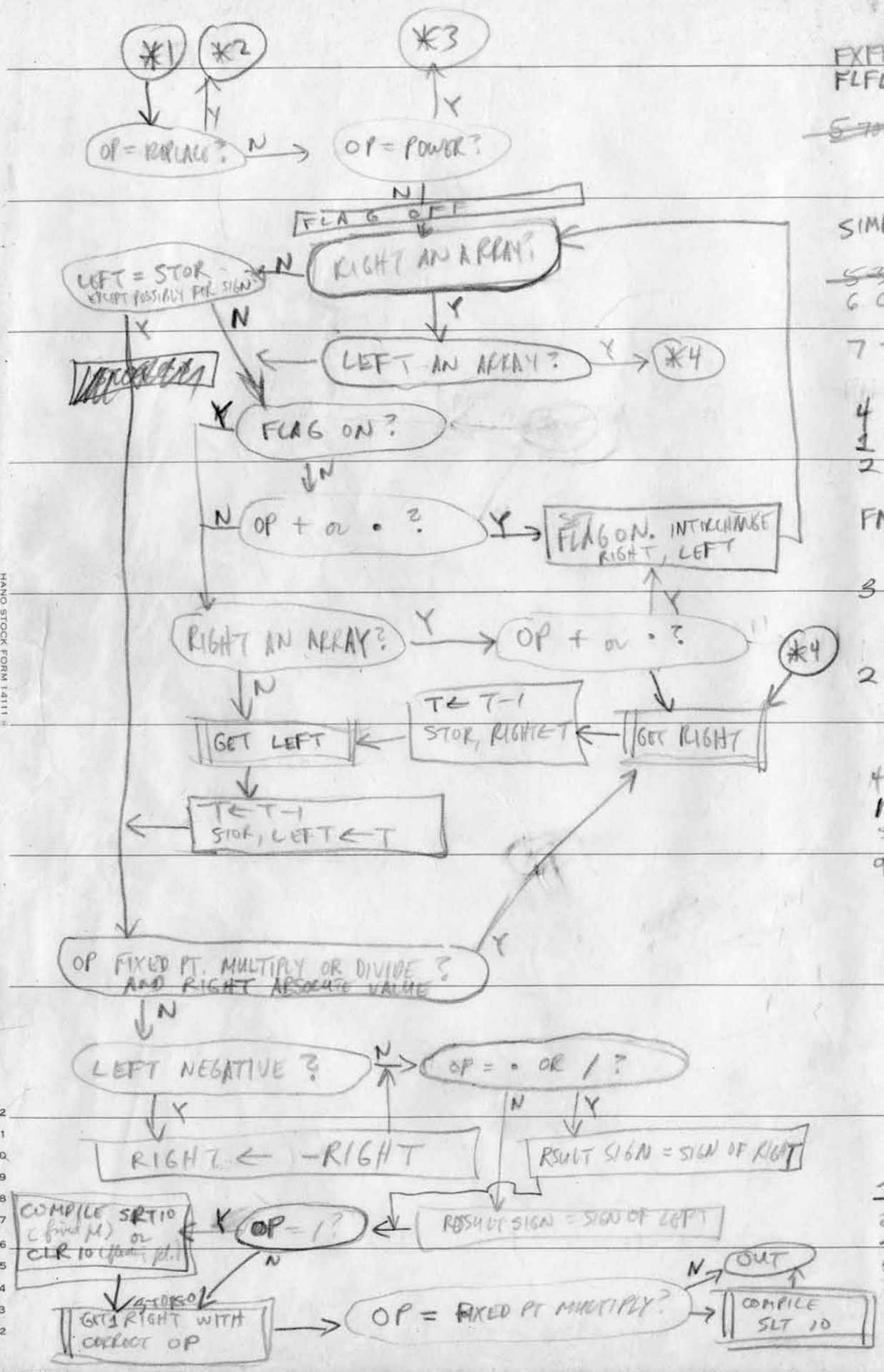



```
MASTR  STA MEXIT
       CAD RIGHT
3 3200 EXT 4F 7
       CNZ 1F 7
       CAD LEFT
       EXT 4F 7
       CNZ 2F 7
       CUB FLFL
1      CAD LEFT
       EXT 4F 7
       CNZ 3F 7
       CUB FLFX
2      CAD OP
       ADD 3B 7
       CCB 9F
       CUB FXFL
3      CUB FXFX
4      =11000000=
9      CAD SEG1
       OSD 7+ 7
       BOF 4+ 7
       SUA HIGHL
       STA HIGHL
       STA SEG1
       CAD RIGHT
       SUA STOR
       CNZ 2+ 7
       CUB 2F
       CAD RIGHT
       LDB 2- 7
       CUB GET
2      LDB 3F 7S
       CAA SEG1
       ADD SEG1 +1
       CUB BCALL
4      CAD T
       SUB ONE
       STA T
       STA STOR
       STA RIGHT
       CUB EQUALS
3      CUB 4B
```

```
FXFL   CAA LEFT
       ADD 4F 7
       CCB WORRY
FXFL1  CAD ONE
FLFX   STA MFL
       LDB MFL
       -CAD RIGHT
       ADD 5F 7
4 7000 BOF 1F 7
       CAD SEG1
       OSD 7+ 7
       CCB 4F
       SUA HIGHL
5 9600 STA HIGHL
       STA SEG1
1      -LDB RIGHT
2      -CAD NUMST
       CUB 1+
   2   CLR 60
       NOR 1F 7
       SRT 10
       CAD 2B 7
       SSC
       SLT 8
1      -STC NUMST
       CUB FLFL
2      ~~60~~
4      -CAD RIGHT
       STA 1B 7
       EXT 1F 7

       SUA STOR
       CNZ 2+ 7
       CUB 2F        ADD STOR
       CAD 1B 7      STA RIGHT
       LDB 2-
       CUB GET
1      11000000
2      CAA SEG1
       LDB 1F 7
       CUB BCALL
4      CAD T
       SUB ONE
       STA T
       STA STOR
LDB MFL CAD RIGHT-1
       EXT 5F 7
       STA 4B 7
       CSA STOR
       EXT 4B 7
```

```
WORRY  CAA RIGHT
       ADD 4F 7
       BOF 2+ 7
       CUB FXFL1
       CAD RIGHT
       LDB 3F F
       CUB GET
4 9000 CAD T
       SUB ONE
       STA T
       STA STOR
       STA RIGHT
       CUB FXFL1
5      CUB 4B
```

MFL
-STC RIGHT
CUB FLFL
3 CUB 4B
5 (= 11000000)




```
EXEX  CSU 4F 7
FLFL  ADD OP
      STA OP
      ADD SIMPL7
      CCB EQULS
      ADD 6F 7
      CCB POWER
SIMPL3 CAA RIGHT
      ADD 7F 7
      CCB MRAY
      CAA LEFT
      EXT 1F 7
      SUA STOR
      CNZ 2F 7
      CUB FN
4     =0018 0000 [fin]=
1     =1100 1111 11=
2     CUB 7F

FN    CAD OP
      EXT 4F 7
      CNZ 2F 7
3     OSD LEFT
      BOF 8F 7
      CUB 9F

2     CAD RIGHT
      EXT 1F 7
      NOR 3B 7
      CUB HARD
      =0010 0010 0=
      =0 0110000 =
fudge so multiply overflow

9     CAA OP
      EXT 1F 7
      CNZ 3F 7
      CAD RIGHT
      STA LEFT
      CAA OP
      EXT 2F 7
      CNZ 3F 7
      SLT 6
      ADD 5F 7
      STA TEMP1
      CAD 3F 7
      CUB OUTPT
      =0000010 0=
      =000 010 011
      SLT 10
      =-PN
      CUB 14

HARD  LDB 8F 7
      CAD RIGHT
      CUB GET
```

```
      CAD RIGHT
      STC TOGET
      CAD OP
      EXT 1F 7
      LDB 2F 7
      CUB GET1
3     CAD OP
      EXT 4F 7
      CNZ MEXIT
      CAD 5F 7
      STA TEMP1
      CAD MEXIT
      CUB OUTPT
1     = 11 =
2     CUB 3B
4     0010011000
5     = (SLT 10) =

MRAY  CAD LEFT
      ADD 1F 7
      CCB HARD
7     CAD OP
      OSD 1F 7
      CCB 7F
      EXT 2F 7
      CNZ 3+ 7
      CUB 7F
COMUT CSA OP
      STA OP
1 7000 CAD RIGHT
      STA 7B 7
      CAD LEFT
      STA RIGHT
      CAD 7B 7
      STC LEFT
      CCB SIMPL
-2    =0000110000=

7     CAD RIGHT
      ADD 1F 7
      BOF 2F 7
6     CAD LEFT
      LDB 5F 7
      CUB GET
      CAA OP
2     EXT 3F 7
1 7000 CNZ 4F 7
5     CUB 7F
3     0000 11 0000
4     CUB COMUT
8     CUB 9B
```

```
CAD RIGHT
STA TOGET
CAD OP
STA STOR
EXT 1F 7
LDB 2F 7
CUB GET1
```

```
CAD RIGHT
STA STOR
STA LEFT
STA FN
CUB
```

```
CAD RIGHT
STA STOR
STA LEFT
STA FN
CUB
```

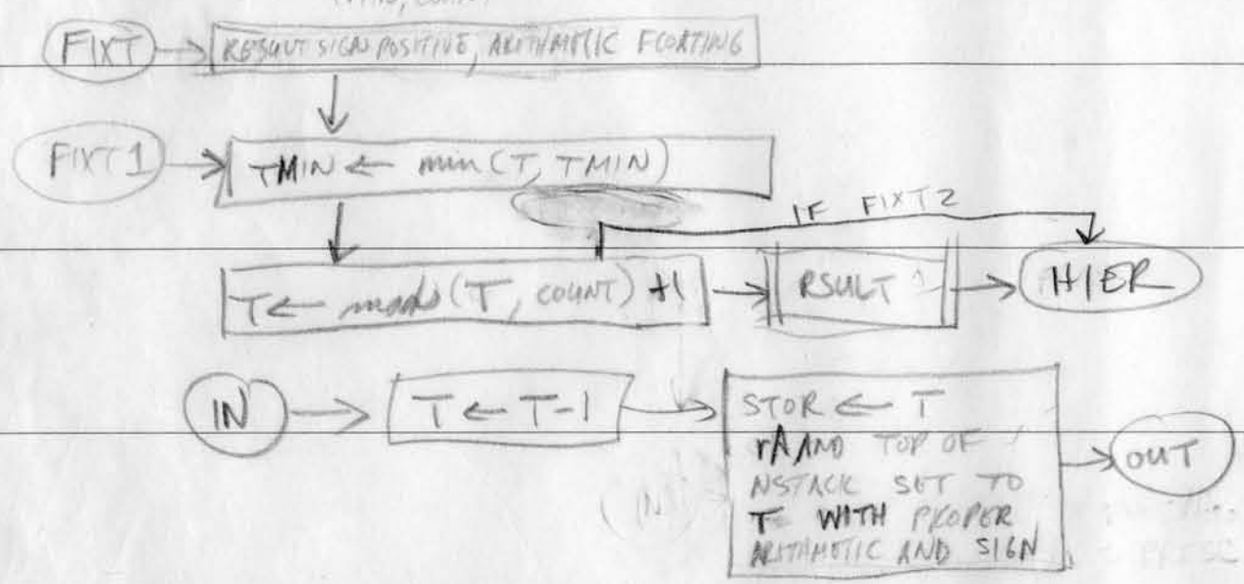
```
CAD RIGHT
ADD 1F 7
BOF 2F 7
CAD LEFT
LDB 5F 7
CUB GET
CAA OP
EXT 3F 7
CNZ 4F 7
```

```
0
```

next page has the end of MASTR.

RSULT SUBROUTINE and FIXT routine. Before returning to the scanner after generating coding, we find a T-storage into which the A-register can be put if necessary. We also check to see how much T-storage we have used up so far, so that during the next procedure declaration we can avoid common T-storage.

Entry: rA = exit
Exit: rA, top of NSTAK = result of last operation.
Temp Storage used: OP, LEFT, CNAME, T, STOR
FIXT: TMIN, COUNT



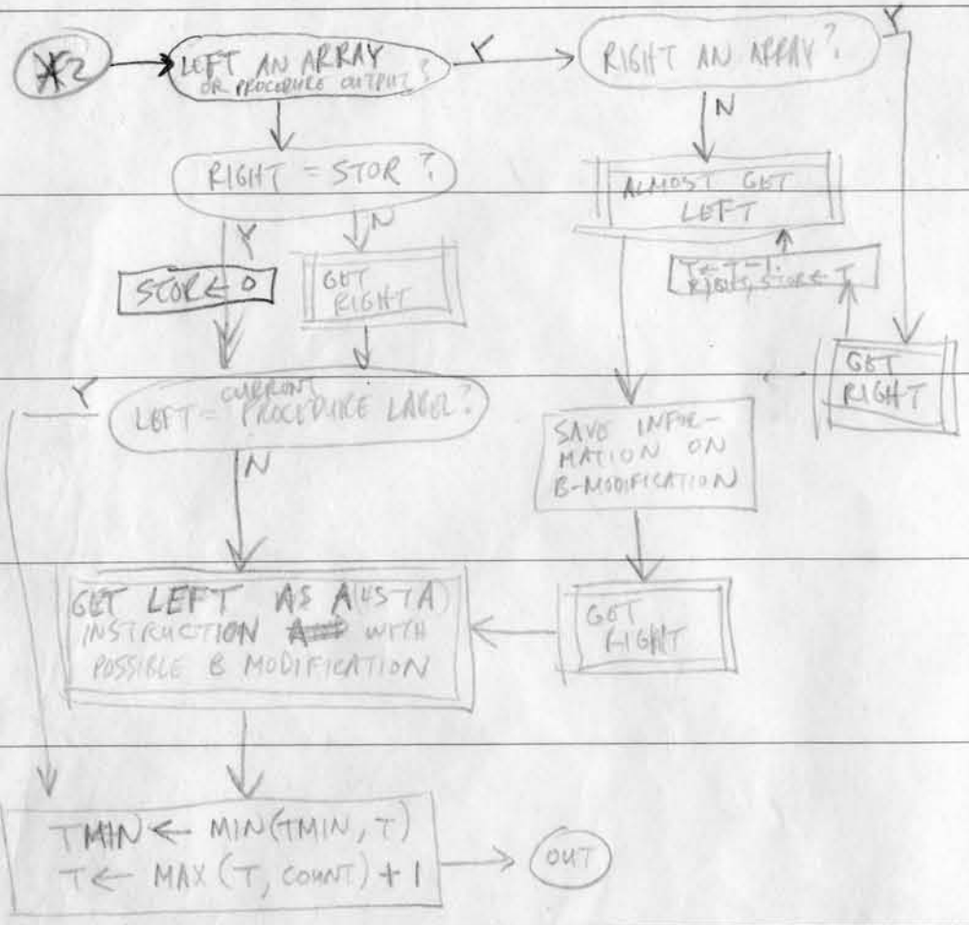

```
FIXT3  STA  6F 7
FIXT2  CAD  4F 7
       STA  5F 7
FIXT   STC  ~~LEFT~~ OP
FIXT4  STC  ~~LEFT~~ OP
       STC  ~~OP~~   LEFT
FIXT1  CSU  T
       ADD  TMIN
       OSD  7+ 7
       BOF  3+ 7
       CAD  T
       STA  TMIN
       CAD  COUNT
       CNZ  2+ 7
       CAD  T
5      CUB  1F
RSULT  STA  3F 7
       CAD  T
       SUB  ONE
1      STA  T
       STA  STOR
       CAD  OP        SLT 1
       EXT  4F 7
       ADD  5F 7
       SRT  10
       CAD  LEFT
       SLT  10
       STA  1B 7
       CSU  STOR
       EXT  1B 7
       LDB  CNAME
       STA  NSTAK
3      CUB  HIER
4      0010 ---- 0
5      11001 ---- 1
```

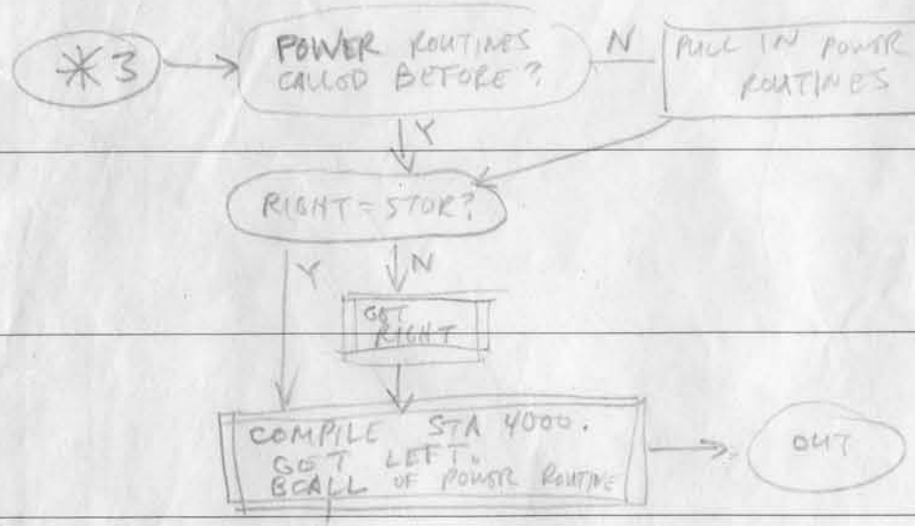
STA T
6 CUB HIER
4 ADD ONE.

LATER IDEAS:

FIXT2 fixes up TMIN and then sets T = 1 higher than the next T-storage available. Exits to HIER
~~FIXT3~~ same as FIXT2 – but exits to place specified in rA
FIXT5 fixes up TMIN only, exits to $[rA]_b$. On exit rA = next available T-storage
~~FIXT~~ fixes up TMIN, puts next avail T-storage onto top of NSTAK and floating N. and + sign. exits to HIER
FIXT taken out!
FIXT4 same as FIXT with rA open entry finally specifying fixed point
FIXT1 same as FIXT with sign and sign of NSTAK determined by OP and LEFT

RSULT same as end of FIXT, except used for intermediate results.






```
GOULS  CAA LEFT
       ADD 3F 7
       CCB 2F
       CAD RIGHT
       EXT 4F 7
       SUB STOR
       NOR 1F 7
       LDB 1F 7
3 6500 CAD RIGHT
       CUB GET
4      = -1100111111=
1      CUB 7F
2      CAA RIGHT
       ADD 6F 7
       BOF 1F 7
       CSA LEFT
3 6 7000 STA TOGET
       LDB 2F 7
       CUB GET1
1      CAD RIGHT
       LDB 4F 7
       CUB GET
8      CAD T
       SUB ONE
       STA T
       STA STOR
       STA RIGHT
       BUN 3F 7
2      CUB 8F
4      CUB 8B

8      CAD TOGET
       EXT 1F 7
       STA COLUM   LDB 2F 7
       CAD TODO
       STA PVAR
       CAD RIGHT
       CUB GET
1      +220000 (11
2      CUB 8F

7      CAD LEFT
       SRT 8
       SUB 1F 7
       CNZ 2+ 7
3      CUB 6F
       CAD LEFT
       STC TOGET
       BUN 2F 7

8  CAD COLUM
   STA TOGET
   CAA PVAR
2  EXT 5F 7
   LDB 3B
   CUB GET1
1  =34=
5     11 (HEXIA)
6

6  CSU T
   ADD TMIN
   OSD 7+ 7
   BOF 3+ 7
   CAD T
   STA TMIN
   CAD COUNT
   CNE 2+ 7
   CAD T
   ADD ONE
   STA T
   BUN MEXIT

POWER  CAD SEG2
       OSD 7+ 7
       BOF 4+ 7
       SUA HIGHL
       STA HIGHL
       STA SEG2
       CAD RIGHT
       SUA STOR
       CNZ 3+ 7   STA STOR
       CUB 1F
       CAD RIGHT
       LDB 2- 7
       CUB GET
1      CAD 2F 7
       STA TEMP1
       CAD 3F 7
       CUB OUTPT
4      CAD LEFT
       LDB 1F 7
       CUB GET
5      CAD OP
       EXT 6F 7
       CNZ 7F 7   ADD SEG2
8      LDB MEXIT
       CUB BCALL
2      (STA 4000)
3      CUB 4B
1      CUB 5B
6      =0001000 07
       CAD SEG2+1
       BUN 8B 7
```

STANDARD CODES FOR OPERATORS.

SORT THIS!

B STORAGE	26
INTOCK	95
ARRAY	96
T STORAGE	28 00 AA EEEE
COMMENT	99999999
SIMPLE VARIABLE	22 00 AA EEEE, 22 88 00 EEEE
✓ RIGHT PARENTHESIS	01 0000 0001
✓ LEFT PARENTHESIS	40 0000 0001
✓ COMMA	42 08
✓ PLUS	52 20 01 0026, find: 52 02 0100 04
✓ TIMES	52 22 10 0030, find: 52 04 1000 08
✓ DIVIDE	52 24 00 1034, find: 52 06 00 10 12 MOD 200 001012
✓ POWER	67 26 0000 32, find: 67 08 000000
✓ MINUS	47 26
✓ EQUAL	68 06 00 0000
EQL	54 18 09 1044
NEQ	54 18 09 0144
GTR	54 18 0 01000
GEQ	54 18 1 1 01 00
LSS	54 18 1 0 10 00
LEQ	54 18 0 10 1 00
AND	55 14 X x
OR	56 12 X x
NOT	57 16 x
SEMICOLON	41 02
IF	52 04 XX 9999
UNTIL	59 04 XX 9999
BEGIN	40 0000 0002
END	01 0000 0002
SWITCH	60 10
GO	72 88
PROCEDURE	94 88 0000
FOR	44 08
STOP	43 06
PCS	61 88 000188
ABS	51 88 00 1234
CONSTANT	03 00 AA RRRR
INPUT	90 08 0000
OUTPUT	91 08 0000
ENTER	76 88
SUBROUTINE	93 88 0000
FINISH	5001 30 (BUILD)
RETURN	4501 30 (SCAN1)
FORMAT	92 0000
PROC. SIMPLE OUTPT	39 08 AA BBBB, 39 88 00 BBBB
PROC. VECTOR	38 00 AA BBBB
PROC. MATRIX	37 00 AA BBBB
NORMAL VECTOR	36 00 AA BBBB
NORMAL MATRIX	35 00 AA BBBB,
CURRENT PROCEDURE NAME	34 60 0 0
DEFINED PROCEDURE NAME	32 60 00 BBBB
DEFINED LIBRARY FUNCTION	31 88
LIBRARY ROUTINE	7
PROCEDURE PROCEDURE	33 60 00 BBBB
LIBRARY PROCEDURE	8
LABEL	12 9999 9999
PROC. LABEL	13 00 1 BBB BBBB

UNCALLED-FOR

UNCALLED-FOR

TF
0018 XXXX 14

XTF

L-R

QQ= arithmetic. non-zero when fixed. (10 on fixed point constants).
AA = addr. value
BBBB equivalent
RRRR reference to NUMST
BBBB reference to PART 2

OPERATION CODES

0	CAD	22	LDB
1	CJU	23	LDB
2	CAA	24	LDB
3	CSA	25	LDB
4	AOD	26	FAD
5	SUB	27	FSU
6	ADA	28	FAI
7	SUA	29	FSA
8	MUL	30	FMU
9	MUL	31	FMU
10	CCB	32	FMA
11	BUN	33	FMA
12	DIV	34	FDV
13	DIV	35	FDV
14	CUB	36	FDA
15	SCUB	37	FDA
16	SRT	38	CLR
17	SLT	39	IBUN
18	QCUB	40	CLRA
19	QSTA		
20	4STA		
21	CNZ		

01), END
03 CONSTANT

NULLSTRING 09 00 13 0008
MOD 6588

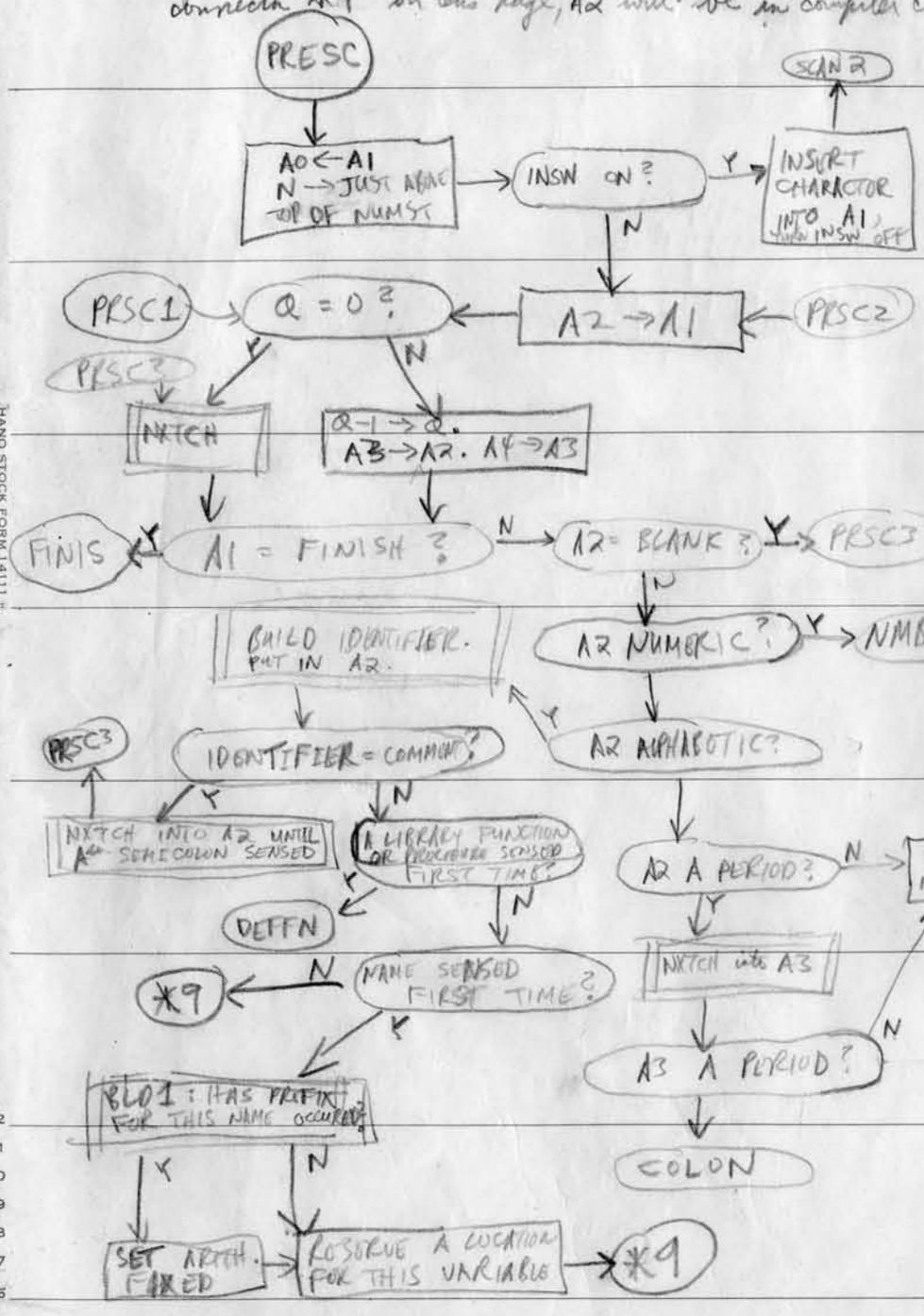
9999 = ho: 4444 is a chain address

MXTCH code

	00 →	00	(00)
.	03 →	01	(03)
)	04 →	07	(00 5900)
;	$ 13 →	07	(13)
+	10 →	04	(10)
−	20 →	05	(20)
−	34 →	05	(20)
*	14 →	06	(28) (00 6000)
/	21 →	02	(21)
?	23 →	08	(23)
(	24 →	10	(00 5300)
=	33 →	03	(00 6900)

35 and over is an alphabetic (by definition)

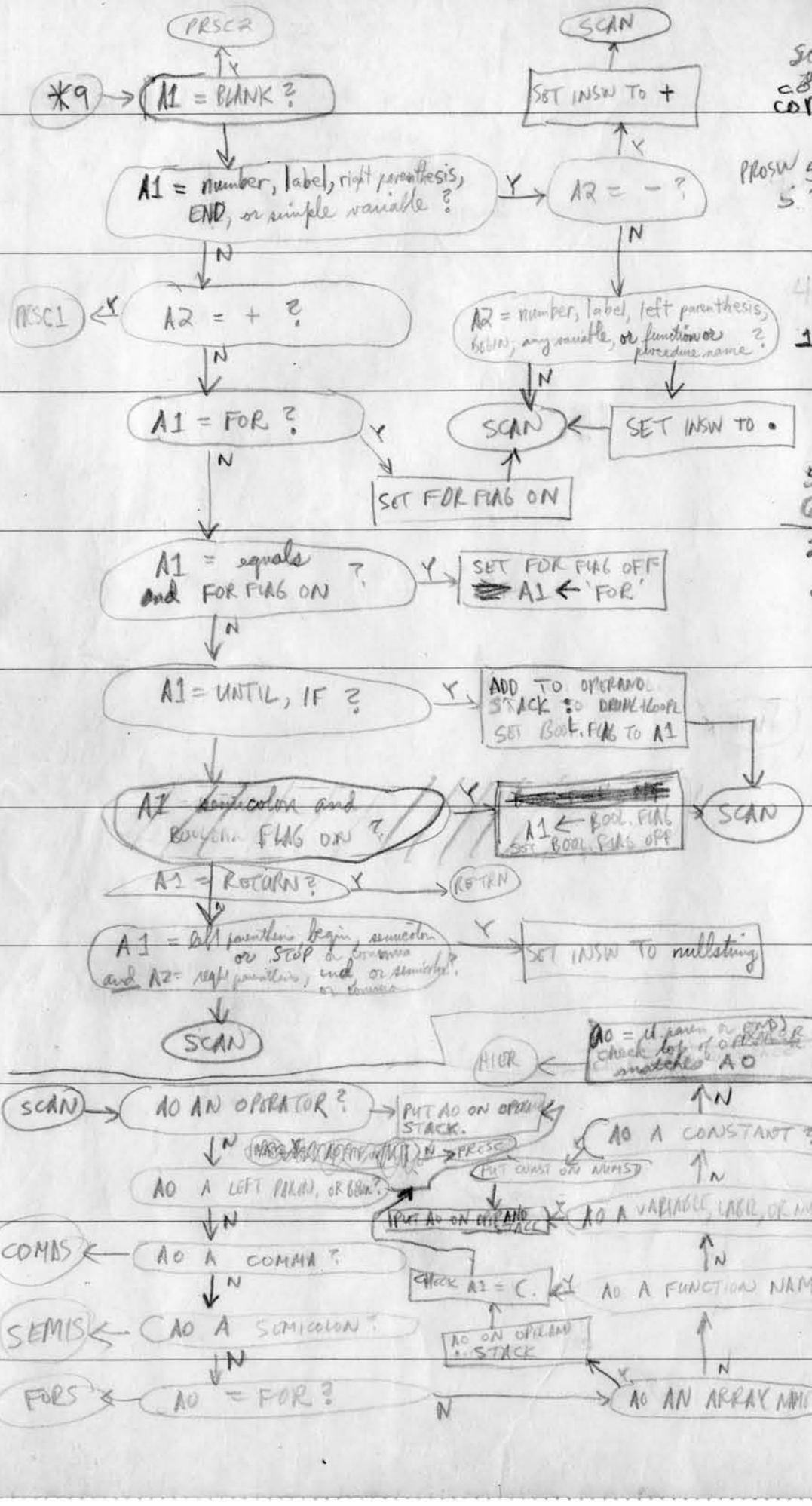
SCANNING. ~~Th~~ Scanning in this compiler operates not only on symbol pairs but on strings of 3 to 5 symbols. Characters come in from the NXTCH routine into A2, A3, or A4 and from there they proceed leftward until they reach A0 and A1. Not all characters get as far as A0 — many of them are processed earlier, such as statement labels, integer declarations, and so on. The processing of A0 and A1 is done by the scanner, and the control of what gets into A0 and A1 is handled by the prescanner. / When characters come in from NXTCH they are in either Cardation code or a special code for special characters. The contents of A3 and A4 are always in this code. The input language is changed into compiler code, however, and groups of symbols (e.g., names) become a single entity. The contents of A0 and A1 are in compiler code. A2 is the transition point: When we reach connector #9 on this page, A2 will be in compiler code, but at the beginning A2 is in NXTCH code.




```
PRESC  CAD  A1
       STA  A0
       LDB  CNO
       CAD  N
      -STA  NUMST + 1
       CAD  INSW
       CNZ  2+ 7
       CUB  PRSC2
       CUB  SCAN2
PRSC2  CAD  A2
       STA  A1
PRSC1  LDB  Q
       BA
       CNZ  2F 7
PRSC3  CAD  3F 7
       CUB  NXTCH
2      SUB  ONE
       STA  Q
       CAD  A3
       STA  A2
       CAD  A4
       STA  A3
3      CUB  4F
4      CAD  A1
       SUB  1F 7
       CNZ  2+ 7
       CUB  FINIS
       CAD  A2
       CNZ  2+ 7
       CUB  PRSC3
       ADD  2F 7
       CCB  NMBER
       ADD  3F 7
       BOF  2+ 7
       CUB  4F
       CAD  A2
       STA  A3
       CAD  5F 7
1  5001 CUB  BUILD
2       9999999920
3          39
5       CUB  8F
```

```
PRSC4  STC  2F 7
       STA  Q
       BUN  2+ 7
4      ADD  1F 7
       BOF  2F 7
       LDB  ONE
       CAD  3F 7
       CUB  NXTCH
4      SUB  ONE
       CNZ  2F 7
       CUB  COLON
       LDB  A2
2     -CAD  CHTBL
       CUB  CODE1
1      39
3      CUB  4B
6      SUB  1F 7
       CNZ  2+ 7
       CUB  PRSC3
7      LDB  2F 7
       CAD  3F 7
       CUB  NXTCH
1          <;> match code
3      CUB  6B
8  4600 LDB  TEMP3
6  050- CAD  PART2
7  3100 STA  A2
       ADD  6F 7
       CCB  7F
       ADD  5F 7
       CCB  DEFFN
       ADD  8B 7
       CCB  CODED
6      LDA  6B 7
       CCB  4F
       ADD  7B 7
       CCB  CODED
5  0001 CUB  2F
2  1000 BA
       STC  EQUIV
       STA  EXIT
       STA  TEMP2
       STA  TEMP5
```

```
       CAD  A1
       ADD  2B 7
       CCB  DECLR
       ADD  2B 7
       CCB  INTAR
       ADD  2B 7
       CCB  ENTGO
       CAD  R1
       EXT  1F 7
       STA  R+10
       CUB  BLO1
YES..  CAD  3F 7
       BUN  2F 7
1     =111100=
NO...  STC  3F 7
2      LDB  EQUIV
      -ADD  PART2
       STA  TEMP1
       CAD  LOOP6
       CNZ  1F 7
       CAA  HIGHL
       ADD  TEMP1
       ADD  1F 7
      -STA  PART2
       EXT  5F 7
       STA  HIGHL
       CUB  CODED
3     =0010-0=
4      219-59=
5     =1111=
1      CUB  1+
       SUB  ONE
       STA  LOOP6
       ADD  1F 7
       ADD  TEMP1
      -STA  PART2
       CUB  CODED
1      22 0000 6000
4      CAD  12
       EXT  =111100
       ADD  TEMP3
       CUB  CODE1
```



SCAN. Except for some more complicated maneuvers if A0 is a comma or semicolon, according to its type it puts information on or takes it off the various stacks.

```
CODE 1  STA A2
CODE(E) CAD A1
        CNZ 2+ 7
        CUB PRSC2
PROSW 5900 ADD 1F 7
      5 3800 CCB 9F
        CAD A2
        SUB 5F 7
        CNZ 1F 7
        CAD 6F 7
        CUB SCAN1
      1 7000 CAD A2
        ADD PROSW 7
        CCB SCAN
        ADD 5B 7
        CCB 2F
        CUB SCAN
      5 minus code
      6 plus code
      2 CAD 1F 7
        CUB SCAN1
      9 CAD A2
        SUB 6F 7
        CNZ 2+ 7
        CUB PRSC1
        CAD A1
        SUB 2F 7
        CNZ 3F 7
        CAD 4F 7
        STA FORFL
        CUB PRSC2
      3 CAD FORFL
        SUB A1
        CNZ 3+ 7
        CAD 2F
        CUB 9F
        CUB 3F
      1 times code
      6 plus code
      2 for code
      4 equals code
      3 LDB A1
        IB
        DBB 2F 7
        CAD A1
        STA BOOFL
        CAD DRUML
        ADD LOOPL
        CUB 5F
        STA BOOFL
        CUB PRSC2
      2 CAD BOOFL
        CNZ 2+ 7
      4 CUB 2F
        CAD A1
        SUB 3F 7
        CNZ 4B 7
```

```
        STC A1
        STA BOOFL
        CUB SCAN
      3 semicolon code:
        INSERT FROM BELOW:
        CAD A1          rewrite
      2 SUB 8F 7
      3 0400 CNZ 2+ 7
        CUB RETRN
        CAD A2
      6 0900 SRT 8
      7 SUB ONE
        CNE 2F 7
      5 CAD A1
        ADD 2B 7
        CCB SCAN
        ADD 3B 7
        BDF 7F 7
        DBB 1F 7
      3 CUB
      2 SUB 4F 7
        CNZ 3B 7
        BUN 5B 7
        CAD 6B 7
      7 8 4501 CUB SCAN1
      4 =40=
      9 STC A1
        STA FORFL
      8 BUN SCAN
        STC A1
        STA INSW
        CAD A0
        ADD 9+ 7
        CCB 9F
      3 2300 ADD 8+ 7
        CCB 8F
      5 0400 ADD 9+ 7
        CCB 7F
        ADD 4B 7
        CCB 6F
        ADD 3B 7
        CCB 5F
        ADD 5B 7
        CCB 4F
      3 0400 CUB 1+
```

```
        LDB COP
        CAD OPST -1
        ADD 2F 7
        BDF 4F 7
        ADD 3F 7
        CUB FEDUR
      4 DBB 1+ 7
        BA
        STA COP
        CAD OPST+1
      2 5600 SUB AD
        SUB 1F 7
      3 0200 CNZ 2+
        CUB HIER
        CUBR ALARM
      1 =390=
      4 CAD CNO
        ADD ONE
        STA CNO
        ADD A0
        STA A0
        BUN 5F 7
      7 CAD 4F 7
        STA 1F 7
      5 LDB CNAME
        IB
        BA
        STA CNAME
        CAD A0
        STC OPST
      1 CUB HIER
        STA A0
      6 CAD A1
        SRT 8
        SUB 1F 7
        CNZ 3F
        CUB 9F
      4 EXT 1+ 7
        1188 00 1111
      1 =40=
      3 CUBR ALARM
      8 CAD A0
        SRT 8
        STA 9F 7
        LDB 9F 7
        BUN 38- 7
        CUB 5B
        CUB 9F
        CUB SEMIS
        CUB COMAS
        CUB 9F
        CUB FORS
      9 LDB COP
        IB
        BA
        STA COP
        CAD A0
        STC OPST
```

INSERT

```
      5 ADD 6F 7
        LDB CNAME
        STA NSTAK+1
        IB
        BA
        STA CNAME
        CUB PRSC2
      2 CAD A1
        SUB 3F 7
        CNZ 4F 7
        CAD BOOFL
        CNZ 2+ 7
      4 CUB A2F-LOBONE
```

PRESC here

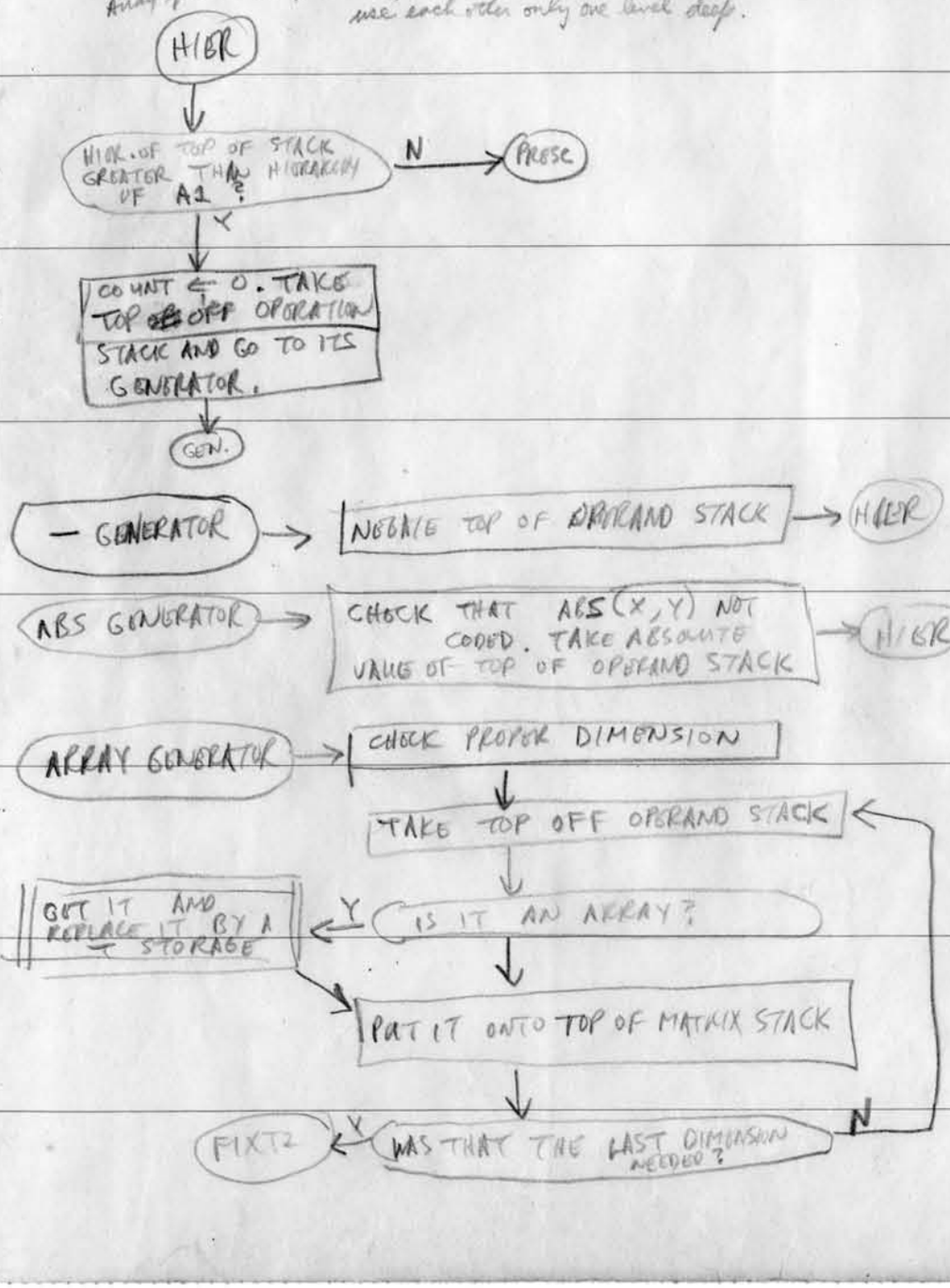
STACKS

~~Boolean stack; counter: CBOOL~~
NUMST Constant stack; counter: CNO
NSTAK Operand stack; counter: CNAME
OPST Operation stack; counter: COP
MATRX Matrix operands stack; counter: CMTX
TFST True-False stack; counter: CTF

All are 'last-in... first-out' stacks except NUMST. In the latter case it is possible to take things off the stack from somewhere near the top, but it is rigged so that whenever SCAN returns there are no holes in the table. The fact that MATRX must be last-in-first-out results in special complications in the MASK generator, for if both RIGHT and LEFT are wanted we must get RIGHT before we get LEFT.

HIER. Here's where things start to happen; operators are taken off the operation stack and compiled-for if their hierarchy is stronger than the hierarchy of A1. After an operation is compiled it should return to HIER.

-generator is unary minus. Binary minus is replaced in the precanner by +- pair.
Array generator compiles array subscripts immediately so GET and MASK routines use each other only one level deep.




```
HIER  LDB COP
      CAD A1
      EXT 1F 7
      STA HIER+7
     -CAD OPST
      EXT 1F 7
      SUB HIER 7
      OSD HIER 7
      CCB PROSC
2     DBB 1+ 7
      BA
      STC COP
      STA COUNT
     -CAD OPST+1
      SRT 8
      STC 2B 7
      LDB 2B 7
     -2BUN MASK
1     0011000000
```

```
MASK+31  defined bl. function
MASK+32  procedure procedure
MASK+33  procedure-regular
MASK+34  cannot precede name
MASK+35  CUB ARR2
MASK+36  CUB ARR1
MASK+37  CUB ARR2
MASK+38  CUB ARR1
MASK+42
MASK+44  FOR
MASK+47  CUB MINUS
MASK+51  CUB ABS
MASK+52  CUB ASMD
```

```
MASK+54  Relations
MASK+55  AND
MASK+56  OR
MASK+57  NOT
MASK+58  IF
MASK+59  UNTIL
MASK+60  SWITCH
MASK+61  PCS
MASK+67  CUB PWR
MASK+68  CUB REPL
```

```
MINUS  LDB  CNAME
       -CSU  NSTAK
       -STA  NSTAK
       CUB  HIER
ABS    SLT  4
       SUB  1F 7
       CNZ  2F 7
       LDB  CNAME
       -CAD  NSTAK
       EXT  3F 7
       -STA  NSTAK
       CUB  HIER
1      =8800=
2      CUB ALARM
3      =11 11 22 11 11=
ARR2   CAD ONE
ARR1   STC A0
       SLT 4
4 7000 SUB 1F 7
       SUB A0
       CNZ 2F
8      LDB CNAME
       DBB 1+ 7
       BA
       STA CNAME
       -CAA NSTAK+1
       ADD 4B 7
       CCR 4F
       -CAD NSTAK+1
       -CUB 5F
```

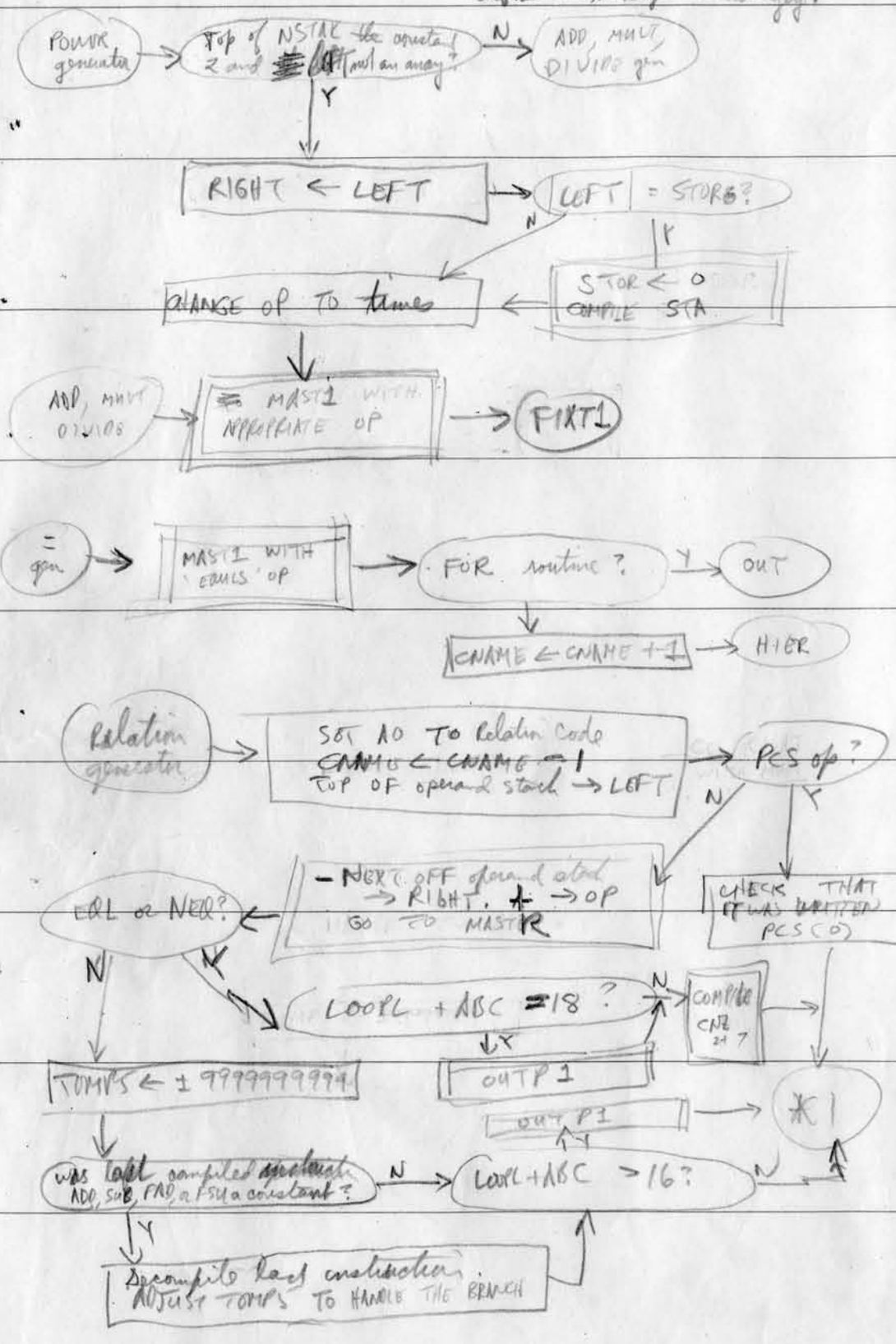
CUB ALARM =8800=

```
4  -CAD NSTAK+1
   LDB 3F 7
   CUB GET
6  CAD T
   SUB ONE
   STA T
   STA STOR
5  LDB CMTX
   -STA MATRX+1
   1B
3  BA 6B
   STA CMTX
   CAD A0
   CNE 2+ 7
   CUB HIERR
   SUB ONE
   STC A0
   CUB 8B
```

```
1  SUB TMIN
   OSD 2- 7
   BOT 3+ 7
   CAD T
   STA TMIN
```

MAST1 entrance to MASTR: Decrements CNAME and puts top operands of NSTAK into LEFT and RIGHT.

Relation generator: This compiles blocks of coding with undefined linkage addresses. The AND, NOT, OR, IF, and UNTIL generators will define the linkages accordingly.



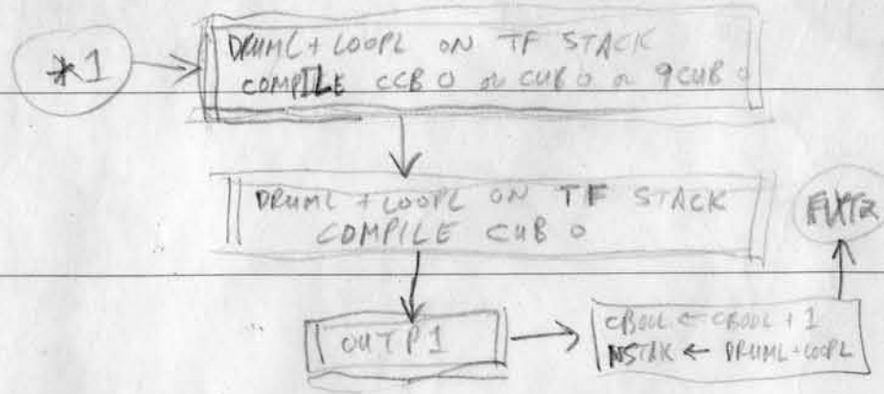

```
PWR   LDB  CNAME
      -CAA NSTAK
      ADD  1F 7
      CCB  ASMD
      EXT  2F 7
      CNZ  3+ 7
      CAD  3F 7
      BUN  2+ 7
      CAD  4F 7
      -LDB NSTAK
      -SUB NUMST
1 9600CNZ  6F 7
      CUB  5F
      =0010 0=
2     = 2.0 =
3
4     =2=
6     CUB  ASMD
5     STA  TEMP1
      CAA  TEMP1
      ADD  1F 7
      CCB  ASMD
      -CAD TEMP1
      -STA NSTAK
      SUB  STOR
      CNZ  2F 7
      STA  STOR
1 7000ADD  3F 7
      STA  TEMP1
      CAD  2E CUB OUTPT
2     CUB  4F
3     28 0000 0000
      -19 0003
4     CAD  1F 7
      BUN  3+ 7
ASMD  LDB  COP
      -CAD OPST+1
      STA  OP
      BUN  2+ 7
MAST1 STA  2F 7
      LDB  CNAME
      DBB  1+ 7
      BA
      STA  CNAME
      -CAD NSTAK+1
      STA  RIGHT
      -CAD NSTAK
      STA  LEFT
      CAD  2F 7
      CUB  MASTR
      multiply code
1     CUB  FIXT1
2
```

```
REPL  LDB  1F 7
REPL1 CAD  2F 7
      STA  OP
      BA
      ADD  3F 7
      CUB  MAST1
1     CUB  4F
2     equals code
3     CUB  0
4     CAD  CNAME
      SUB  ONE
      STA  CNAME
      CUB  HIER
PCS   CAD  3F 7
      SLT  4F 7
      SLT  8
RELAT CRA  4
      STA  TEMP7
      CAD  1F 7
      STA  CNAME
      -CSU NSTAK+1
4     CUB  MAST3
      SUB  ONE
      STA  CNO
      CUB  RFIN
3 CAD CUB  1+
1     +
2     CUB  1+
      CAA  TEMP7
      ADD  1F 7
      CCB  EQNEQ
      CSU  LEFT
      OSD  6+ 7
      BOF  3+ 7
      CSU  TEMP7
      STA  TEMP7
      CAD  2F 7
      SRT  10
      CAD  TEMP7
1 1000SLT  10
      STA  TEMP5
      LDB  LOOPL
      -CAD OUT-1
      EXT  3F 7
      CUB  4F
2     =9    9=
3     1000
```

CAD 1F 7
STA OP
CAD 2F 7
STA MEXIT

CAD CNO
SUB ONE
STA CNO
CAA TEMP1

(New flowcharts on prev. page.



CTF initially ≥ 10

TF stack code

```
7   LDB  2F 7
    CUB  CNOUT
4   SUB  1F 7
    CNZ  2F 7
    CAD  3F 7
    MUL  OUT 1
    RND  9F
    CNZ  3+ 7
    CAD  CON
    CUB  4F
    CSU  CON
    CUB  4F
    CUB  8F
2   8000
1   = 00005 0000 =
3
```

```
4   ADD  TEMP5
    BOF  4F 7
    STA  TEMP5
3   CAD  ONE
8   ADD  PHOR3
    LDB  GF 7
    CUB  FULL2R
    STA  TEMP2
4   CAD  TEMP7
    SRT  1
    STA  TEMP7
    CAD  4F 7
    SUA  TEMP2
    SRT  10
    CSU  TEMP5
    SLT  10
    BUN  3B 7
4   =9=   =9=
```

```
    DBB  1+7
    SBA  NEQ1
    STA  LOOPL
    LDB  ABC
    DBB  1+7
    STA  ABC
```

```
    6 BA
    STA ABC
```

```
    LDB  CTF
    CAD  TEMP5
    -STA TFST+2
    EXT  1F 7
    SUB  1F 7
    SUB  TEMP5
    -STA TFST+2
    BA   2
    ADD  1+ 7
    STA  CTF
    CAD  3F 7
    STA  TEMP1
    CAD  4F 7
    CUB  OUTP7
1   /0000
3   (CUB 0)
4   CUB 1+
```

```
7   LDB  3F 7
    CUB  1+ 7
9   SUB  3F 7
    STA  TEMP1
    CAD  4F 7
    CUB  OUTPT
3   22 0000
4   CUB RFIN1
```

```
RFIN1 CAD PHOR3
      CNZ FORF
RFIN  CAD TEMP7
      EXT 1F 7
      ADD CBOOL
      CIRA 7
      ADD DRUML
      ADD LOOPL
      LDB CTF
      -STA TFST+2
      BA
      ADD 2F 7
      STA CTF
      CAD 3F 7
      STA TEMP1
      CAD 4F 7
      CUB OUTP7
2     = 2 =
3     = (CUB 0) 1 0010 0
4     CUB 1+
```

```
      CAD CNE
      LDB 5F 7
      CUB FULL
NEQ1  CAD 3F 7
      STA TEMP1
      CAD 4F 7
      CUB OUTPT
3     (CNZ 24)
4     CUB RFIN
```

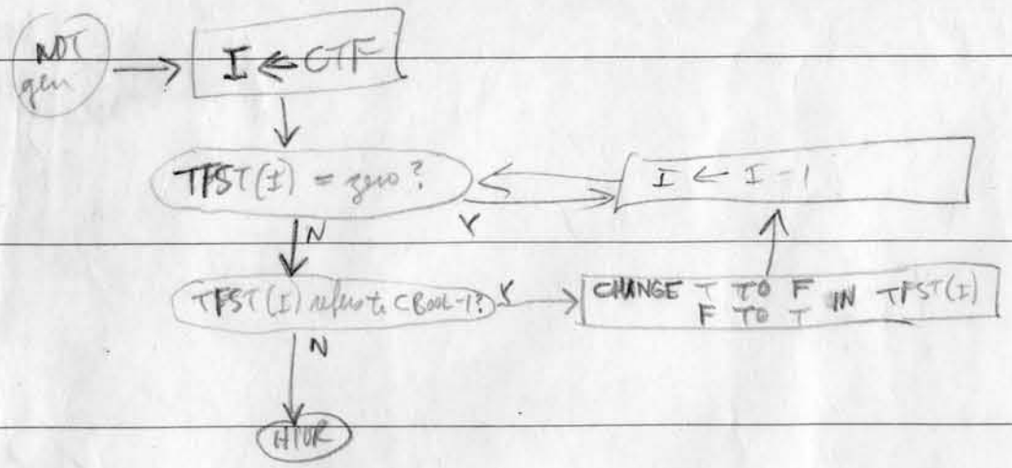
```
2     LDB CNANGLE
      CAD DRUML
      ADD 3F 7
      -STA NSTAK
      CAD CBOOL
      ADD ONE
      STA CBOOL
      CUB FIXTR
3     1300 0400D
```

```
PCS1  SUB 1F 7
      SUB CNO
      LDB CNO
      -SUB NUMST
      CNZ 2F
      DBB 1+ 7
      BA
      STA CNO
      ADD 3F 7
      BOF 21 7
      CUB RFIN
      CUB OUTP1
1     0310 0
2     CUB ALARM
```

RFIN1 CAD PHOR3
RFIN CNZ 1F 7
CAD TEMP7
EXT 3F 7
CIRA 7 ADD CBOOL
ADD DRUML
ADD LOOPL
STA TEMP5
CAD TEMP7
EXT 2F 7
SRT 1
ADD 2F 7
STA TEMP1
CAD 4F 7
CUB OUTP3
CUB RFIN
1 0000
2 0010 0
3
4 CUB 1+

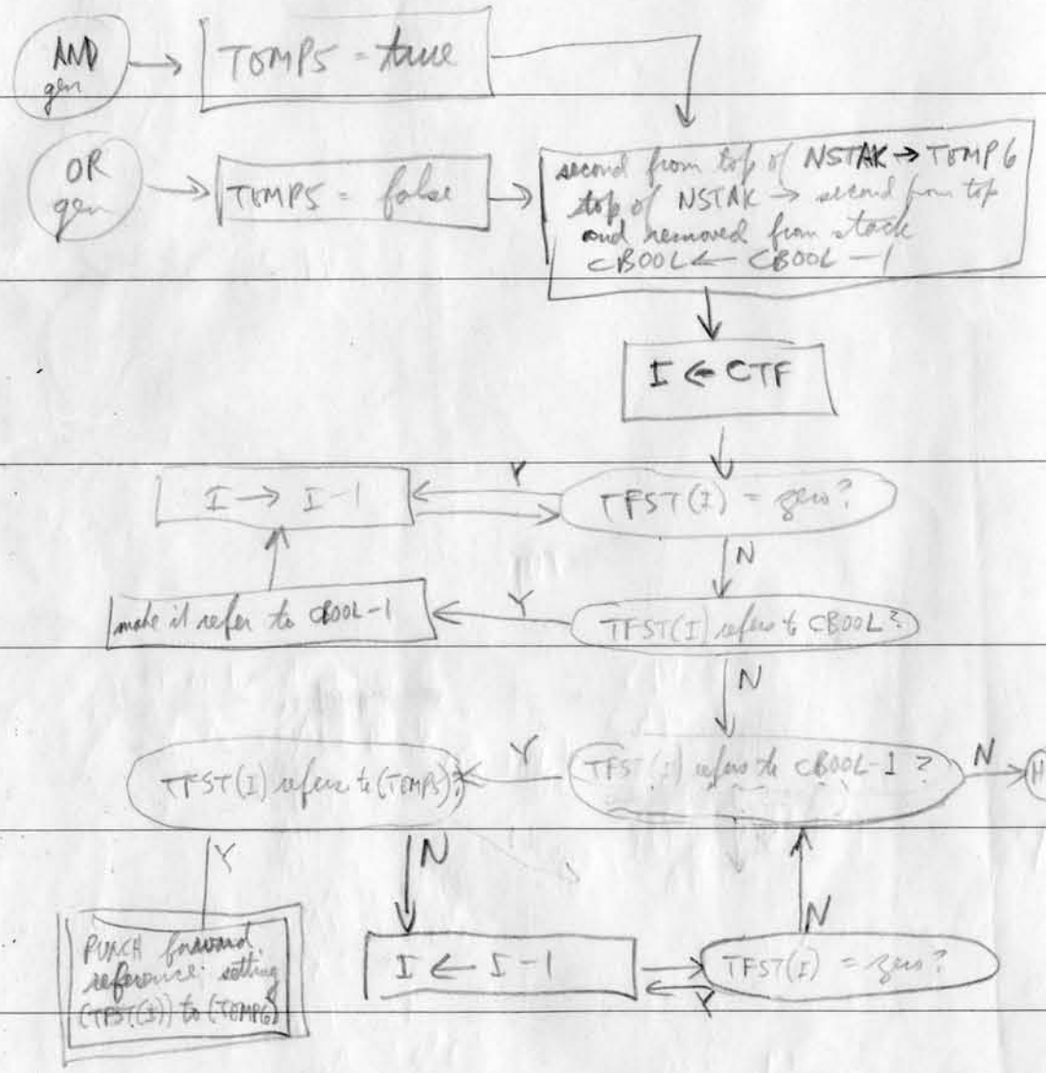
19

17




```
NOT  LDB  CTF
3    -CAD  TFST
     NOR  1F  7
     SRT  8
     SUB  CBOOL
     ADD  ONE
     CNZ  2F  7
     CAD  3F  7
     SLT  4
     CLA  7
     -ADA  TFST
     -STA  TFST
1    DBB  3B  7
2    CUB  HIER
3    (OP)
```

```
NNOR LDB  TEMP7
     CUB  HIER
4    -CAD  TFST  NOR 2F 7
     SRT  8
     SUB  CBOOL
     ADD  ONE
     CNZ  3F  7
     SLT  4
     SUB  TEMP5
     CNZ  2F  7
     CUB  BOOT
2    DBB  4B  7   = SUB HIER
     CAD  TEMP6
     SLT  ST
     STA  TEMP2
     CAD  FREEL
     SUB  ONE
     STA  FREEL
     STA  TEMP1
     BA
     STA  CNAME
     CAD  2+  7
     CUB  PUNCH
     CUB  1B
```




```
AND  CAD  ONE
OR   STA  TEMP5
1    STA  CNAME
     -CAD  NSTAK
     STA  TEMP6
     SRT  4
     SUB  1-  7
     CNZ  2F
     CAD  CBOOL
     SUB  ONE
     STA  CBOOL
     -CAD  NSTAK+1
     STA  OR  7
     DBB  11  7
     BA
     STA  CNAME
     CAD  OR  7
     STA  NSTAK
     CUB  3F
```

```
2    CUBR  ALARM
3    -STA  NSTAK
     LDB  CTF
3    -CAD  TFST
     NOR  1F  7
     SRT  8
     SUB  CBOOL 7
     CNZ  2F  7
     ADD  CBOOL 7
     SUB  ONE
     SLT  8
     -STA  TFST
1    DBB  3B  7
2    CUB  4F
     CBOOL  15
```

1000 001

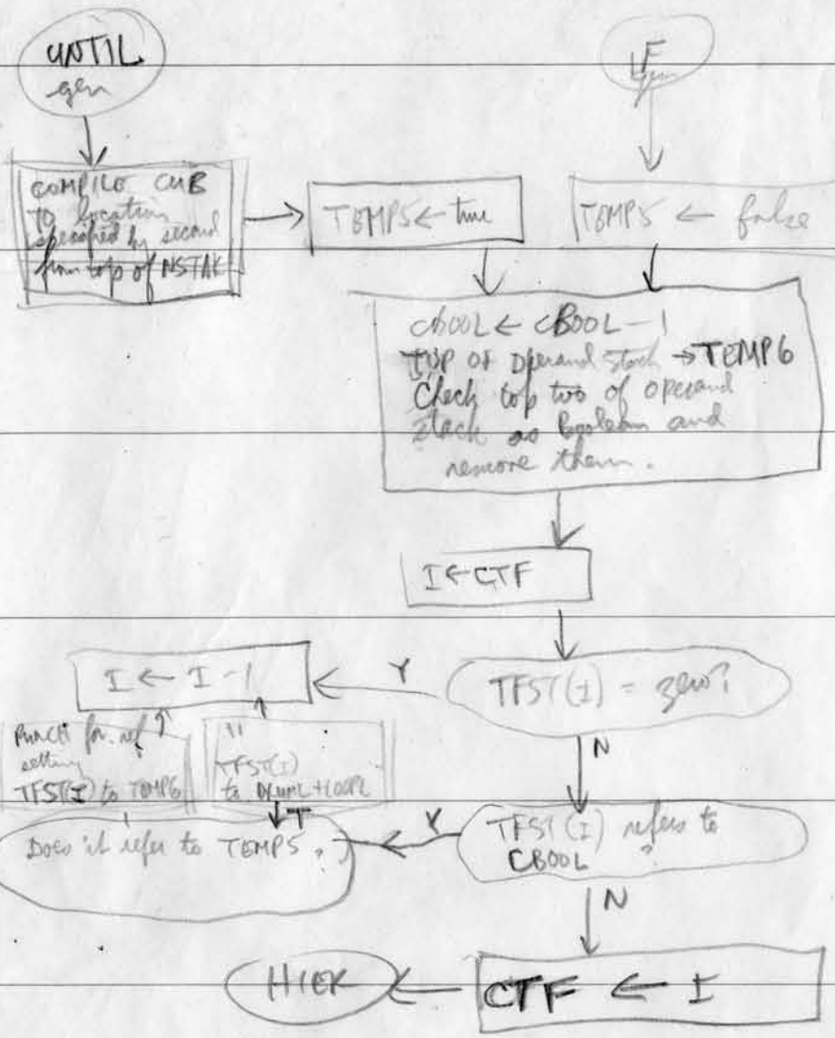
1000 001

1000

1000 001

1000 001

CTF counter starts at zero.



Colon routine. If no forward references have been made to this label, we have erroneously classified it as a variable.

~~DEFINE definition, defines label as~~

```
UNTIL  LDB  CNAME
       CAD  NSTAK
       SUB  1F 7
       STA  TEMP1
       CAD  2F 7
       CMB  OUTP7
       ~~AI  UF~~
       ~~CU  #6~~
1      [260  STA  COLUM-  (1CMB)  ]
       ~~CMB  3B~~
       CMB  1+
       CAD  ONE
IF     STA  TEMP5
       CAD  CBOOL
       SUB  ONE
       STA  CBOOL
       LDB  CNAME
       CAD  NSTAK
       STA  TEMP6
       SRT  4
       SUB  1- 7
       CNZ  3F
       BA
       SUB  2F 7
       STA  CNAME
       LDB  CTF
       CMB  4F
2
3      CMBR  NAME
       COLON HERE
       LDB  TEMP7
1
4      CAD  TEST
       INDR 2F 7
       SRT  8
       SUB  CBOOL
       CNZ  5F 7
7 7800 SLT  4
       SUB  TEMPS
       CNZ  6F 7
       CAD  DRUML
       ADD  LOOPL
       CMB  7F
       ~~AI  1  ST  8~~
6      CMB  8I
5      BA   PRSC4
       STA  CTF
       CMB  HIER
2      DBR  4B 7
```

```
BOOT   CAD  3F 7
       STA  6F 7
8      CAD  TEMP6
7      SLT  4
       STA  TEMP2
       ~~CAD  PROFL~~
       ~~SUB  ONE~~
       ~~STA  PROFL~~
       ~~STA  TEMP1~~
6      BA   1B
       STA  TEMP7
       DRB  6B 7
       ~~BAIF  CMBR  NAME~~  CMB  1+  PUNCH2
3      BA   ANDOR
2      ~~UNT  1B~~
```

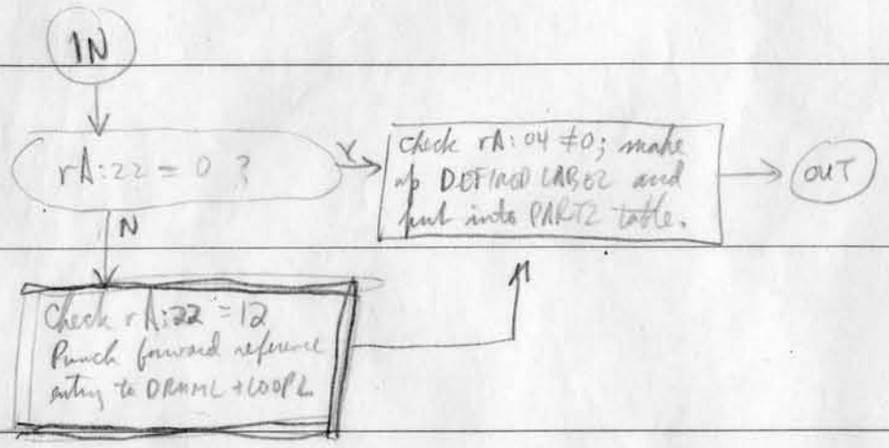
```
COLON  CAD  A1
       ADD  7F 7
       CCB  9F
       CAD  A1
9      ADB  6F 7
       CMB  DFINE
       ~~BOOT  fix~~
```

```
COLON  CAD  A1
       ADD  7F 7
       BOF  1F 7
       SUB  7B 7
       LDB  2F 7
       CMB  DFINE
1      LDB  EQUIV
       CAD  PART2
       SUB  A1
       CNZ  8F
       CAD  A1
       EXT  3F 7
       SUB  4F 7
7 7800 CNZ  2+ 7
       CMB  5F
       CMB  6F
1      LDB  EQUIV          2   CMB  PRSC4
       CAD  PART2          3   1000
       SUB  A1             4   6000
       CNZ  8I
       CAD  A1             5   CAD  LOOP6
       EXT  3F 7               ADD  ONE
       SUB  4F 7               STA  LOOP6
       CNZ  5F 7               BUN  4F 7
       CAD  LOOP6          6   CAA  HIGHL
       ADD  ONE                ADD  ONE
       STA  LOOPL              STA  HIGHL
       BUN  1F                 LDB  2F 7
5      CAA  HIGHL              CAD  EQUIV
       ADD  ONE                CMB  DFINE
       STA  HIGHL          2   CMB  PRSC4
1      CMB  9B
       3  1000
       4  6000
8      CMBR  NAME           8   CMBR  NAME
```

DFINE subroutine. Used to set the equivalent of a label to DRUML + LOOPL and to fix up forward references if necessary.

Entry: rB = exit rA = code (if rA:22 = 0, the symbol has never before occurred. if rA:04 = 0 this is an error condition, the label having been previously defined rA:04 should contain a PART2 reference).

Temp Storage used EXIT0 TEMP3 DRUML FREFL
Subroutine used PUNCH

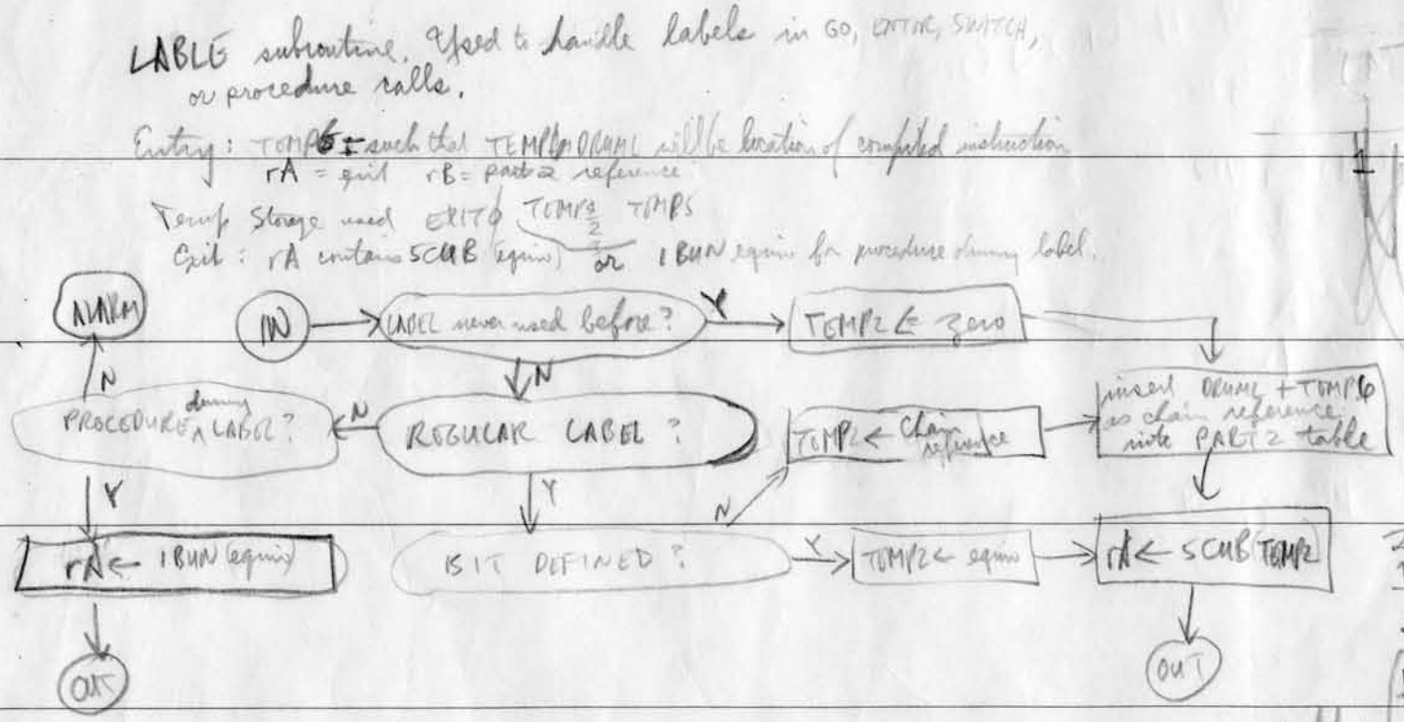



```
DFINE  STA  TEMP3
       BA
       ADD  [illegible]
       STA  EXIT0
       CAD  TEMP3
       SRT  8
       NZ   2F  7
1      CUB  DFINE  7   6F
       SUB  2F  7
       CNZ  [illegible]   7F
       [illegible]
       CAD  7B  7
       CUB  PNCH  1
2      =12=
```

LABLE subroutine. Used to handle labels in GO, EXTRN, SWITCH, or procedure calls.

Entry: TEMP6 = such that TEMP6 0 DRUML will be location of compiled instruction
rA = exit rB = part2 reference

Temp Storage used EXIT0 TEMP3 TEMP5
Exit: rA contains SCUB (equiv) or IBUN equiv for procedure dummy label.




```
LABLE  STA  EXIT0
       [illegible]
       CAD  PART2
       NOR  3F  7
       SRT  8
       SUB  2F  7
       CNZ  3F  7
       SLT  8
       BUN  EXIT0
3      ADD  ONE
       CNZ  4F
1      BA
       ADD  5F  7
       BUN  EXIT0
2      =13=
5      (SCUB) 9000
4      CUBR ALARM
       CLR
       SLT  4
       ADD  1F  7
       ADD  TEMP3
       LDB  TEMP3
       STA  PART2
       CAD  TEMP2
3      ADD  2F  7
       BUN  EXIT0
4      SUB  ONE
       CNZ  4F
       SLT  8
       BUN  EXIT0
1      =120  0=
2      =(SCUB) 9000
4      CUBR ALARM.
```

```
1      SUB  2F  7
       CNZ  7B
       SRT  1
       [illegible]
       ADD  LOOPL
       [illegible]
       [illegible]
       CAD  FREFL
       [illegible]
       CAD  1F  7
       CUB  PNCH  1
2      =12=
1      CUB  GB
```

```
       CLR  4B
       LDB  TEMP3
       DEB  2F  7
       CUBR ALARM
       CAD  3F  7
       ADD  DRUML
       ADD  LOOPL
       CLRA 3
       STA  PART2
       LDB  EXIT0
       CUB  0
3      120000
```

Enter → TEMP1 ← LOOPL+1, LABLE on A2 → ACALL → PRSC4

60 → TEMP1 ← LOOPL, LABLE on A2 → COMPILE RESULT → OUTP1 → PRSC4

Switch: At time of entry
A0 = ?
A1 = 0
A2 = 5

SWITCH generate → take top off operand stack → GET → Is it floating? Y → Is it double precision? Y → GET IT → COMPILE FAD ±5810000000

Is it floating? N → T ← T-1, COMPILE STA T, LDB T, - RUN NEXT, OUTP7

Is it double precision? N → COMPILE FAD ±5810000000

COMPILE NEXT: CUB 0000 AND ONE REFERENCE FOR EVERY LABEL IN THE SWITCH STATEMENT

COMPILE FORWARD BRANCH ENTRY FOR NEXT, SET AK = dimension → FIXT2 → PRSC4

continued on next page.

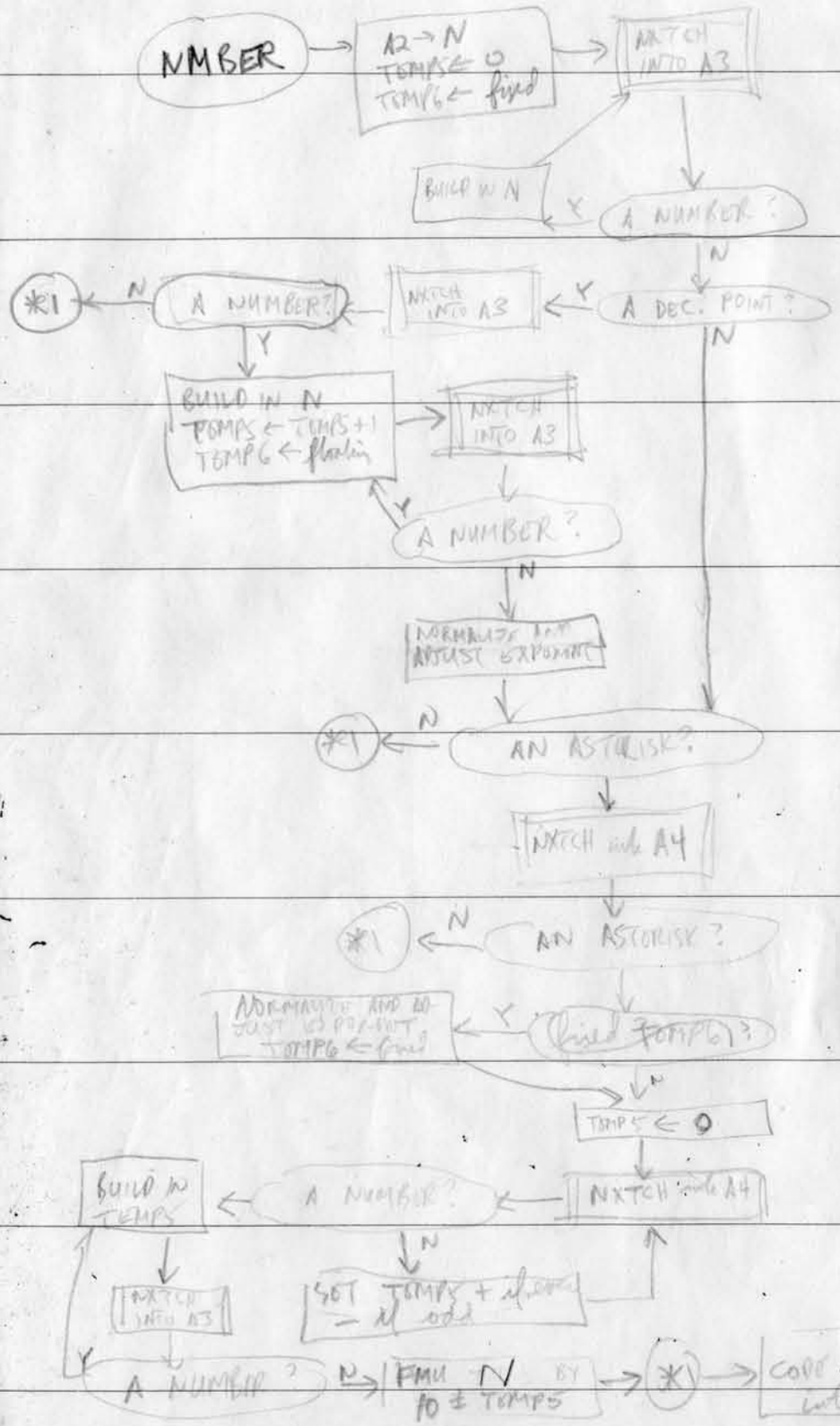
```
ENTGO  CAD 1F 7
       LDB EQUIV
       CUB LABLE
2      LDB A1
       DRB 2F 7
       STA TEMP1
       CAD 3F 7
       CUB OUTP7
1      CUB 2B
2      LDB 3F 7
       SUB 4F 7
       CUB ACALL
4      14 0000
3      CUB PRSC4
```

```
ENTGO  CAD LOOPL
       LDB A1
       DRB 2F 7
       ADD ONE
       STA TEMP1
       CAD 1F 7
       LDB EQUIV
       CUB LABLE
2      LDB A1
       DRB 2F 7
       LDB 3F 7
       CUB ACALL
1      CUB 2B
2      CAD 3F 7
       CUB OUTP7
3      CUB PRSC4
4      CUB 9B
       14 0000
```

```
SWICH  BA 1F
       STA CNAME
       -CAD NSTAK+1
       STA A0
       LDB SWICH 7
       CUB GET
1      CAD A0
       EXT 1F 7
       CNZ 6F 7
       CAD 2F 7
       LDB E7 7
       CUB CONOT
4      CAD 6F 7
       CUB OUTPT
1      00110
2      5810
6      CUB 9F
       HLT HR
5      DFIN67 IN HERE
```

```
SWICH  BA 1F
       STA CNAME
       -CAD NSTAK+1
       STA A0
       EXT 1F 7
       CNZ 6F 7
       CUB
       CAD A0
       EXT SWICH 7
       CNZ 7F
5      CUB 5F
1      00110
7      CAA A0
       LDB SB 7
       CUB GET
       CUB 9F
6
3      CAD 1F 7
       SFT
       CAD A0
       SLT 10
       LDB DFIN67
       CUB CONOT
1      5810
       FHLO HR
       CUB 7F
```

```
9      CAD LOOPL
       ADD ABC
       ADD 1F 7
       BDF 2F 7
9      CAD 2F 7
STADB  STA EXIT1
       CAD 1F 7
       STA TEMP1
       CAD 4F 7
       CUB OUTPT
       CAD 5F 7
9      STA TEMP1
       CAD EXIT1
       CUB OUTPT
2      CUB 3F
       194001
1      CUB 9B
4      224001
5
3      CAD DRUML
       ADD LOOPL
       ADD ABC
       ADD ONE
       STA A0
       ADD 4F 7
       STA TEMP1
       CAD 5F 7
       CUB OUTP7
4      2201 RUN 0000
5      CUB 1?
       CAD A0
STORE  STA DRUML
       STA TEMP1
       STA SWICH
       CAD 6F 7
       STA TEMP2
       CAD 2F 7
       CUB PUNCH
1      CUB 0000
       CAD A1
       SUB 3F 7
       CNZ 4F
       CAD A2
       SUB 5F 7
       CNZ 4F
       LDB ONE
       CAD 6F 7
       CUB NXTCA
       CUB 7B
3      CAD A3
       SUB 1F 7
       CNZ 9F
       CUB 1?
       CUBR ALARM
```




```
1   CAD 2F 7
    CUB BUILD
3   LDB TEMP5
    CAD ONE
    STA TEMP6
    CAD 4F 7
    CUB LABLE
5   STA TEMP2
    CAD DRUML
```

```
CAD 1F 7
STA TEMP1
CAD BIT 7
STA SWTCH
CAD 2F 7
STA PPOSW
```

```
CAD A0
STA DRUML
CUB 8F
```

```
1  # (CUB 0) 4F
2  CUB
3  CUB PPOIN
4  CAD A1
   STA TEMP1
   CAD A2
   SUB 1F 7
   CNZ 8F 7
   CAD 4F 7
   CUB OUTP7
   CAD 3F 7
   CUB FMUL
8  CAD 3F 7
   CUB OUTP7
1  CUB SWFN1
4  CUB PRSC4
3
```

```
STA TEMP2
CAD A3
SUB 1F 7
CNZ 2F 7
CAD DRUML
ADD ONE
STA DRUML
CAD TEMP2
STA TEMP1
CAD 3F 7
CUB OUTP7
2  CAD DRUML
   ADD ONE
   STA DRUML
1  CUB 1+
3
```

```
1  CAD DRUML
   SUB 5
   ADD A0
   STA TTEMP2
   CAD 1F 7
   STA A0
   LDB 21
   CUB PNCH
1
2
4  CUB
```

```
NMBER  STC N
       STA TEMP5
       CAD 1F 7
       STA TEMP6
       LDB ONE
       BUN 1F 7
3      ADD 4F 7
       BOF 5F 7
       ADD 6F 7
       CCB 9F
       CAD 8F 7
       CUB NXTCH
1      =0010 0
5      SRT 1
       CAD N
       SLT 1
       STA N
7      CAD 3F 7
       CUB NXTCH
2      CUB 3B
4      9 920
6        78
8      CUB 1F

1      ADD 2F 7
       BOF 3F 7
       CUB NFIN
4      ADD 2F 7
       BOF 3F 7
       CUB 4F
6      SRT 1
       CAD N
       SLT 1
       STC N
       STA TEMP6
       CAD TEMP5
       ADD ONE
       STA TEMP5
       CAD 6B 7
       CUB NXTCH
       9 920
```

```
6  STA 9F 7
4  CLR 60
   CAD N
   NOR 4F 7
   SRT 10
   CAD 4B 7
   SSC
   SUB TEMP5
   SLT 8
   STC N
4  CAD A3
9  SUB 1F 7
   CNZ 2F 7
   1B
   CAD 3F 7
   CUB NXTCH
1  CUB NFIN
2  CUB 1+
3  SUB 1F 7
   CNZ 2F 7
   STA TEMP5
   STA TEMP6
   CAD 3F 7
   CNZ TEMP5
7  SUB 4F 7
   CAD 1+ 7
   DBB 1+ 7
   CUB NXTCH
   CAD 5F 7
3  CUB 6B
   CUB 1F
1  CUB
4  CUB NFIN
2  CUB 1B
   CUB
5
```

```
1  ADD 2F 7
   CCB 3F 7
   SLT 9
   STA 2F 7
   AA
   SUB 2F 7
   <IRA 0
   STA TEMP5
   CAD 4F 7
   CUB NXTCH
5  ADD 2F 7
   BOF 3F 7
   CUB 7F 5B
4  CUB
3  SRT 1
   CAD TEMP5
   SLT 1
   STA TEMP5
   CAD 6F
   CUB NXTCH
6  CUB 5B
7  CAD N
   SRT 8
   ADD TEMP5
   SLT 8
   STA N BUN2F
NFIN  CAD 1F 7
      ADD TEMP6
      STA A2
      CUB CODE
1     050
```

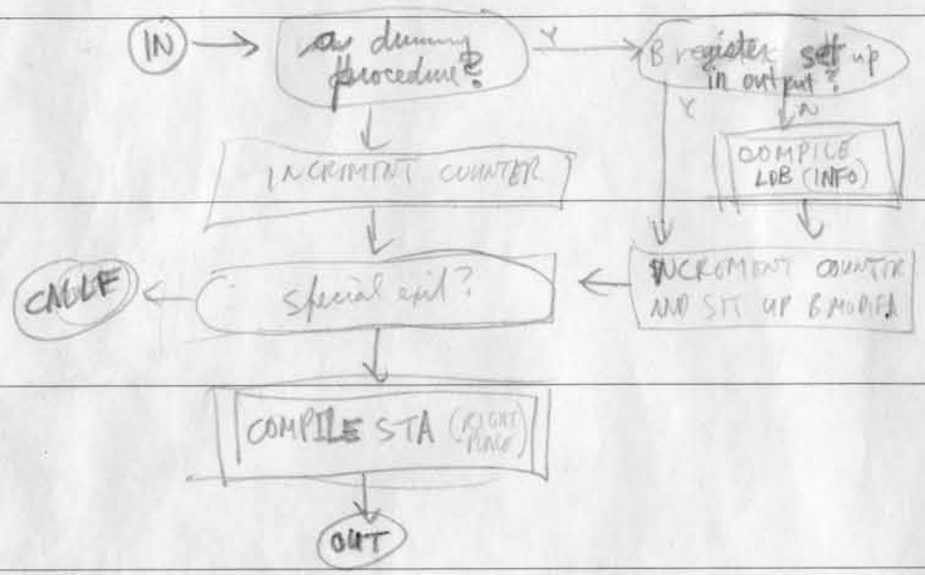
CAD TEMP7

28

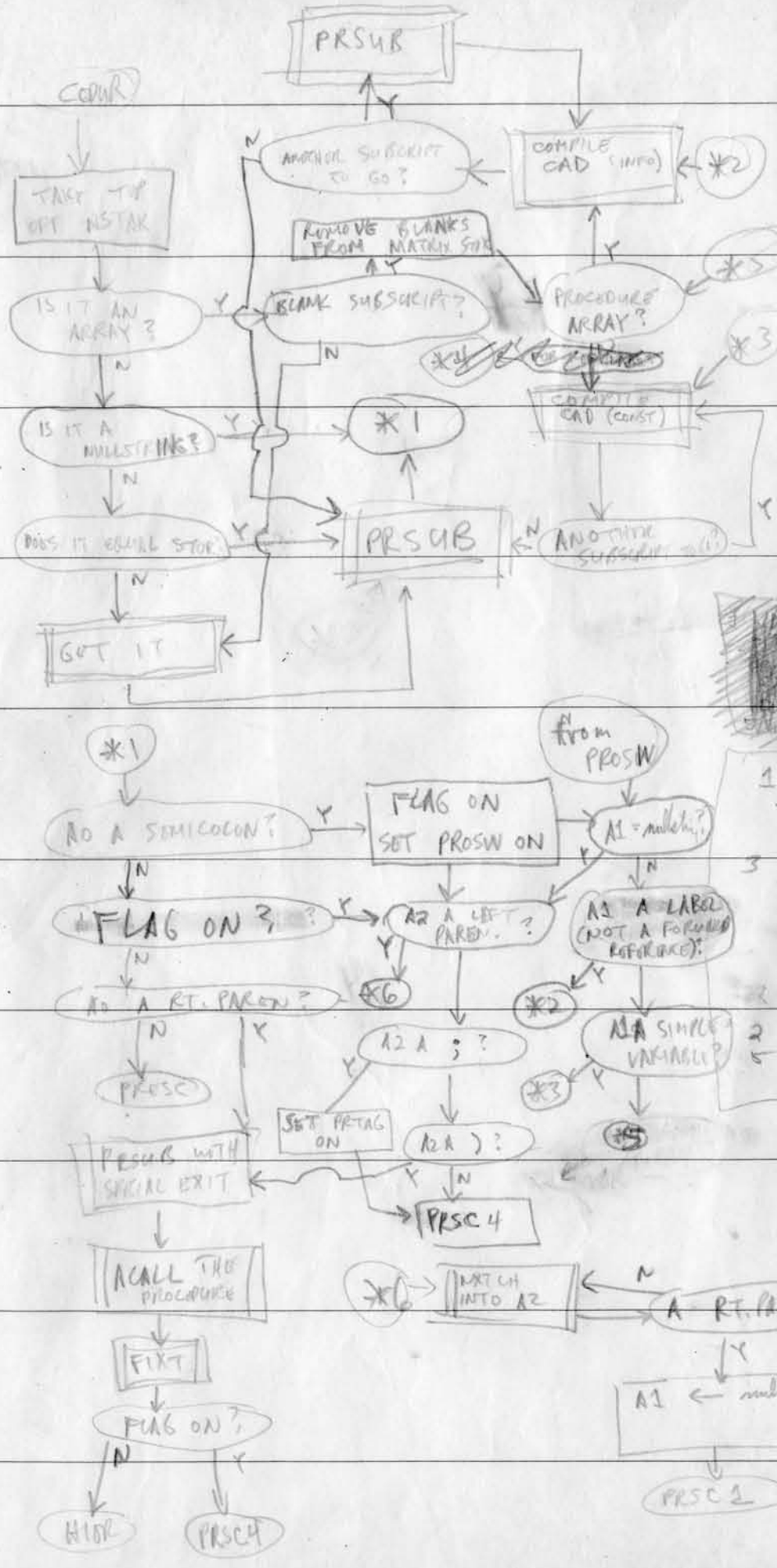
Procedure Callout. Dflp until the first semicolon, the routine branches from the normal scanner, an exit being made to CEDUR whenever a , ; or) occurs in NO when a procedure callout is next to the top of OPST.
When the semicolon is sensed, we switch out of normal operating mode and use the prescanner and A1, A2 for information until the end of the procedure.

PFSUB This subroutine puts out the command which stores the input arguments in the proper place for the procedure.
Entry: A= exit. If b negative a special exit is made to CALLF before compiling the STA instruction.

Temp Storage EXIT1 RB TEMP5
Subs used OUTPT.

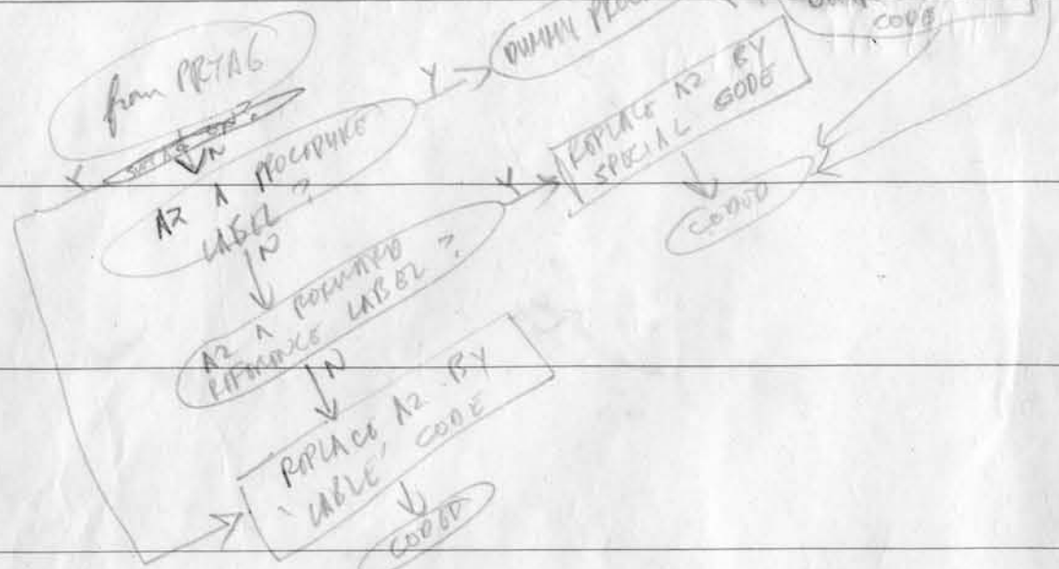


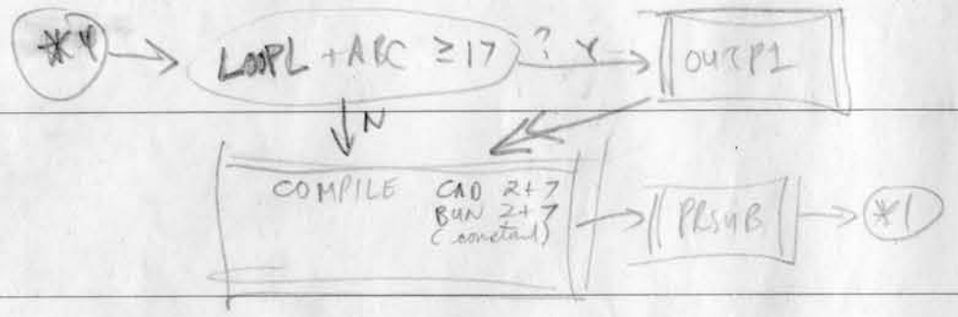

```
PFSB1 CSA 6+7
PFSUB STA EXIT1
      LDB COP
     -CAD OPST+1
      STA TEMP5
      ADD 1F 7
      CCB 2F
      CAD TEMP5
      ADD ONE
     -STA OPST+1
      EXT 3F 7
   8  OSD EXIT1
      CCB CALLF
      ADD 4F 7
1 6700 STA TEMP1
      CAD EXIT1
      CUB OUTPT
   3  0 — 01111
   4     18 9999
   2  CAD RB 7
      ONE 2F 7
      CAD TEMP5
      EXT 3F 7
      STA TEMP1
      CAD 4F 7
      STA RB
      CUB OUTPT
   2  LDB COP
      CAD TEMP5
      ADD 5F 7
     -STA OPST+1
      SRT 4
      EXT 6F 7
      CUB 8B
   3     22 1111
   4  CUB 2B
   5     1 0000
   6  20 — 01 1
RB
```




```
CDDUR  CAD  OPST+1
       STA  OPST+1
       LDB  CNAME
       DBB  E1A+7
PCDRE  BA   8F
       STA  CNAME
       CAD  NSTAK+1
       STA  TEMP6
       ADD  3F=7
       CCB  1F
       CAD  TEMP6
       SUB  2F 7
       CNZ  4F 7
       CUB  9F
       CAD  TEMP6
       LDB  PCDRE+7
       CUB  GET
       SRT  8
       STA  2F 7
       LDB  TEMP6
       CAD  PART2
       STC  TEMP6
       LDB  2F 7
       RUN  34-7
       CUB  2F
       CUB  1F
       CUB  4F
       CUB  3F
       CAD  ONE
       STA  TEMP7
       CAD  TEMP6
       EXT  5F 7
       LDB  2F 7
       CUB  CONOT
       STA  TEMP1
       CAD  7F 7
       CUB  OUTPT
       CUB  6B
       CAD  TEMP7
       CNZ  2+ 7
       CUB  8F
       CAD  6F 7
       CUB  PRSUB
       =1111=
       CUB  5B
       CAD  TEMP6
       ADD  ONE
       STC  TEMP6
       CUB  3B
       CUB  2B
       CAD  1F 7
       CUB  PRSUB
       CAD  A0
       SUB  2F 7
       CNZ  3F 7
       CUB  8F
       SUB  5F 7
       CNZ  6F 7
       CUB  PRSB1
       CUB  9B
       semicolon
       SUB  ONE
       CNZ  4B 7
       CUB  SEVEN
       CUB  PRESC
       jmp - (semicolon+1)
       CAD  1F 7
       STA  PROSW
       CAD  2F 7
       STA  A0
       CAD  A1
       SUB  5F 7
       CNZ  2+ 7
       CUB  SEVEN
       CUB  8F
       =1111=
       CAD  AZ
       SUB  1F 7
       CNZ  2F 7
       CUB  6F
       SUB  5F 7
       CNZ  7F 7
       CUB  PRSB1
       SUB  3F 7
       CNZ  4B 7
       CAD  2F 7
       STA  PRTAG
       CUB  PRSC4
       left paren
       semicolon
       left paren
       nullstring
       CUB  PROIN
       SUB  ONE
       LDB  1F 7
       CUB  ACALL
       CAD  COP
       SUB  3F 7
       STA  COP
       CAD  A0
       SUB  3F 7
       CNZ  4F 7
       CUB  SWFIN
       CUB  6B
       semicolon+1
       LDB  COP
       CAD  OPST+1
       SRT  4
       CUB  FIXT4
       SUB  2F 7
       CNZ  6F 7
       CAD  2F 7
       STA  A1
       CUB  PRSC1
       CAD  1F 7
       CUB  MATCH
       left paren
       nullstring
       CUB  SR
```

There's a problem in case a label for the procedure is a forward reference.






```
PROIN   ADD 1F 7 5400 BA 0000
        CCB 5F          OSD SWTAG 7
        LDB A2          BOF 4F 7
        DRB 2F 7
        CAD A2
        NOR 7F 7        LDB TEMP3
1 6000  CAD 3F 7        4 CAD ONE
        CUB LABLE       STA JUMP6
        [illegible]
        7 CAD TEMP3
        2 ADD 5B 7
        [illegible]
        3 CUB CODE1     <- SWTAG=0
        5 SRT 8
          CNZ 2+ 7
          STA 2F 7
          CAD TEMP2
          EXT 5F 7
          ADD 2F 7
          CUB CODE1
        2 170 ---- 0
        3 220000 1111

WORT    CAD LOOPL
        ADD A2
        ADD 6F 7
        CCB 1F 7
      7 CAD 2F 7
        STA TEMP1
        CAD 3F 7
        CUB OUTPT
      5 CAD 4F 7
        STA TEMP1
        CAD 5F 7
        CUB OUTPT
      6 9 983
        (CAD 2+7)
      2 3 CUB 5B
        (BUN 2+7)
        4
        5 CUB 1+
          CAD LOOPL
          STA TEMP6 STA TEMP7
          LDB A1
          CAD 2F 7
          CUB LABLE
        1 CAD 4F 7
          CUB OUTP1
        2 CUB BAKKK
        1 4 CUB 7B
```

```
SLOBE

PROIN   ADD 1F 7
        BOF 5F 7
        CAD 3F 7
        CUB LABLE
      5 SRT 8
        CNZ 2+ 7
        STA 2F 7
        CAD A2
        EXT 4F 7
        ADD 2F 7
      3 CUB CODE1
      2 170 ---- 0
      4 220 ---- 0
```

```
HURT    STA TEMP7
        CAD ONE
        LDB 2F
        CUB FILL2
      2 CAD 3F 7
        STA TEMP1
        CAD 4F 7
        CUB OUTPT
      1 CUB 2B
      2 CAD 5F 7
        STA TEMP1
        CAD 6F 7
        CUB OUTPT
      4 CUB 2B
      2 CAD A1
        CUB BAKKK
      3   (CAD 2+7)
      5   (BUN 2+7)
      6 CUB 2B

SWFIN   CAD =
        STA A0
        CAD =
        STA SWTCH
        CAD =
        STA PROSW
        CAD = CUB PRSCH
        CUB FIXT3
        =
        =
        =
        = CUB PRSCH
```

Declaration of Procedures, Subroutine, Format, Input, Output: (all ~~[struck out]~~ go with main scanner for completion)

```
DECLR CAD A1
      ADD DRUML
      ADD LOOPL
      STA A1
      CAD 1F 7
      STA TEMP1
      CAD 2F 7
      CUB OUTP7
3     CAD A2
      CNZ 2+ 7
      CAD EQUIV
      LDB 3F 7
      CUB DFINE
1     (1CUB 0)
2     CUB 3B
3     CUB 4F

4     CAD A1
      SKT 8
      ADD 2F 7
      STC 2F 7
      LDB DRUML

2     BUN 89- 7     CUB 3B
      CUB FMT       CUB 3B
      CUB SBR   —  CAD THIN
                    STA 7
      LDB EQUIV
      BA
      SKT
      SKT 2F 7
      ADD 4F 7      LDB EQUIV
      CUB 1F  0=    -STA PART2
      =346 0        -SA PART1
4
      SWFN 1 LDB 3F 7
             CAD A0         CAD LOOPL
             CUB PNCH4      ADD ONE
      1 -STA PART1  BA SWFIN  STC LOOPL
                2   STC THISE 8 CNZ 7F
      8 STA TEMP5
        STA TEMP6
      2 ADD 1F 7
        CCB 3F
      9 CAD 4F 7
        LDB NUM
        CUB NXCH1   PROCC =100=
      1 9  93
      7 CUB 3B
      3 CNZ 5+ 7   6 CAD TEMP5
                     ADD 5F 7
        CAD 1F 7     CUB 8B
        CUB BUILD    4 CUB 2B
        CAD A3
      2 CNZ 2F 7
      4 CAD 3F 7
      7 LDB NUM
        CUB NXCH1
        CUB 2B
      1 =3=
      5

13: BA 4F
    STA PROR1
    CAD ONE
    STA LOOPL
    CUB PRSC1

CAD PROCC 7
STA PROCT
ADD PROCC 7
STC PROCC

2   ADD 1F 7
    CCB 8F
    CAD LOOPL
    ADD ONE
    STA LOOPL
    LDB TEMP6
    CUB 2F
1   =9= 9A
3   CUB 1B
2   -ADD (CODE)
    LDB TEMP3
    -STC PART2
     CAD TEMP5
     STC TEMP6
     CAD TEMP5
     STA DUMMY
     1B
     BA
     STC TEMP5
    LDB A3
    -BUN 6- 7
    CUB 6B
    CUB 9B
    CUB SBR1
6   CAD TEMP5
    ADD 3F
    STA TEMP5
    CNZ 6F 7
    CUB MCH2
6   ADD 3F 7
    CAD TEMP6
    ADD ONE
    STC TEMP6  STA LSTCH
    CAD 2F
    CUB NXCH1
    CUB 1B
    ADD 3F 7
    CUB 1B
3   9   91
7   CUBR ALARM
2   CUB 1B
SBR CAD
    STA
    SBR1  STA 6- 7
          LDB 7- 7
          ADD 1F 7
          STC TEMP1
          -STA PROCX
          CAD 2F 7

    CUB OUTPT
    (STA 9151)
1.
2   CUB PRSC3

CODE 13 0038 9999
     33 5
     38 9999999
     37 9999999
     21 9999999
     37 9999999
     36 9999998
```

2nd note made 1995: I don't recognize this handwriting - DEK

```
ARRAY  F(50) = (1, 1); UP(2,3), LOW(2,3);
  FOR  I = (3, 1, 20); F(I) = F(I-1)
  + F(I-2);

FOR N = (5, 1, N+1); BEGIN

K=1; UNTIL  N LEQ F(K+1); K=K+1; X=1;
  Y = 2; Z = 3; SUM = 0; IF + N,
  LEQ f(K) + f(K-2); Go To CASE1;
UP(1,1) = f(K-2); UP(1,2) = 0; UP(1,3) = N-F(K);
LOW(1,1) = 1; LOW(1,2) = 1; LOW(1,3) = 2
UP(2,1) = 0; UP(2,2) = 0; UP(2,3) = f(K+1) - N;
LOW(2,1) = 1, LOW(2,2) = 1; LOW(2,3) = 1;
  SUM = UP(1,3) + UP(1,3) + UP(2,3);
  Go To HELL;

CASE1.. UP(1,1) = N - f(K); LOW(1,1) = 2;
        UP(2,1) = f(K-2) - UP(1,1); LOW(2,1) = 1;
        UP(1,2) = UP(2,2) = 0; LOW(1,2) = LOW(2,2) = 1;
        UP(1,3) = f(K-1), UP(2,3) = 0;
        LOW(1,3) = LOW(2,3) = 1;
        SUM = UP(1,1) + UP(1,1) + UP(2,1);
```

```
HELL.. IF F(1,X)+F(2,X) EQL 0; GO TO END;
       J=1;
FIBI; FOR I = 1,2; BEGIN IF UP(I,X) NEQ 0;
IF UP(I,X) LEQ UP(1,Z); (UP(1,Z) = UP(1,Z) - UP(I,X);
   SUM = SUM + UP(I,X)(LOW(I,X) + LOW(1,Z));
   UP(J,Y) = UP(I,X); LOW(J,Y) = LOW(I,X)
   + LOW(1,Z); J = 2; GO L ); IF UP(1,Z)
   UP(I,X) = UP(I,X) - UP(1,Z);
   SUM = SUM + UP(1,Z)(LOW(I,X) + LOW(1,Z));
   UP(1,Y) = UP(1,Z); UP(1,Z) = 0; LOW(1,Y)
   = LOW(I,X) + LOW(1,Z); UP(2,Z) =
   (UP(2,Z) - UP(I,X); LOW(2,Y) = LOW(I,X)
   + LOW(2,Z); SUM = SUM + UP(I,X) LOW(2,Y);
   UP(2,Y) = UP(I,X); UP(I,X) = 0; L.. END END;
   J = X; X = Z; Z = Y; Y = J;
   IF PCS(1); WRITE(; OT, FR); GO TO HELL;
I=IBI.. WRITE(; DATA, FR); END;
OUTPUT OT (FOR J=1,2; FOR I=1,2,3; (UP(J,I), LOW(J,
   DATA (N, K, SUM, N.J, J);
I=2; J=1; UNTIL N LEQ I; (I=I+I; J=J+1);
FORMAT FR( 6 I 15, W0; 6 I 15, 2W0)
```

```
RETRN  CAD 1F 7
       ADD RETRO 7
       STA TEMP1
       CAD 2F 7
       CUB OUTP 7
   1   (SUN 9151)
   2   CUB PRSC1
RRCDR  CAD TMIN
       STA I
       STA PRST
       LDB RETRO 7 STA RETRO
SBRTN  LDA 0 7
       CAD PROCX
       CNZ 3+ 7
       BA  4F
       ADD 3F 7
       LDB 2- 7
       CUB DFINE
RETRO  0
   3   =151= THISE 0
       3
   4   CAD 3F 7
       STA TEMP1
       CAD 4F 7
       CUB OUTP 7
   3   =HLT 1359=
   4   CUB FORMT
```

```
RETRN  LDB ONE
       -CAD PROCX
       CNZ 2+ 7
       LRB 2- 7
       SRT #5
       CAD DRUML
       ADD LOOPL
       STA PROCX
       CAD 1F 7
       SRT #5 ADD 1F
       STA TEMP1
       CAD 2F 7
       CUB OUTP 1
       CUB PRSC2
```

```
       STA 2F 7
       OSD 2F 7
       CCB (return)
```

```
INTAR  SRT 8
       CNZ 1F 7
   INT CAD 2F 7
       LDB ONE
       CUB NXTCH
   1   CUB AR
   2   CUB 1+
       SUB =7=
       CNZ 2+ 7
       CUB PRSC1
       ADD =9 980=
       BOF 2+7
       CUB INT
       CAD 1F 7
       CUB BUILD
   2   LDB TEMP3
       -CAD PART2
       EXT =11 02 1111=
       -STA PART2
       CAD A3
       CUB 2F 7 16
   1   CUB 2B
       =9 980=
       =1121 1=
   2   ADD =9 98=
       CCB INT
       STA A3
       CAD R
       EXT =1111 00
       STA R+10
       CUB BLD1
```

make into a subroutine with ARRAY

STA TEMP2 STA TEMP3

last of subroutine

```
   3   =HLT 1359=
   4   CUB FORMT
```

```
       CAD 3F 7
       STA TEMP1
       CAD 4F 7
       CUB OUTP 7
       CUB FORMT
```

```
   4   CUB PNCH1   1 CUB 2F
   2   CAD 3F 7
       STA TEMP1
       CAD 4F
       CUB OUTPT
   4   CUB FORMT
       =STOP 0=7=
   3   SUB =9 2+ CUB PRSC1
       ADD =9 967
       CCB 3F
       CAD 2F 7
       LDB ONE
       CUB NXTCH
       9 960
   2   CUB 1B
   3   CAD 2F 7
       CUB BUILD   CAD ONE
   4   STC TEMP7   STA TEMP6
   5   STA TEMP5
       CAD 5F 7
       LDB ONE
       CUB NXTCH
       ADD =9 920=
   6   CUB 7F 7
       SUB =9 920 + comma =
       CNZ 8F 7
       CAD TEMP5
       STC TEMP6
       CUB 9B   STA TEMP7
   7   SRT 1
       CAD TEMP5
       SLT 1
       CUB 8B
```

```
   8   CUB 1F=
```

```
   1   SU TEMP7
       CNZ 2+ 7
       SU TEMP6
       MUL TEMP5
       SLT 10
       ADA HIGHL
       STA HIGHL
       SUB ONE
       SUB TEMP6
       LDB TEMP3
       -CAD PART2
       EXT =0010
       CIRA 10
       ADD TEMP6
       SRT 4
       ADD =35=
       ADD TEMP7
```

SRT 4

SLT 8
-STA PART2
CUB AR

32

A3 must be a left pair

```
DEFN   SRT 4 8 7
       STA TEMP1
       LDB A2
      -CAD  0
       OSD  7+ 7
       BOF  4+ 7
       SUA  HIGHL
       STA  HIGHL
      -STA  0
       CAD  TEMP1
       CIRA 3
       SLT  4
      -ADA  0
       ADD  1F 7
       LDB  TEMP3
      -STA  PART2
       CUB  CODE1
1      3188000000
```

```
FMT    CAD 1F 7
       STA A2
       CAD 2F 7
       STC TEMP6
       STA TEMP7
FMT1   CAD 3F 7
FMT2   STA TEMP5   CAD TEMP7
                   STA STCH1
       CAD 3F 7
       CUB NXCH1
1      left pairs
       =17=
4      ADD TEMP7
       CNZ 2+ 7
       CUB NXCH2
       SUB 2F 7
       CNZ 2+ 7
       STA TEMP7
3      CUB 5F
3      CUB 4B
2      =17=

5      ADD 6F 7
5      ADD ONE
       CNZ 2F 7
       CAD TEMP6
      -ADD 3F 7
       STA TEMP6
       CNZ 3+ 7
       CUB 9F
2      DBB 5B 7
       CAD A3
       SUB 6F 7
       CNZ 3+ 7
       CAD 7F 7
       STA TEMP7
       CUB 4F
6      =6=
3      -17
       +17
7      =11=
```

```
4      CAD A3
       SRT 2          LDB ONE
       CAD TEMP5
       ADD 1F 7
       CEB 4F
       SLT 2
       CUB FMT2
4      SLT 2
       STA TEMP2
       CAD DRUML
       STA TEMP1
       ADD ONE
       STA DRUML
      -CAD 2F 7
       CUB PUNCH
2      CAB PRESC
       CUB FMT1
FRMT1  BA 4B
       STA CNAME
FORMT  LDB COP
      -CAD OPST+1
       SRT 5
       LDB 2+ 7
       CUB PNCH4
       HLT HIER
```

```
       SB 2
       STA TEMP2
       CAD DRUML
       SUB 1F 7
       STA TEMP1L
       CAD 1F 7
       CUB OUTP7
       CAD DRUML
       ADD ONE
       STA DRUML
       CUB FMT1
1      CUB 3F
2      CUB PRESC
```

```
1  830      0
7  CAD TEMP2
   CLR
   NOR 0+7
```

MOVE INTO 'GET' ROUTINE

```
       LDB CNAME
       MSB 1+ 7
       BA
       STA CNAME
FORMT  LDB COP
       CAD OPST+1
       SRT 5
       CAD DRUML
       STA LOOP1L
       SUB 5F 7
       STA TEMP2
       CAD 2+ 7
       CUB PNCH1
       CUB HIER
```

FORMT (generator)

```
(          (
;          multiply )
(
multiply   )  ;
)  ;
```

Need two errors:

1. FUDGE LSTCH before * so all blanks come through
2. * TEMP7 must be higher.

FOR V=E

GET E
ACALL 1F

FOR V=(E1,E2,E3)

GET E1
CALL 2F

ADD V+E2
STA V
2 SUB V-E3
TEST
BOIF 2+?
CALL 2F
ACALL 1F

CCB 2F
or ACALL 1F

2

FCALL here

```
OFOR  -CAD OPST
      ADD 1F 7
      CCB OFOR2
      -CUB REPL1
IFOR  -CAD OPST
      ADD 1F 7
      CCB IFOR2
      CUB IFOR1
1     559900   FCALL
IFOR1 LDB 1F 7
      CUB REPL1
3     CAD A1
      SUB 4F 7
      CNZ 3+ 7
      CAD 5F 7
      BUN 2+ 7
      CAD 6F 7
      STC PHOR3
      LDB CDOOST   -STA PROCX+1
1     BA 3B
      ADD 7F 7
      STA TEMP1
      CAD 8F 7
      CUB OUTP7
2     44010  1  0
4     47260000
5     02181101 00
6     02180101 00
7       CUB 9152
8       CUB PRESC
IFOR2   CAD DRUML
        ADD LOOPL
        STA PHOR1
        CAD 1F 7
        LDB 2F 7
        CUB MAST1
3       CAD 4F 7
        CUB RSULT
5       -STA NSTAK+1
2         1B 3B
          BA 6F
          STA CNAME
          LDB 2- 7
          CUB REPL1
1       5220101026
4         CUB 5B
```

```
FORF  CUB 3F
6     LDB CDOO5F
      -CAD PROCX+2
       SRT 4 10
       LDB CNAME
       1B PRESC
1      BA 8F
2      STA CNAME
       CAD T
       SUB ONE
       STA T
       STA STOR
      -STA NSTAK
       LDB 1B 7
       CUB PNCH1
OFOR2  CAD 2F 7
      -STA OPST
       CAD PHOR3 7
       SRT 8
       CUB RELAT
1      CUB PRESC
2      440   0   PHOR3 = 0
FORF  CAD TEMP7
      EXT 1F 7
      CNZ 2F 7
      CAD 3F 7
      STA TEMP1
      CAD 4F 7
      CUB OUTP3
1     1010   0
3     = (BOF) 7002
2     CAD 5F 7
      CUB FORG
4     CUB FOR5
5     = (CCB) 0 =
FOR5  CAD 3F 7
6     STA TEMP1
      CAD 1F 7
      CUB OUTP3
2     LDB 3F 7
7     CAD 4F 7
      ADD CDOO3F
      CUB ACALL
1     CUB 2B
3     CUB 0 =
4     = 9151 =
```

```
FORIN CAD TEMP7
      LDB 2F 7
      CUB PNCH4
```

```
      LDB CDOO5F 7
     -CAD PROCX
      SRT 4 10
      LDB 1F 7
      CUB PNCH1
2     CAD CDOO5F 7
      CUB SBR1
DOO   LDB CDOO5F 7
1     BA 7B
      SUB ONE
      STA CDOO5F
      CUB DOOIN
      CUB 2B
CDOO  1
```

FCALL LDB 8F 7
BUN 7F 7

```
8     CAD TEMP7
      LDB 1F 7
      CUB PNCH1
      CAD CNAME
      SUB ONE
      STA CNAME
```

```
9     CAD A0
      SUB 2F 7
      CNZ 1F 7
      CUB PRESC
3   2 comma
1     LDR OP
      CAD 3F 7
      ADD DRUML
      ADD LOOPL
      -STA OP
      CAD 4F 7
      STA TEMP1
      CAD 5F 7
      CUB OUTP7
3     44010  0
4     = Jend 0 =
5     CUB 1+
```

continued at bf of box (149)

(150)

FORM 14111

```
ALARM  HLT  505
       PTW  8510
            BA
       PTW  8010
       SLT  10
       PTW  8204
       STC  ALARM 7
       PTW  8419
       LDB  CURNT
       CAD  3+ 7
       ADD  ONE
       STA  1+ 7
       CAD  CURNT
       PTW  8410
       DBB  5- 7
       BUN  0+
```

```
MOD    CAD  1F 7
       STA  TEMP1
       CAD  2F 7
       CUB  OUTPT
       CUB  THIS
       CUB  10
2      (SET)
1
```

```
       SRT  8
1-0    CNZ  1F 7
       LDB  CNAME
OTPUT  DRB  1+
       BA   0F
       STA  TIME
       CAD  NSTAK
       LDB  3F 7
       CUB  GET
       CUB  INPUT

1      LDB  5F 7
       BUN  2F 7
2      LDB  6F 7
       BUN  2F 7
9
```

```
INPUT  LDB  4F 7
       CAD  PHOR1
       CUB  BCALL
2      HLT  7F
6
```

```
THIS   LDB  CNAME
55     IB   INPUT
4      BA   2-
       STA  CNAME
       CAD  ONE
       SUB  T
       STA  T
       STA  NSTAK
       STA  STOR
       LDB  5B 7
       CUB  REPL1
1      00110        0
```

```
INOUT  CAD  A0
       SUB
       NZ   2+
       CUB  PRESC
       CAD
       STA
       CAD
       CUB  OUTPT
       CAD  CNAME
       STA  TEMP1
       CAD
       SUB  ONE
       CUB  OUTPT
       CAD  TEMP1
       SUB  1F 7
       STA  TEMP1
```

```
INOUT  CAD  CNAME
       SUB  ONE
       STA  CNAME
IN     CAD  A0
A      SUB  1F 7
       CNZ  2F 7
       CAD  3F 7
       CUB  FIXT3
2      CAD  4F 7
       STA  TEMP1
       CAD  ST 7
       CUB  OUTPT
1      comma
1      CUB  PRESC
3
4      7000
4      CUB  1+
5      CAD  1F 7
       STA  TEMP1
       CAD  2F 7
       CUB  OUTPT
7      CAD  6F 7
       CUB  FIXT3
1  2   CUB 9B 6 CUB FORM T
```

```
LIBRY  CAD  NSTAK+1
       LDB  STOP 7
       CUB  GET
       CAD  BUN 0+

STOP   BA   2F
       STA  CNAME
       CAD  NSTAK+1
       SUB  ST 7
       CNZ  2+ 7
       CUB  1F
       ADD  5F 7
       LDB  2- 7
       CUB  GET
2      LDB  COP
       LDB  OPST+1
       LBA  FIXT4
       LDB  1- 7
       CUB  BCALL
5      nullstac
1      CAD  4F 7
       STA  TEMP1
       CAD  5F 7
       CUB  OUTPT
4      = HLT 0137 =
5      CUB  FIXT2
```

ONE = 4000

assignments of regions

OPST	30
NSTAK	30
Block (DUOST)	9
MATRX	15
NUMST	15
IFST	40
MASK	~40
OPTBL	~44
code TBL	11
HOLD	20
TRANS	35
PART1	150
PART2	150
PART3	250

initial buffer 60

OUT	20
CON	8

Results of 2nd count

53	
747	
684	
710	
457	
194	
175	
221	
189	
330	
3707	

LOOPS should contain things which are not likely to change during GOTPOS or MASTR

LIST OF ALL TEMP STORAGE (I THINK)

CURNT (5) for the MASK region

5-Q
5-WORD
5-Z
5-LSTCH
5-A0
5-A1
5-A2
5-A3
5-A4
5-EXITP
~~5-[illegible]~~
4-TEMP1
4-TEMP2
4-TEMP3
4-TEMP4
4-TEMP5
4-TEMP6
4-TEMP7
★PROCT
★PROCC
4-LLOC
★FREFL
4-DRUML
4-LOOPL
4-ABC
5-EXIT1
5-MEXIT R0
5-RIGHT R1
5-LEFT R2
5-OP R3
5-GEXIT R4
5-TOGXT R5
5-TODO R6
5-MFL R7
4-STOR
4-CNO
4-COP
4-CNAME
4-CMTX
4-CTF
4-C1
4-C2
5-PVAR R8
5-SPEC R9
5-COLUM R10
4-T
★HIGHL
★SEG1 # for ★ MASK
★SEG2 # "
★SEG3 # "
★TMIN
4-COUNT
★LOOPG
★EQUIV

★N
★BOOFL
★FOFFL
★INSW
★PHOR1
★PHOR2
★CBOOL
★IRTAG
★SWTAG
★RB
★DUOST

CARD

★ means put with the same.

COMPILER EXTENSIONS

```
FLFX  STA TEMP1
  2   BA  13
      STA EXIT
      CSA TEMP1
      FSU 4F 7
      ADD 4F 7
      SRT 8
      CNZ 1F 7
      SRT 2
      CAD TEMP1
      SLT 10
      -CUB 0
      ~~1 CAD TEMP1~~
  1   LDB 2B 7
      CUB ERROR
          58 10
          4
```

```
58 89   9
   10   0
59 10   0
58 10   0
   1
```

ERROR 13
$|X| \geq 90000000$

```
FXFL  CLR 60
      NOR 1F 7
      SRT 10 STC TEMP1
      SSC
      ADD FXFL 7
      SRT 2 CAD TEMP1
      SLT 10
  1   -CUB 0
```

```
SQRT  CNZ 1F 7
      -CUB 0
  1     STA TEMP1
        BA EXIT
        STC EXIT
        LDB 3F 7
        OSD TEMP1
        CCB ERROR
        CUB 14
  1     CAD 2F 7
        MUL TEMP1
        ADD 1F 7
        SLT 2
  3     UFA 4
        SLT 18
  1     STA 2F 7
        CAD TEMP1
        FDV 2F 7
        FAD 2F 7
        FMU 7F 7
        DBB 1B 7
        LDB EXIT
        -CUB 0
        50
  3     2550
  1     5050
  7
```

ERROR 4
$X < 0$

```
LN    STC TEMP1
LN1   BA 5
      STC EXIT
      CAD TEMP1
      LDB 3-7
      ADD 5F 7
      BOF 2+ 7
      CUB ERROR
      CCB ERROR
      EXT 2F 7
      FAD 3F 7
      STA TEMP2
      FSU 4F 7
      CUB 1F
      481   13
  2   48 31622777
  3
  4   48 6324 5553 99 98
  1   FDV TEMP2
      STA TEMP2
      FMU TEMP2
      STA TEMP3
      CAD 6F 7
      FMU TEMP3
      -FAD 7F 7
      DBB 2- 7
      FMU TEMP2
      STA TEMP3
      CAD TEMP1
      SRT 2
      CUB 1F
      7 constants
```

```
  1   EXT 5F 7
      FSU 6F 7
      FMU 7F 7
      FAD TEMP3
      LDB EXIT
      -CUB 0
      3 constants
```

ERROR 5
$X \leq 0$

```
PFLF2  FMU TEMP4   = 4000
       LDB TEMP5

EXP    STA TEMP1
       FMU 2F 7
       STA TEMP4
       FSU 1F 7
       SRT 10
1      5820 UA 0051
       SLT 18
       STA TEMP2
       SLT 12
       FAD 1B 7
       FSU TEMP4
       STA TEMP5
       FMU TEMP5
       CUB 1F
2      -5043429448
1      STA TEMP3
       FMU 8F 7
       FAD 2F 7
       FMU TEMP3
       FAD 3F 7
       FMU TEMP3
       FAD 4F 7
       FMU TEMP5
       STA TEMP5
       CAA TEMP4
       ADD 6F 7
       CCB 9F
       CUB 1F

8
2
3
4
6      4752000000
1      CAD 5F 7
       etc
       etc
       CUB 0
       BA 2332
9      LDB 1 7
       STA EXIT
       CUB ERROR

ERROR 2332  |x| ≥ 48
```

```
COS    FAD 4F 7
SIN    FMU 5F 7
       STA 5F 7
       FAD 6F 7
       FSU 5F 7
       STA TEMP1
       FMU TEMP1
       STA TEMP2
       CUB 1F
4      5115707963
5      -5015915494
6      580 ——— 0
1      BA 7
       ADD 3F 7
       STA 3F 7
       CAD 2F 7
       LDB 1B 7
       FMU TEMP2
       -FAD 1F 7
       DBB 2-7
       FMU TEMP1
       CUB 0
3
2
1
```

```
ATAN  (in other listing)
```

```
ERROR  CAD TEMP1
       HLT 7070
       PTW 8W0
       BA
       PTW 8W0
       LDB EXIT

60 / ERROR  -BA
61          HLT 7070
62          PTW B510
63          CAD TEMP1
64          PTW 8010
65          LDB EXIT
66          HLT 7270   PTW 8010
67          -CUB 0
68
```

31

71

B = 4000 = TEMP4

```
PFXFX   CNZ 21 7
        CUB 0      STA TEMP2
        CAD =1=
        STA TEMP3
        CUB 2F
3       OSP 100
        BOF 37 7
        CAD TEMP3
        CUB 0
        CAD =1=
        SRT 10
        DIV TEMP3
        CUB 0
        =1=
2       CAD =5(11)=
        MUL TEMP4
        STC TEMP4
        SLT 10
        NDR 1F 7
        CAD TEMP3
        MUL TEMP2
        SLT 10
        STA TEMP3
1       CAD TEMP4
        NOR 2F 7
        CAD TEMP2
        MUL TEMP2
        SLT 10
        STA TEMP2
        BUN 2B 7
2       CUB 3B
        =5(11)=
```

```
PFLFL   STA TEMP1
        BA  PFLF2
        STA TEMP5
        LDB 2- 7
        CUB LN1
```

```
PROCEDURE SIMPS (A, B, EPSILON, BOUND; VALUE; F()); BEGIN
COMMENT   A, B ARE THE LIMITS OF INTEGRATION. F() IS THE
          FUNCTION TO BE INTEGRATED. BOUND IS GREATER THAN
          THE MAXIMUM VALUE OF F() ON A, B.
          EPSILON IS THE PERMISSIBLE DIFFERENCE BETWEEN TWO
          SUCCESSIVE SUMS;
IBAR = BOUND. TH A);
N = 1;
TH = (B-A);
J = 0.5 (F(A) + F(B)) TH;
LOOP.. S = 0
H = 0.5 TH
FOR K=(A+H, TH, B); S = S + F(K);
I = J + 4H·S;
IF PCS(0); WRITE(;; INTGR, FMT)
IF ABS(I-IBAR) LEQ EPSILON; BEGIN VALUE = I/3; RETURN END
IBAR = I;
J = 0.25(I+J);
N = N+N;
TH = H;
GO TO LOOP;
OUTPUT INTGR(I/3);
FORMAT FMT(F20.8,W)END;
```

```
WRITE   HLT
        BUN 1F 7
        STA EXIT
        CAD 2-
        STA 4000
        CAD WRITE
        STA 4001
        CAD 4F 7
        STA 70
        CAD 3F 7
        STC WRITE+1
        BUN 2F 7
1       STA EXIT
        CAD WRITE
        STC 4000
2       LDB 5F 7
        CUB 2F
3       BUN 1B 7
4       CUB WIN
5       =15=
-----------------
2       -STA BUFFR 7
        DBB 1-7
        BF7 2-
        CUB 2F
BUFFR   HLT

        16 of these
-----------------
2       CSU 4000
3       SUB 1F 7
        BOF 2F 7
        LDB 4000
        -CSU 0
        STA 4000
        BUN 3B 7
2       SUB 4F 7
        STC FORMAT
        CAD 2F 7
        STC ZIN
        STA ZOUT
        CUB NXCH4
1       99950 0
2       008 00 002
4       339999
ALPHA   CAD 6F 7
        STA 4F 7
NXCH1   CAD 2F 7
        STC N1
NXCH3   STC N2
NXTCH   STA 4002
NXCH2   CAD ZIN 7
        EXT 1F 7
        STA ZIN
```

```
        SFT 8
        STA 4003
        LDB 4003
FORMAT  CAD 0
        LDB ZIN 7
        -SLT 98
4       CUB 5F
6       CUB ALFA
2       =1=
1       1130 03
-----------------
5       ADD =20
        BOF 4F 7
        ADD =40
        BOF 5F 7
        ADD =32
        CCB GENER
(20)    ADD =02
        CCB ALPHA
(40)    CAD 4002
        CUB NXCH3
4 (32)  SRT 9
        CAD 4002
(02)    SLT 1
        CUB NXTCH
5 (32)  MUL =30=
        STA OP
(02)    CAD 4002
        CUB NXCH1
        =30=
-----------------
GENER   CAD 4002
        STA N
        LDB OP 7
        -BUN 11 7
B       BUN 1F 7
F       SRT 0
I       LDB 4001
        -CUB 0
OP      HLT
P       CUB P
T       CUB T
W       CUB W
X       BUN 6-7
1       ADD ZOUT
        STC ZOUT
        CUB NXCH1
WIN     STA 4000
        BA
        STA 4001
        LDB OP
        -BUN 04 7
CAD=1=
STA 4002
```

```
(F)     CUB F
(J)     CAD 4000
        NOR 3F 7
        STC 4000
        SSC
        ADD =10=
        CUB 4F
(X      CUB X
3       STC 5- 7
        CUB 4F
        =10=   =1=
-----------------
4       STC α
        STA γ
        STA β
        CAD N 7
        CUB LOOP
X       CAD 4000
        SFT 8
        ADD =-49=
        OSD =-49=
        BOF 2+ 7
        STC 5- 7   ADD =1=
        STA β
        ADD N 7
        STC α      CLR
        STA
        CUB IF
        N
        =-49=
1       CAD 4000
        SLT 2
5100    STC 4000
        CAD N2 7
        CUB LOOP
N1      CAD =1=
F       STA β
        ADD N
        STA α
        CAD 4000
        CNZ 2+ 7
        CAD =51
        SFT 8
        STA
        CAD N2 7
        SUB =4=
        STA N2 7
1
```

```
LOOP    SUB α
        ADD ZOUT
        STA ZOUT
        CAD 4000
        OSD neg
        CCB 3F
        CAD =20
        LDB 2- 7
        CUB OUTPT
-----------------
1       CAD α
        CNZ 1F 7
        CAD β
        CNZ 2F 7
        CAD N1
CUB'    SUB =1=
        CNZ 2+ 7
        CUB NXCH1
        STA N1
        LDB 4001
        CUB 0 7
1       SUB =1=
        STA α
CAD ZOUT CAD β
ADD =1=  SUB =1=
STA ZOUT STA β
LDB neph CNZ 3F 7
         CAD =03
         CUB OUTPT
3        CAD 4000
         SLT 1
         STA 4000
         SLT 18
         EXT =81
         CUB OUTPT
on next page
```

```
α 4003
β 4004
γ 4005
```

```
OUTPT   SRT  8
        STA  OUTPT 7
        CSA  ZOUT  7
        SUA  ZOUT  7
        ADD  1F    7
        STA  1F    7
        SRT  1
        ADD  2F    7
        STA  2F    7
        SUB  3F    7
        STA  3F    7
        CAA  OUTPT 7
        SRT  8
        ADD  OUTPT 7
1       IS   0000
2       72   BUFFR+5016
        BOF  1+7
3       72   0000
        -CUB 0
ZOUT    0
```

```
8       CAD  α
        CNZ  1F    7
        CAD  2     7
        CNZ  2F    7
        CAD  N1    7
        SUB  =1=
        CNZ  2+    7
        CUB  NXTCH
        STA  N1
        LDB  4001
        -CUB 0
        CUB  3F
2       SUB  =1=
1       STA  α
        CAD  ZOUT
        ADD  =1=
        CUB  4F

N1
        =1=
4       STA  ZOUT
        CAD  β
        SUB  =1=
        STA  β
        LDB  =8B=
        CNZ  3+    7
        CAD  =03
        CUB  OUTPT
        CAD  4000      CLR 8B
        SLT  1
        STA  4000
03      SLT  18
        EXT  =81
        CUB  OUTPT
```

```
3       CIRA 7         SUB=510—0=
        STC  4000
        STA  7         STA β
        CAD  ZOUT
        ADD  =1=
        STA  ZOUT
        CAD  =23
        LDB  =4F=
        CUB  OUTPT
        CAD  =2=
4       STA  α
        ADD  =1=
        CUB  LOOP
        =4F=
        =2=
        =510—0=
```

```
ALFA    STA  4000
        SRT  8             4F
        SUB  =14=
        CNZ  2F    7
        CAD  ZIN   7       7F
        EXT  =     =
        STA  ZIN
        CUB  NXCH2
2       CAD  ZOUT          1F
        ADD  =1=   7
        STA  ZOUT
        LDB  =ALPHA=       3F
        CAD  4000
        CUB  OUTPT
        =1=
        =14=
        =     =
        =ALPHA=
```

```
T       CNZ  1F    7
        CAD  ZOUT
        ADD  ZOUT
        SUB  =2=
        SRT  1
        STA  T     7
        LDB  3+    7
        CAD  =1=
        SUB  1+    7
        STA
        CAD  BUFFR+16      CUR BUFFR
        PTUB 8410  7       EXIT
        DBB  RUN   EXIT    =1=
                           =2=
1       PTWF 500           (RUN T→)
        SUB  =1=
```

```
W       CIRA 8     7
        ADD  2+    7
        STA  1+    7
3       CWR  BUFFR 7
        BUN  EXIT
```

α 4003
β 4004
γ 4005

HANO'STOCK FORM 14111

object program load routine available locations 4980 - 5019 partly 6000 - 6005 and 60( ) - 6019.

```
1   (nnn0 CAD 3nnn-1)
        STA  TØ
        [illegible]
        STA  T1
3       LDB  TØ
        DBB  2F
5       CAD  1B
        ADD  =0010—01=
        BOF  3F
        STA  1B
        BUN  1B

2       -CAD  1
        STA   TØ
        EXT   =1111 0000=
        ADD   T1
        -STA  1
        BUN   3B
```

```
5000:   CAD   5004
1       ADD   5018
2       BOF   7011
3       STA   5004
4   nnn0 CAD  3nnn (= FREL-1)
5       STA   6000
6       SLT   2
7       SRT   6
8       STA   6001
9       LDB   6000
10      DBB   5012
11      BUN   5000
12      -CAD  1
13      STA   6000
14      EXT   5019
15      ADD   6001
16      -STA  1
17      BUN   5009
18      0010000001
19      1111110000
```

CARDS

```
7010   BUN  5000
7011   CAD  7013
7012   STA  7010
7013   BUN  7000
```

PTAPE

```
6000   CSA  6008
6001   LDB  6004
6002   -STA 4000
6003   DBR  6002
6004   BF4  20
6005   BF4  0
6006   BF4  40
6007   PTR  6000
6008   0
```

F.B. 1:8 0" 3 — 3 $(01^{10})^{15}$ $01^{8}00$
2,3:4 333 $(31^{10})^{16}$ 0 ——— 0

```
BEGIN 1000 CRD 5000
      CAD FREFL
      CLR
      SLT 7
      ADD FREFL
      ADD 5004
      STA 5004
      LDB TEMP5
      CRD 5000
      CAD SUP7
      SUA 6F 7
      STA TEMP7
      NOR 2F 7
      LDB TEMP7
    - CAD SEG4
      OSD 6F 7
      BOF 9B 7
      STA TEMP1
      SUB 7F 7
    - STC SEG4
      CUB 1F
      -6=
      -600=
      LDB TEMP6
      CAD 5006
      -SRT 0
      EXT 1F 7
      STA TEMP7
      NOR 2F 7
      ADD 5B 7
      RCB 9B
    - CAD 4999
      EXT 3F 7
      ADD TEMP7
      CUB 4F
    - CAD 4999
      CUB 5F
      1000
      SRT 10
    - CAD 4999
      SLT 10
      STA TEMP2
      DBB 1+7
      BA
      STA TEMP6
      CUB (PUNCH+1)
```

SUB ONE

```
      CAD 1F 7
      STA PEXIT
      CUB 8F
```

1 CUB 5F

```
      CAD 5000
      STA TEMP1
```

```
      OSD 5006
      CUB 4F
```

```
    - CAD SEG4
      STA TEMP5
      LDB 6F 7
```

format bands followed by CRD 4000

```
      CRD
      CRD
      CRD
      CRD
      BF 5
      CRD 6000
    6 CUN 6000
```

6005 6004 6003 6002 6001 6000

333333330000000

12
11
10
9
8
7
6
5
4
3
2
1

MEASY

```
          START 1000
          loading routine here
BEGIN 0010 CRF FI
      0030 CWF F01
      ~~0230 CWF F02~~
      0340 MOW 1000
           BOF 1-7
           BTS PROC
           BUN PROC 5
```

first card of deck
began searching the tape

```
PROC  0010 CRD 6005
LOOK       BFG 1600
           CAD 6012
           CNZ 1F 5
ADDR       CAD 6011
           CNZ 2F 5
           LDB 6009
           DBB 3F 5
FLAG       OSD 6012   CUB STRT
                      CUB LOCAT
1          ADD 4F 5
           LDB 5F 5
           CUB TABLE
           CUB SYMBL
2          -CUB FLAG+1
3          =7=
4          CUB 4F
5          START FLAG+10
4          STA 1614
           SRT 2
           CNZ 1614
           BUN ADDR 5
3          CAD 6012
           HLT 8421
           STA 6012
           BUN LOOK 5
           START FLAG+20
           CSU 6010
2          ADD 6021
EQ         ADD 6013
           STA 1613
           BUN FLAG 5
```

```
LOCAT CAD 6017
      CNZ 1F 7
      CAD 6018
      CNZ 2F 7
FINIS CAD 1613
      SRT 4
      CAD 1608
      CUB FIN1
1     LDB 3F 7
      CUB TABLE
2     LDR 6018
      -CAD 3950
      STA 6022
```

```
      CAD 6021
      -STC 3900
      -STA 3950
      BUN FINIS 7
3     CUB A4
```

```
      HLT 7557
CHAIN CAD 6019
      ADD 1F 7
      STA 6019
      CAD 6015
      SRT 3
      CAD 6008
      SLT 3
      STC 1615
      CSU 6021
      -STC 2900
      STA 1608
      CUB LOCAT 1   040 0
START CSU 6012
      STA 6012
      BUN ADDR 5
      START FLAG+42
      LDB 6010
      -CAD 3900
      CUB EQ
      START FLAG+46
      CAD 6010
      ADD 1F 7
      STA 9+7
      LDB 8+7
      -CAD 2900
      CUB CHAIN
1     1050

SYMBL LDB 1F 7
TABLE STC 6023
      BA
      ADD 3F 7
      STA 9F
      CAD 2F 7
      MUL 6023
      STC SYMBL 7
      LDB SYMBL 7
      CUB SERCH
1     CUB EQ
3     CUB 4
2     1001001001
SERCH -CAD 1900
      CNZ 2F 7
      CAD 6023
      -STC 1900
      HLT
9 3   -999
2     SUB 6023
      CNZ 2F 7
      -CAD 2900
      OSD 7- 7
      BOF 9B 7
      LDB 9B 7
      -19 9996
```

```
2     DSB SERCH 7
      LDB 3B 7
      BUN SERCH 7

A4    STA 6021
      CAD 3F 7
      STA 6019
      CUB FINIS
A3    STA 6022
      CAD 6021
      -STA 2900
      CUB FINIS
3     00 48 540303
FIN1  CNZ 1F 7
2     CAD 1614
      SRT 2
      CAD 1615
      SRT 4
      CAD 6016
      CIRA 9
      SLT 10
      STA 6020
      CUB 2F
1     SLT 1
      CAD 6008
      SRT 1
      BUN 2B 7
2     LDB 8F 7
      -STA 11
      CAD 6022
      SRT 4
      CAA 6021
      SLT 4
8 9000 STA 4000
  0050 CWR 6005
      CAD 6021
      ADD 1F 7
      STR 6021
      STA 6022
      STA 6019
      CAD 8B 7
      ADD 2F 7
      CCB 4F
      CUB 5F
1     =1=
2     90 02
5     140 MOW 8000
      CAD 1F 7
4     STA 8B
      BUN PROC 5
STRT  STC 7+7
      STA 6021
      0030 CWR 6005
      BUN PROC 5
1 9000 STA 4000
```

```
BEG 1030 CUR 6005
     LDB 8F 7
     CAD 1F 7
     -STA 6021
140  MOW 8000
     HLT 1111
040  MTS 0000
340  MRD 1000
     BOF 1F 7
     CUB START
1 9
```

165 lines

FI same as before

FOI XXX 31 6 2 4 3 5 1 2 4 00200222R 00200222 31 10 8 2 31 2 0031 10 8 2 31 2 7 4 3 2 4

6022 6021 6020 6019 6018 6017 6016 6015

3 9 2 3 7 2 4 0031 10 0031 10 8 2 312 9 31 2 8 2 312 0031 10 31 11 31 11

6014 6013 6012 6011 6010 6009 6008 6007 [6006] [6005]

16
18
18
14

MEASY Modified EASY Assembly System
or, "More Elegant Assembly System Yet"

MEASY is an improved version of EASY, for use with 205's which have magnetic tape and on-line 407. The input language is the same as regular EASY, with the exception that if instructions have a location greater than 3999, they will not be loaded into that location.

Operating Instructions: Mount any calibrated tape on TSU 4, enable writing. Then load the MEASY deck in the Card Reader, followed by your input cards. Ready the 407 with a skip-to-1 carriage control tape. [Standard 120-130 board with new print contour except in 4,9,34,42,43,66,69] MEASY is a 1½ pass assembler; during the first pass a 407 listing will be produced. Forward references will not be shown on this listing; a "lozenge" will appear next to an address which is to be fixed up later. If the same symbol occurs twice in the symbolic program before the location field, the flag "HM.." will appear on the listing.

The result of the first pass is a self-loading tape — it can be loaded by

```
0040 MTS 0000
0340 MRD 1000
BOF  1 -
CUB  (T6)
```

The loading routine on this tape is the other ½ pass of the assembly program — it fixes up forward references which were made. [>3999 ok]

There are two programmed stops in MEASY.

08 8421 mispunched symbolic OP-code is displayed in the A register. Change the A register to the correct OP-code and depress program start; the new code will be substituted for the old.

08 IIII END card missed. TSU 4 contains your program tape. Depressing program start will cause your assembled program to be loaded. [Control (1) on TCU should be set to 5 (press B).]

If the number of symbols in your program plus the number of symbolic OP-codes in MEASY's skeleton exceeds 1000 the assembler will get into a closed loop — there is no stop for this.

LOADING ROUTINE

```
SLT
MREAD 0140 MRD 8
      LDB 1F 7
GET   -CAD 4001
      STA 5000   SF 7
      ADD 3F 7   STA 0002
      BOF HLT
1     BA
      ADD 3F 7
      STA 5001
IRY   LDB 5000
      DBB FIX 6
      LDB 5002
      -CAD 3980
      LDB 5001
      -STA 0
      BA 9980
      ADD 1- 7
      BOF 1F READ
      BUN SB 1
```

```
6     MREAD 6140 MRD 8000
1           CAD =9000 644000
2           STA GET 7
3     GET   CAA 4000
4           STA 5000
5           SRT 4
6 2         STA 5001
7           ADD =1111100000 7
8           BOF HLT
9           BUN IRY
10    NXT   CAB 5001
11          ADD =9999999996 6000=
12          BOF 3F 7
13          LDB GET
14          -CAD 1
15          LDB 5001
16          -STA 0
17 3        CAD GET 7
18          ADD =90      02=
19          BOF 1B 7
```

```
FIX   -CAD 1
      STA 5000
      EXT =1111110-0=
      ADA 5001
      SRT 9
      CNZ 1F 6
3     SFT 1
      CAD 5000
      SLT 10
      -STC 1
TRY   LDB 5000
      DBB FIX 6
      BUN NXT 7
1     STC 5002
      SLT 6
      EXT =1111110-0=
      ADD 5002
      SRT 7
      BUN 3B 6
5
6
```

P.S. There has been some question as to the meaning of the word "elegant" as used in EASY. This is to be taken in its mathematical connotation of "short and sweet". Actually the OP-table entries take the number of non-zero locations in MEASY program as assembly begins is less than 300 (including format bands and the loading routine)!

```
START 1000
      BOF 3F 7
      BTB FIX
      CUB MREAD
MREAD NO MRD  8000
         CAD  5F 5
1        STA  GET 7
GET      CAA  4000
         STA  T0 5
         SRT  4
2        STA  T1 5
         ADD  T0 5
         BOF  5699
         BUN  TRY 6
NXT      CAD  T1 5
         ADD  7F 5
         BOF  3F 7
         LDB  GET 7
         -CAD 1
         LDB  T1 5
         -STA 0
3        CAD  GET 7
         ADD  8F 5
         BOF  1B 7
FIX      -CAD 1
         STA  T0 5
         EXT  6F 5
         ADA  T1 5
         SRT  9
         CNZ  1F 6
```

cont. on next page

Title Card
Tape Search for 00 – 0000 TSU 4
Dump Zero

Drum Loading Routine:

```
6  440—0   4004 4003 4002 4001 4000   44 4080   6810
6  440—0   4009 4008 4007 4006 4005   44 4085   7810
6  440—0   4014 4013 4012 4011 4010   44 4090   8810
6  440—0   4019 4018 4017 4016 4015   44 4095   9810
6  440—0   6205 [440—0] BF4 [CAD 5000]
```

etc.

load this routine into loc 5

[equiv. | alphanumeric | similar] 44 4000

END title card

```
CRD 4000
LDB 4000
CAD 4001   ADD =7=
STA 1900
CAD 4002
STA 2900
BUN 8003
=7=
```

HLT ~~BUN 3B~~ 6 HLT 7555

```
5  9000 CAA 4000
6  1111 11 0000
7  999999 6000
8  9000 00 0002
9  1111 11 1110
T0   BTS HLT
T1   BT6 FIX
T2   CUB MREAD
```

```
8   ~~BT4 GET~~
    LDB =4=
1   CAD 1000
    STA 4001
    DBB 1B 7
    CAD 1B 7
    SUB =5=
    STA 1B   LDB 1B 7
    CAD 4000   CWR 4000
    ADD =10—05=
    ECB 1F
    STA 4000
    BUN 1B 7
```

```
    ~~BT4 SEC~~
    BA
    ADD = BF4 – 15 =
    STA 4011
010 CWR 4010
    CUB 8B
```

```
START 0
6810444000
0
0
0
0
0
6618440
4   =4=
5   =5=
INC =10—05=
0810445000
0
13 0000
13 0000
13 0000
13 0000
6600205
FURGE 40 0015
BEGIN CWF F0
      CUB 8F
8     BT4 0
      LDB 4B 4
1     –CAD 1000
      –STA 4001
      DBB 1B 7
```

```
      CAD 1B 7
      SUB 5B 4
      STA 1B 7
10 CWR 4000
      CAD 4000
      ADD INC 4
      BOF 2F 7
      STA 4000
      BUN 1B 7
2     CAA 1B 7
      SUB PURGE 4
      STA 4011
10 CWR 4000
      BUN 8B 7

F0 3200002002200
   222000000
   2"2"2"2"2"
```

HANO STOCK FORM 14111 H

12
11
10
9
8
7
6
5
4
3
2
1

Frankly, it's a miracle if it does.

How does the compiler work? The comments given here however serve as a key to understanding the internal mechanisms to translate the Algol language. They are not merely a restatement of what the program does: the listing is a precise step-by-step account of exactly what happens, but these notes are abstracted one level from the program itself and then attempt to tell what is happening from an overall standpoint. The reader who encounters these notes for the first time is advised to write a simple flowgram and try to follow it through the flowcharts. Then when he gets a feel for what is happening he should take closer looks at the steps.

The order of presentation here is intended to be such that the first routines illustrated use no other routines internally, the routines use these first ones, and so on, each routine using only those which have been discussed earlier. This order was impossible to follow in the case of the "GET" and "MASTF" subroutines, which use each other, but elsewhere it's all right.

Some of the author's terminology may need explanation: a "stack" is a word used for a table which initially contains nothing but things get piled into it, one on top of another; and these things are generally only taken away when they are removed from the top. Several stacks are distinguished in this compile: the operator stack, the operand stack, the subscript stacks, the constant stack, the true-false stack, and the "do-stack". The operation of these stacks will become clear by working through some examples. Phrases such as "put onto the top of the operator stack" and "removed from the top of the subscript stack" will be used in this description and they should be clear by the analogy above.

The style of presentation is adapted from the practice of compiler programmers. The routines are identified by code numbers which reference the text, so the diagrams indicate only the topology of the situation. The explanatory text which accompanies the diagrams tells what goes on inside each box.

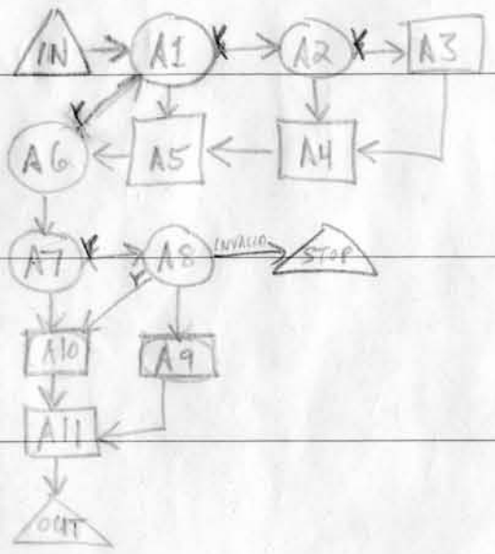
A "counter" is a storage location which keeps count of something or other. Counters are distinguished from other temporary storage in that they can be used to store only one type of information throughout the program, while temp. storage is a relatively short term storage and can be used for many things.

More general information about compiling may be found in section I, where the program itself begins.

The accompanying "coding details" are to be bypassed on first reading — they apply directly to the symbolic machine language and are included here for reference in partial explanation of the coding, for someone who'd like to make changes.

A. "NXTCH": Next-Character Subroutine. This serves as the sole communication link between the compiler and the input-handling routines. Two routines are necessary, one for Cardatron input and one for paper-tape input. It also is tied in with the ALARM routine, as it prepares the input for possible Flexowriter typeouts.

Remarks: Temp. Storage WORD is initially empty as is the "input buffer" and ALARM buffer. Input to this routine is the counter Q which is either 0, 1, or 2. The subroutine yields as output the next character, from left to right, of the input's program.



WORD EMPTY? A1. If WORD is empty, go to A2; else to A5.

BUFFER EMPTY? A2. If the input buffer is empty go to A3; else to A4.

FILL BUFFER A3. Fill input buffer with next card or next paper tape record. Stop if a paper tape record too long is sensed. To A4.

FILL WORD A4. Fill WORD with next five alphanumeric characters. To A5

REMOVE NEXT A5. Take next character from WORD. To A6

TWO BLANKS? A6. If it and the last character were both blanks, return to A1; else to A7.

SPEC. CHAR.? A7. If it is alphabetic or numeric go to A10, else to A8.

ON FLEX.? A8. Stop if it is not a valid special character. Otherwise, go to A9 if the Flexowriter cannot type this character, to A10 if it can.

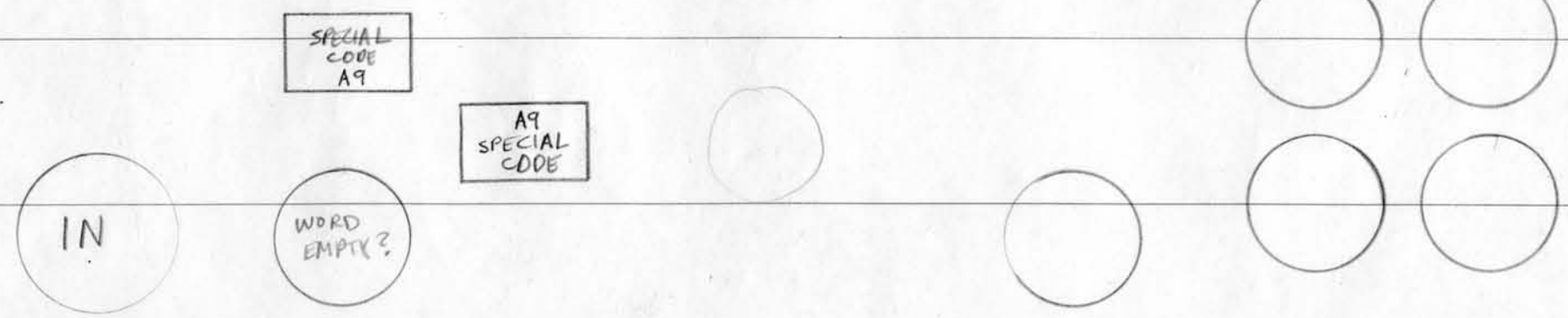
SPECIAL CODE A9. Push ~~it~~ space-letter-spacewise right side of ALARM buffer. To A11

FLEX. CODE A10. Push it into right side of ALARM buffer. To A11

SET A(2+Q) A11. Put it into temp storage location A(2+Q). Exit.

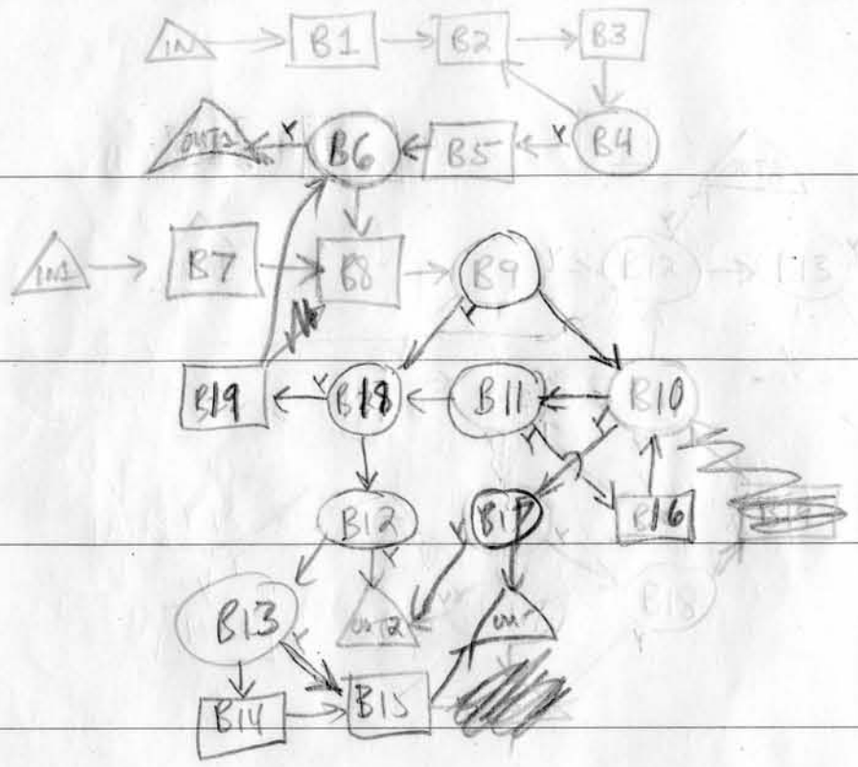
Coding Details. Entry NXTCH: rA = shift, rB = stored value of Q.
Entry NXCH1: rA = exit. Q will be set to 1.
Exit: rA, LSTCH, and A(2+Q) contain next character. rB=Q. Special characters have been converted to a code number between 1 and 10.

Temp. Storage Used: high-speed-loop 6, WORD, Q, A4, A(2+Q), Z, CURNT table, LSTCH, NEXIT.



HAND STOCK FORM 14111

8. "BUILD" This subroutine obtains an identifier from the NXTCH routine and converts it into a number between 1 and 225. The latter number is the code number which the compiler works on. Remarks: Input to this routine is the first (alphabetic) character of the identifier in temp storage location A3. Output is the value of I.



INITIALIZE B1. Sets Name buffer to blanks and sets the special flag off.

RECORD A3 B2. Inserts A3 into name buffer.

A3 NXTCH B3. NXTCH with Q=1. [Subroutine A].

END OF IDENT.? B4. If A3 is alphabetic or numeric go to B2.

PREFIX ENTRY B7 Special entry, sets up for special lookup on only first character of name buffer with its procedure tags, as a prefix.

RECORD LENGTH B5. Inserts length of identifier string into the name buffer.

WORD "TO"? B6. Special exit to I8 if the identifier is the word "TO."

SCRAMBLE B8. "Scramble" name buffer into a number I between 1 and 100.

NEW ENTRY? B9. If table entry for I is unused, go to B21; else to B10.

NAME MATCH? B10. If identifier for table entry I matches the name buffer, go to B17.

CHAIN REF.? B11. If table entry I indicates a "chain reference" to another number I, go to B16. Else to B21.

SPECIAL EXIT? B12. Special exit to I18 if called for;

NEW ENTRY? B13. If table entry for I is unused, go to B15;

SET UP CHAIN B14. Find highest unused place in table. ALARM if none are left. Put this as a chain reference into table location I and set I to this number.

STORE NAME B15. Put Name buffer into table associated with I. ALARM if table is packed. Exit I.

PROCEDURE? B18 if flag is off and if we are in the middle of a procedure declaration.

SPECIAL EXIT? B17. Special exit to I18 if called for; else normal exit.

GLOBAL WORD? If the identifier is a reserved word like "CAS" or "AND", or a previously defined procedure name, exit;

HOLD NAME. Insert procedure tag into name buffer, and set flag ON. To B11

TRY AGAIN. B16. Set I to the chain reference. To B10

PROCEDURE? B18. If flag is ON, to B12.

HOLD NAME B19. Set flag on. Insert global word tag into name buffer. Go to B8.

Coding Details: Entry BUILD: A=exit A3 = first char of ident

Entry BLD1: EXITφ = 0 if special exit desired at steps B12 or B17; otherwise EXITφ = exit. R contains ident; a prefix lookup is done.

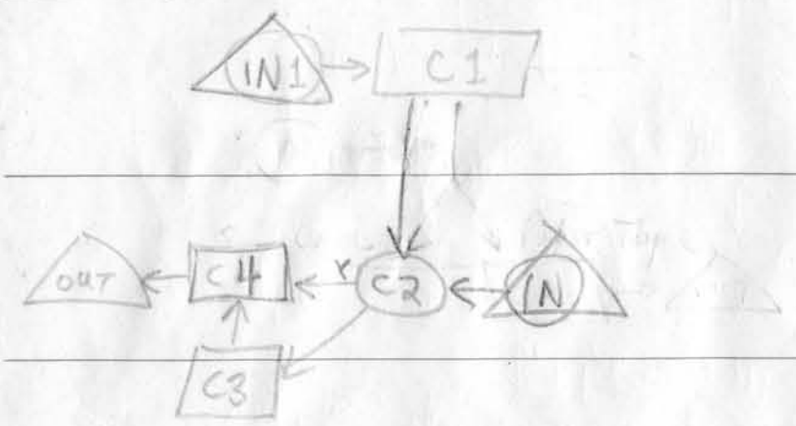
Entry BLD2: same as BLD1, with A = 0.

Normal Exit: TEMP3 contains the equivalent in the 4 low order positions

Temp Storage Used: TEMP1, TEMP2, TEMP3, TEMP4, MEXIT, RIGHT, LEFT, OP, GEXIT, TOGET, TOOO, MFL, PVAR, SPEC, COLUM, A3, EXITφ, C1, C2, PART1 table, PART2 table, PART3 table.

Subroutines used: NXTCH

C. PUNCH subroutine. This is the only link between the compiler and the punching of the output, except possibly for the initialization and FINISH where special leadings are punched. Two routines are necessary, one for Cardatran, one for paper tape.
Remarks: Input to this routine are a compiler word and the location into which it is to be placed; or, a forward reference from a previous location is defined using an alternate entrance.



MAKE FWD. REF. C1. Set location to the next available place for a forward reference table entry, set Number to the forward reference code used by the loading routine.

EASY CASE? C2. [Cardatran] If location is one higher than the previous location, or if the Card buffer area is still full, go to C4.

OUTPUT C4. [Cardatran] Put Number into Card buffer area. If Skip Switch set on, print Location = Number on 407. Exit.

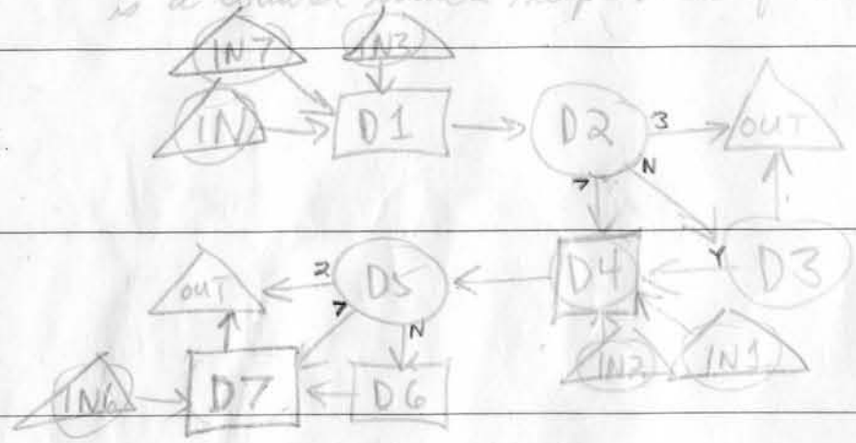
PUNCH C3. [Cardatran] Punch contents of Card buffer area, set location to beginning location of new card buffer load. To C1.

C2. [Paper tape] If location is one higher than the previous location, go to C4.
C3 [Paper tape] Punch "PTR location" with sign of 4
C4 [Paper tape] Punch Number. Exit.

Coding details: Entry PUNCH: rA = [illegible] TEMP1 = location
TEMP2 = number. Other entries are for forward reference table linkage.
Entry PNCH1: rB = exit, high order two of rK = forward reference slot, DRUML+LOOPL = current location.
Entry PNCH4: same as PNCH1 except forward ref. slot is low order two of rA.
Entry PNCH3: rB = exit, for ref table entry in rA.
Entry PNCH2: rB = exit, for ref table entry in TEMP2.

Temp storage used: TEMP1, TEMP2, FREFL, LLOC, high speed loop 6, PCH table, PEXIT.

D. OUTPT subroutine. This subroutine is the normal outlet for compiled instructions; it handles the automatic use of high speed loop 7 in the object program. Remarks: DRUML is the counter which tells where the current loopfull will go on the drum. LOOPL is the position in the buffer where the next compiled instruction will go. ABC is a counter which keeps track of the number of constants which are to go into this loopfull.



STORE INST. D1. Put coded instruction into loop 7 buffer

SPECIAL ENTRY ? D2. If special entrance 3, exit. If special entrance 7, go to D4

LOOPT FULL ? D3. If there is room for more than one instruction in loop 7 buffer, exit.

NEXT DRUML D4. Calculate address of next loopfull. If it is between 981-999, 1981-1999, 2981-2999, round up to next 1000.

SPECIAL ENTRY? D5. If special entrance 2, exit. If special entrance 7, go to D7.

CUB NEXT D6. Put CUB (next loopfull) into loop 7 buffer

EDIT AND UNLOAD D7. Edit instructions in loop 7 buffer from compiler code to 205 code and punch. Punch all constants from constant buffer, if any. Set DRUML to next, and set LOOPL and ABC to zero. Exit.

Coding Details: Entry OUTPT: r4=exit, TEMP1 = coded instruction

Entry OUTP3: same as OUTPT. Omits the test for full loop.

Entry OUTP2: r4 = exit. Acts as if loop was already full.

Entry OUTP5: same as OUTPT except r4 = exit.

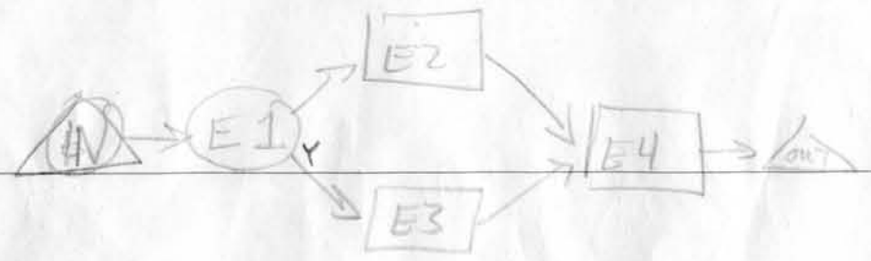
Entry OUTP6: EXITP = exit, TEMP3 = address of next loopfull. Punches the loop only.

Entry OUTP7: r4 = exit. TEMP1 = coded instruction. Combined features of OUTP3, OUTP2, OUTP6. Exit: r4 = 0.

Temp. Storage Used: TEMP1, TEMP2, TEMP3, TEMP4, DRUML, LOOPL, ABC, OUT table, CON table, LLOC, PART2 table, EXITP.

Subroutine Used: PUNCH

E. ACALL, BCALL subroutine. These produce the object program instructions to call a subroutine with exit in the A or B register, respectively. Remarks: input is the address of the subroutine; a special flag is set by the "FOR" processor to cause the output program to jump back to the incrementation phase otherwise control continues in sequence.



LOOP 7 FULL? E1. If there is room for only two more things in Loop 7 buffer go to E3. [Subroutine D]

USE CONSTANT E2. Compile CAD or LDB with next constant. Go to E4.

USE DRUM E3. Compile CAD or LDB with next drum location [Subroutine D entry IN3].

COMPILE INSTRS. E4. Compile CAB(subroutine) [Subroutine D entry IN3]. Get address of next loopfull [Subroutine D entry IN2]. Prepare constant CAB next on, if special FOR flag is set CAB (PHOR1). Add it to the constant buffer. Punch out the loop 7 buffer [Subroutine D entry IN6]. Exit.

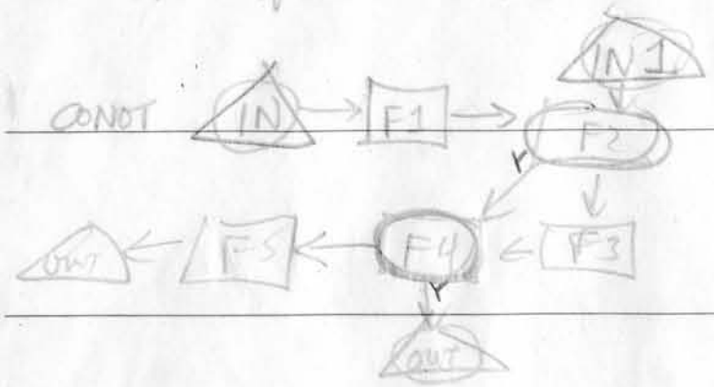
Coding Details: Entry: rA = subroutine address, rB = exit. If rB is zero, exit will be made to FORIN.

Temp Storage Used: TEMP2, TEMP3, TEMP4, TEMP5, LOOPL, ABC, DRUML, RB, CONTable, PHOR1, EXITφ

Subroutines used: OUTPT.

Those two short routines are used fairly often.

F. CONOT and FULL subroutines. CONOT prepares a constant for compilation. FULL makes sure ~~there~~ there is room for M more instructions in the Loop buffer.



M=3 F1. Set M=3.

ROOM FOR M? F2. If there is room for M more instructions in Loop buffer go to F4.

UNLOAD F3. Punch contents of Loop buffer [Subroutine D, entry IN1]

ENTRY IN1? F4. If entry IIN1, exit

STORE CONSTANT F5. Increase ABC by 1, put constant into constant buffer. Exit

Coding details: Entry CONOT: rA- and rB- ~~and~~ ~~rB- ind~~ Temp Storage used: TEMP1, CNST1, ABC Subroutines: FULL

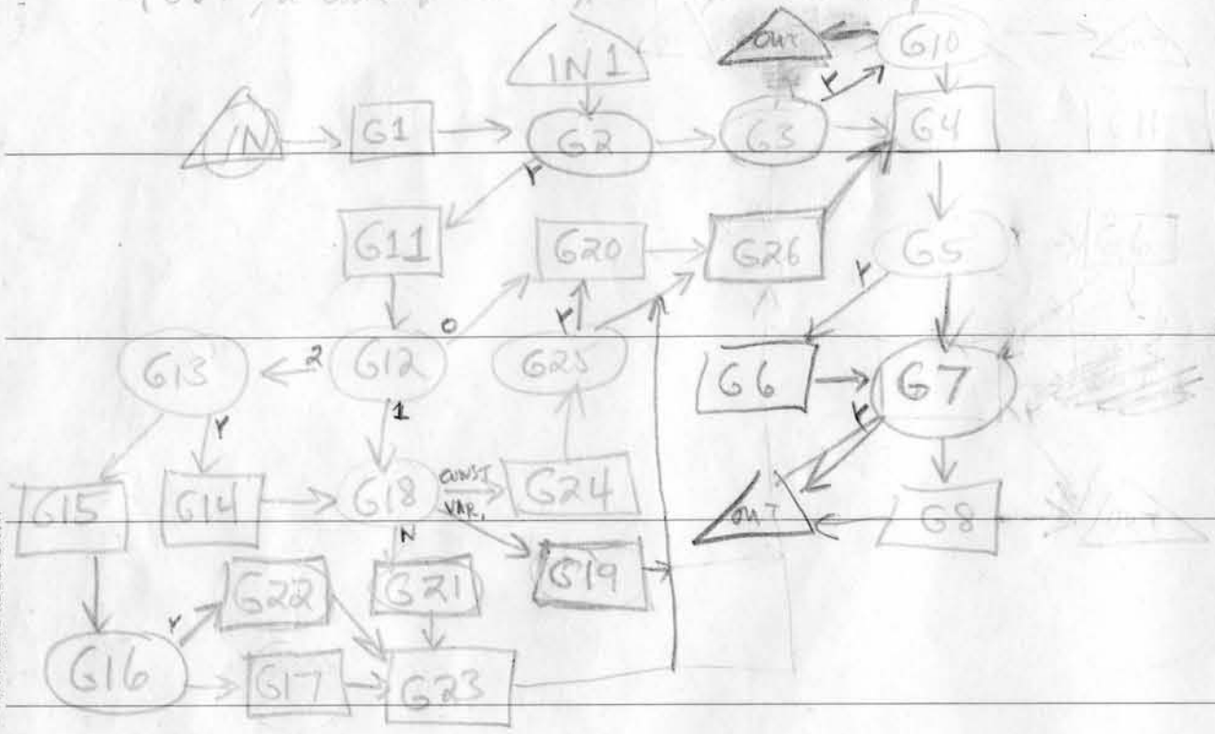
Entry CNOUT: TEMP5 and rB- ind

Entry FULL: ~~LOOP~~ M-2. rB- ind. Temp Storage used: ACC, LOOP1 Subroutines: OUTP1

Entry FULL1 M-3.

The routine also does some of the bookkeeping for allocation of temporary storage.

6. GET routine. This very general routine compiles the instructions to bring a coded quantity into one of the registers. It is practically the only place arithmetic instructions are compiled. Because this routine uses itself as a subroutine, and it also uses MASTR which uses GET as a subroutine, it must store away its exits and inputs to prevent them from being overlaid. Inputs to GET are TOGET, a code for the quantity to be gotten, and TODO, a code for the type of operation to get it with. When we say merely "GET something" we mean TODO = CAD.



Coding details: Entry GET: rB=exit rA=TOGET TODO=0.
Entry GET1: rB=exit TOGET=TOGET rA=TODO
if TODO < 0, we take special exit at step G7.

Temp Storage Used: TOGET, TODO, GEXIT, STOR, TEMP1, TEMP5
COUNT, CNO, NUMSTable, ~~[illegible]~~, SPEC, PVAR, COLUM, HOLD table, CMTX, LEFT, RIGHT, OP, RB, TEMP6 *

Subroutines Used: CONST, GET, MASTR, RSULT, STLOB.

When the temp storage is stored and restored again the entire high-speed loop's is stored. Therefore all ~~[illegible]~~ counters ~~[illegible]~~ is in the loop — ~~[illegible]~~ Q, WORD, LSTCH, A0-A4, ~~[illegible]~~ were chosen so that they would not change during the time the temp storage was given on the drums. Something like CNO, CMTX, STOR, LOOPL must not be kept in loop 5.

TODO=CAD G1. Set TODO = CAD.

ARRAY? G2. If TOGET is an array ~~or a procedure output variable~~ go to G11

T STORAGE? G3. If TOGET is a temp. storage code, go to G10

STORE PREV. RES. G4. If rA in the output contains something that must be stored away, compile a store instruction [Subroutine D].

CONSTANT? G5. If TOGET is not a constant go to G7

SUBR. F G6. If TODO = CAD and constant is zero compile a STC instruction. Otherwise take the constant off the constant stack and execute Subroutine F.

'ALMOST GOT'? G7. If special exit is called for exit.

COMPILE G8. Compile the desired TOGET-TODO instruction by using the following op-code: [Subroutine D]

TOGET MINUS TAG	TOGET ASSUMED TAG	TODO CAD	ADD	FAD	MUL	DIV	FMU	FDV	LOB
OFF	OFF	CAD	ADD	FAD	MUL	DIV	FMU	FDV	LOB
ON	OFF	CSU	SUB	FSU	MUL	DIV	FMU	FDV	LOB
OFF	ON	CAA	ADA	FAA	mul/mud		FMA	FDA	LOB
ON	ON	CSA	SUA	FSA	mu/mu		FMA	FDA	LOB

[Subroutine D]. Exit.

SIMPLE CASE? G10. If TOGET is already in rA in the output, exit. Else to G4.

~~[illegible]~~ G11. INITIALIZE If TOGET is a non-procedure variable, set SPEC to base address, set PVAR to zero. Otherwise set PVAR to input information address and set SPEC to zero.

DIMENSION? G12. If TOGET is a vector, go to G18. If TOGET is a procedure output variable, go to G25. Otherwise TOGET must be a matrix.

CONSTANT? SUBSC.? G13. If TOGET is a non-procedure matrix and its first subscript is a constant, go to G14. Else go to G15.

INTEGRAL MULT. G14. Pull constant off constant stack multiply it by the dimension, and add it to SPEC. To G18.

HAND STOCK FORM 14111

MULT. BY COLS G15. Compile to multiply the first subscript by the number of columns [Subroutine H].

CONST. SUBSC.? G16. If the second subscript is a fixed point constant, go to G22

ADD SUBSC. G17. Compile to add the second subscript to the previous result. [Subroutine H]. Go to G23

SIMPLE SUBSC.? G18. If subscript is a fixed point constant, go to G24

G19. If it is floating point ~~and TOGET is a procedure~~ variable, go to G21

LDB SUBSC G19. Compile to LDB with the subscript [Subroutine G]. To G22

LDB PVAR G20. Compile to LDB with PVAR [Subroutine G]. To G26

GET SUBSC G21. GET the subscript [Subroutine G]. To G23

PAT IN B G23. If the result in rA is floating point compile FAD (581000000) or FSU (5810 0) according as rA contains the true result or the negative of the result. [Subroutines F, D]. Then if TOGET is a procedure variable, also compile ADD or SUB (PVAR). Then compile STR 4001 LDB 4001 [illegible]

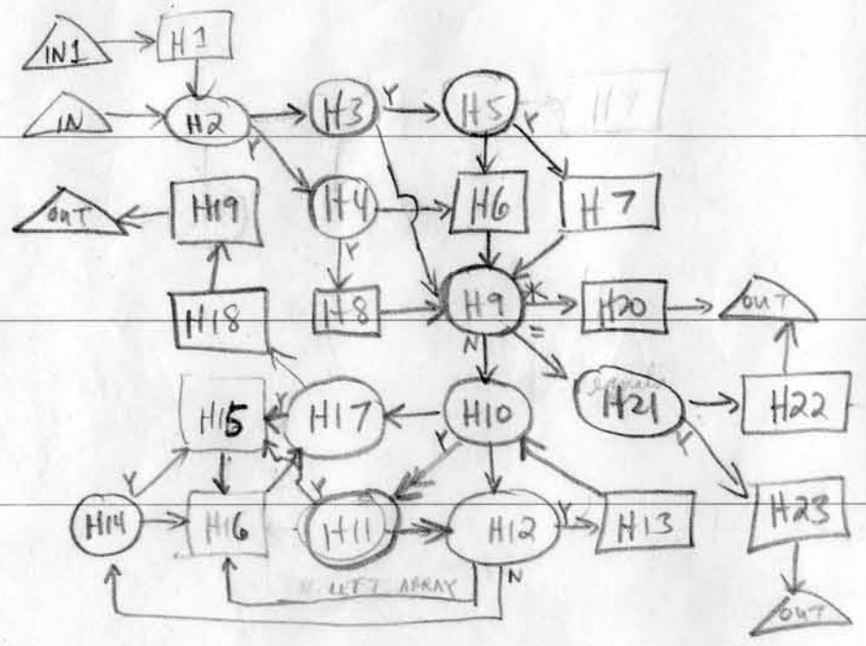
INTERNAL ADD G22. Take the constant off constant stack and add it to SPEC.

INTERNAL ADD G24 Take the constant off constant stack and add it to SPEC.

DUMMY ARRAY? G25 If TOGET is a procedure variable, go to G20

RETURN G26 Change TOGET to SPEC, with or without B modification. Restore temp storage. Go to 64.

H. MASTR routine. This general routine takes charge of compiling the instructions necessary to do addition, subtraction, multiplication, division, exponentiation, and substitution. Remarks: this routine is a little tricky because it uses GET as a subroutine, and GET uses MASTR as a subroutine, and MASTR in turn uses GET. The process is rigged so that the nesting doesn't go any deeper than this. Inputs to MASTR are an operation (OP) and LEFT and RIGHT operands. (OP is stored as a floating point number changed to fixed point by the MASTR routine.) The coding turned out by MASTR is efficient in most respects. The major inefficiency is in the treatment of arrays with very simple (e.g. constant) subscripts, when occasionally store instructions are compiled in anticipation of a difficult array subscript. The other weakness is in fixed point multiplication when LEFT has an absolute value to be taken and both LEFT and RIGHT are simple variables — this case was of such infrequent occurrence it was not accommodated.



SET LEFT, RIGHT — H1. The top of the operand stack is removed and placed in RIGHT. Then the (new) top of the operand stack is removed and placed in LEFT.

RIGHT FIXED? — H2. If RIGHT is fixed point, go to H4

LEFT FIXED? — H3. If LEFT is fixed point, go to H5; else to H9

LEFT FIXED? — H4. If LEFT is fixed point, go to (H8); else to H6

OP = ? — H5. If OP is "=", go to H7

FLOAT — H6. We have one fixed pt. operand and one floating pt. operand. We will float the fixed one; if LEFT is the fixed operand and both RIGHT and LEFT are arrays, we first GET RIGHT [Subroutine G] and rather than keep track of which is the fixed operand, so the subscript stack will be last-in-first-out. Then we reserve room for the float subroutine in case it has not been reserved before; Finally we GET the fixed point operand (Subroutine G) and BCALL the float subroutine (Subroutine E). To H9.

If the operand to be floated is a constant, we merely change the constant to floating point. Otherwise FIX

H7. Same as H6 except we are changing RIGHT from floating point to fixed point and therefore no test for constants. To H9.

FIXED POINT — H8. Set OP as fixed point.

OP = OR *? — H9. Now the preliminaries are all taken care of and we are ready to do the real work. If OP is "=", go to H21; if OP = "*", go to H20.

RIGHT ARRAY? — H10. If RIGHT is an array, go to H11. If LEFT is already in rA except possibly for its sign, go to H17. Else to H15.

LEFT ARRAY? — H11. If LEFT is also an array, go to H15

COMMUTE? — H12. If LEFT is an array, go to H14. If OP is non-commutative, go to H14.

COMMUTE — H13. Set OP as non-commutative. Interchange LEFT and RIGHT. Go to H10.

RIGHT ARRAY? — H14. If RIGHT is not an array, go to H16

GET RIGHT — H15. GET RIGHT [Subroutine G]. Change RIGHT to a temp storage code.

GET LEFT — H16. GET LEFT [Subroutine G]. Change LEFT to a temp storage code.

Coding details: Entry MASTR: rA = exit; RIGHT, LEFT, OP are set up.
Entry MAST1: rB = exit; rA = OP; RIGHT, LEFT will be set up from NSTAK.
Entry MAST2: same as MAST1 except rB is arbitrary, exit will be made to FIXT1
Entry MAST3: rA = RIGHT; rB = CNAME − 1; OP, MEXIT set up; LEFT will be set up from NSTAK
Entry MAST4: rA = zero; RIGHT, LEFT, OP, MEXIT are set up.

Temp. Storage Used: MEXIT, MFL, RIGHT, LEFT, OP, SEG1, SEG2, HIGHL, NUMST table, T, STOR, TEMP1, , COLUM, PVAR

Subroutines Used: GET, BCALL, FIXT3

Comment: If LEFT is an array, commuting will do no good; go to H16.

UNDEFINED OP? H17. Now we have ± LEFT in register A and RIGHT is sufficiently simple to finish the operation, unless OP is fixed point multiply or divide and RIGHT is to be taken with absolute value. In this ~~instance~~ case go back to H15.

FIX SIGN? H18. If rA contains −LEFT, negate the sign of RIGHT. (If OP is +, the sign of the compiled result is the sign of LEFT, otherwise it is the sign of RIGHT.) Now if OP is divide, compile either CLR (floating point) or SRT 10 (fixed point) [Subroutine D]

COMPILE H19. GET RIGHT with the proper OP code [Subroutine G entry IN1]. If OP is fixed pt. multiply, compile SLT 10. [Subroutine D] Exit.

CALL POWER H20. Reserve room for the power subroutine in case it hasn't been used yet. Then GET RIGHT [Subroutine G]. Compile STA 4000 [Subroutine D]. GET LEFT [Subroutine G]. BCALL the power routine [Subroutine E]. Exit.

LEFT ARRAY? H21. If LEFT is an array, go to H23

LEFT = RIGHT H22. GET RIGHT [Subroutine G]. Then ~~if~~. If LEFT is not the name of a procedure currently being defined compile STA LEFT [Subroutine D]. Exit.

LEFT = RIGHT H23. If RIGHT is also an array, GET RIGHT [Subroutine G] and reflect it by a temp. storage code. Then "almost get" LEFT [Subroutine G calling for special exit at step G7J. Then GET RIGHT [Subroutine G] followed by STA (LEFT) [Subroutine D]. Exit.

Coding Details on "hidden" subroutines FIXT and RSULT.

Entry FIXT2: sets TMIN = min (TMIN, T) and sets T = max (COUNT+1, T). Exits to HIER.

Entry FIXT3: same except rA = exit.

Entry FIXT5: rA = exit. sets TMIN only.

Entry RSULT: sets STOR = T, T = T−1, ~~on exit~~ rA = STOR with arithmetic of OP and sign of LEFT, rB = CNAME.

Entry FIXT1: FIXT2 followed by RSULT followed by putting (rA) on top of operand stack, exits to HIER.

Entry FIXT4: FIXT1 with rA = OP, LEFT positive.

I. PRESCANNER.

The general plan of attack in this compiler can be summarized as follows: The program looks at the input string language character by character from left to right... as soon as it sees something it can do, it does it; meanwhile it saves information about what the input has previously contained in counters and stacks.

The control of this process is handled in the following way: There are five locations inside called 10, 11, 12, 13, and 14. Characters come in from the input string into either 12, 13, or 14 and from there they proceed leftward until they are processed. Most characters get all the way to 10 before they are processed, but some get no farther than 13.

When characters originally come into 12, 13, or 14 they are in Cardation code or in a special code for special characters. But the contents of 10 and 11 are in a completely different, more versatile compiler code. 12 is the transition point where the codes are converted from one to another.

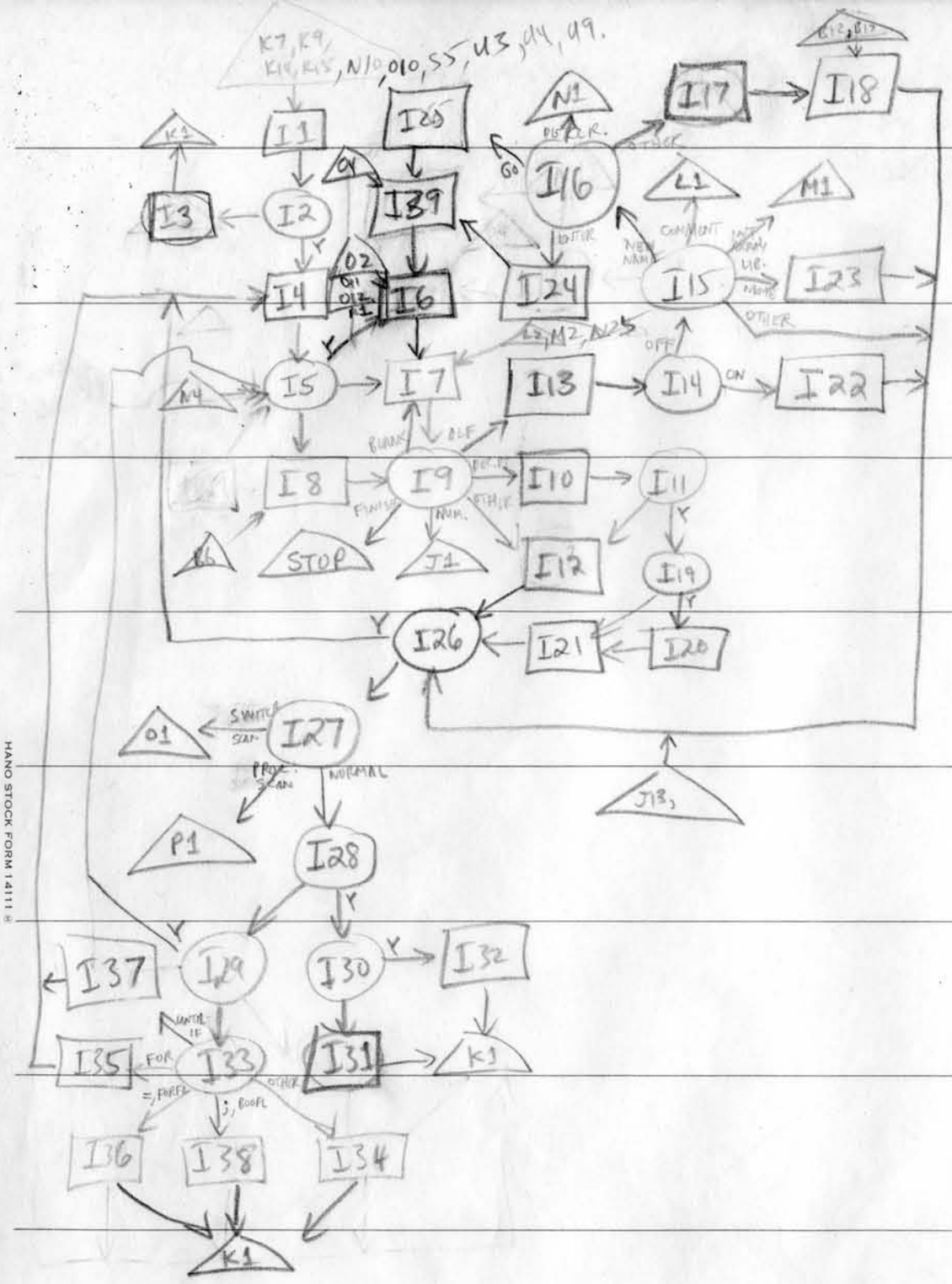
A "scanner" is a routine which looks at a string language by scanning two (or more) adjacent characters at a time. There are four scanners in this compiler: the main arithmetic scanner [routine K] which operates from 10 and 11; the prescanner [routine I] which operates from 11, 12, 13, and 14; and the SWITCH and the procedure call scanner [routine O] which operates from 11 and 12. The Rube-Goldberg procedure call scanner correctly begins by operation from 10 and 11 in conjunction with the arithmetic scanner; then shifts to 11-12 after a semicolon.

A "condenser" is a routine which takes a look at sequential characters in a string language and condenses them into a single entity or discards them entirely. There are five condensers in this compiler: the COMMENT condenser [routine L], the identifier-building condenser [subroutine B], the procedure input string condenser [part of routine N], the integer and array declaration condenser [routine M], the FORMAT condenser [part of routine N], and the constant condenser [routine J].

The Pre Scanner really seems to have the most control over the whole process, for it usually decides which scanner or condenser is to be used next. The job of the prescanner is not only to control the flow of the program, however; it also must edit the incoming language into a form which the other routines can digest. For example, it inserts multiplication signs when implied multiplication is indicated, it handles the translation of 12 from the original two-digit code to the compiler code and it decides whether identifiers are labels or not. Since the prescanner occasionally inserts a character there is an extra location called 11.5 for inserts.

Compilation starts with 10 = a semicolon, 11, 11.5 and 12 blank. A counter Q tells whether 13 and 14 have any information: when Q=0 it means they don't, when Q=1 it means 13 does, and when Q=2 both 13 and 14 are filled. Q is initially zero. After the initialization, compilation begins in box I7

10 = 11	I1.	Move 11 into 10
11.5 BLANK?	I2.	Go to I4 if 11.5 is blank.
INSERT 11.5	I3.	Move 11.5 into 11, clear 11.5. To K1.
11 = 12	I4.	Move 12 into 11
Q ZERO?	I5.	If Q>0 go to I8; else to I7
11 = BLANK	I6.	Blank out 11.
12 NXTCH	I7.	NXTCH into 12 [Subroutine A with Q=0]. To I9
MOVE 13,14 LEFT	I8.	Move 13 into 12, 14 into 13, and set Q = Q-1
TEST 12.	I9.	If 11 = FINISH, stop compilation. If 12 is blank, go to I7. If 12 is a number, go to J1. If 12 is a letter, go to I13. If 12 is a decimal point, go to I10. Else to I12.



Coding Details: Temp Storage Used: #A0, A1, A2, A3, A4, INSW, Q, PARTS table, EQUIV, LOOPL, HIGHL, SEG1, SEG2, SEG3, TEMP1, FORFL, BOOFL, CNAME, CDUO, PROCX table.

Subroutines Used: NXTCH, ACALL, OUTPT, LABLE

IP used

A3 NXTCH I10. NXTCH into A3 [Subroutine A with Q=1].

COLON? I11. If A3 is a second decimal point, go to I19. Else to I12

TRANSLATE I12. Change A2 into compiler code. To I26.

SUBR. B I13. Move A2 into A3 and BUILD an identifier [Subroutine B]. Set A2 to the compiler code for this identifier.

LABEL SWITCH. I14. If the LABEL switch is on, go to I22. (LABEL switch normally off).

TEST A2. I15. If A2 = COMMENT, go to L1.
If A2 is a library function or procedure name that has not yet been used, go to I23.
If A2 = INTEGER or ARRAY, go to M1.
If A2 is a letter followed by a digit, possibly an INTEGER declaration, go to I16. Else to I26.

TEST A1. I16. If A1 = INPUT, OUTPUT, FORMAT, SUBROUTINE, or PROCEDURE, go to N1. If A1 = ENTER, go to I24.
If A1 = GO, go to I25.

PREFIX MATCH? I17. In this case A2 is either a statement label or a variable. Let's act under the assumption it's a variable. Find out if any prefixes for this label have occurred [Subroutine B, calling for special exit at steps B12 and B17].

REFER VARIABLE I18. Adjust arithmetic of the variable accordingly. If loop G is not full, put the variable there, otherwise put it on the drum. Go to I26.

A1 NEW? I19. If A1 has not been coded as a variable, go to I21.

DEFINE LABEL I21. Define A1 as a label equal to DRUML+LOOPL. To I26.

UNDO VARIABLE I20. Free up the location used for the variable and recode A1 as a label.

LABEL CODE A2 I22. Replace A2 by the "label code", which is:
a) 5 CUB equiv for a regular label
b) 1 BUN equiv for a procedure dummy label
c) a simple variable code, for a defined procedure name
d) a procedure output variable code, for a dummy procedure name.
Go to I26

DEFINE SUBR. I23. Reserve space for the library routine on the drum if needed. Calculate address of subroutine and enter it into the compiler code table. Go to I26

ENTER A2 I24. Compile ACALL to the label code for A2. [Subroutine E] Go to I26.

GO TO A2 I25. Compile the label code for A2. [Subroutine D] Go to I39.

A1 BLANK? I26. If A1 is blank, go to I4.

TEST SWITCH. I27. At this point, A1 and A2 are both in compiler code. A switch is set normally to go to I28, or to go to the SWITCH scanner 01, or to go to the procedure call scanner 05.

A1 RT. DELIM.? I28. If A1 is a right parenthesis, END, constant or simple variable, go to I30

A2 PLUS ? I29. If A2 is a plus sign, go to I5 (this ignores unary +); else go to I33

A2 MINUS? I30. If A2 is a minus sign, go to I32 (this changes binary - to unary -).

CHECK IMPL. MULT. I31. If A2 is a constant, any variable, function name, or procedure name, or left parenthesis or BEGIN, set A1.5 to multiplication sign. (This is to test for implied multiplication.) Go to K1

A1.5 = PLUS I32. Set A1.5 to plus sign. Go to K1.

TEST A1, FLAGS I33. If A1 = FOR go to I35
If A1 is an equals sign and the FOR flag is on, go to I36
If A1 = UNTIL or IF go to I37
If A1 is a semicolon and the BOOLEAN flag is on, go to I38

CHECK NULLSTRING I34. If A1 is any of (left parenthesis, BEGIN, semicolon, STOP, comma) and A2 is any of (right parenthesis, END, semicolon, comma) set A1.5 to "nullstring" code. Go to K1.

FORFL ON I35. Set FOR flag on. Go to I4

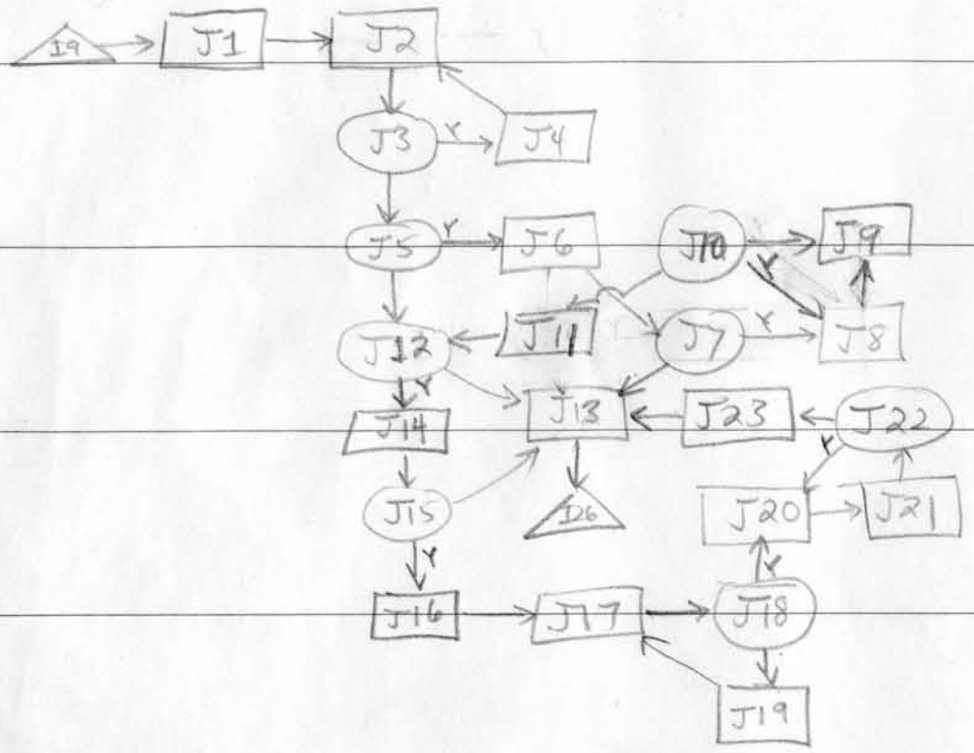
A1 = FOR I36. Replace A1 by "FOR". Set FOR flag off. To K1

BOOFL ON I37. Set BOOLEAN flag to A1. Put DRUML + LOOPL with Boolean storage code on top of operand stack. To I4

Increment the "DO" counter. A1=BOOFL I38. Replace A1 by BOOLEAN flag. Set BOOLEAN flag off. To K1.

I39. Set A1 = CONTINUE. To I5

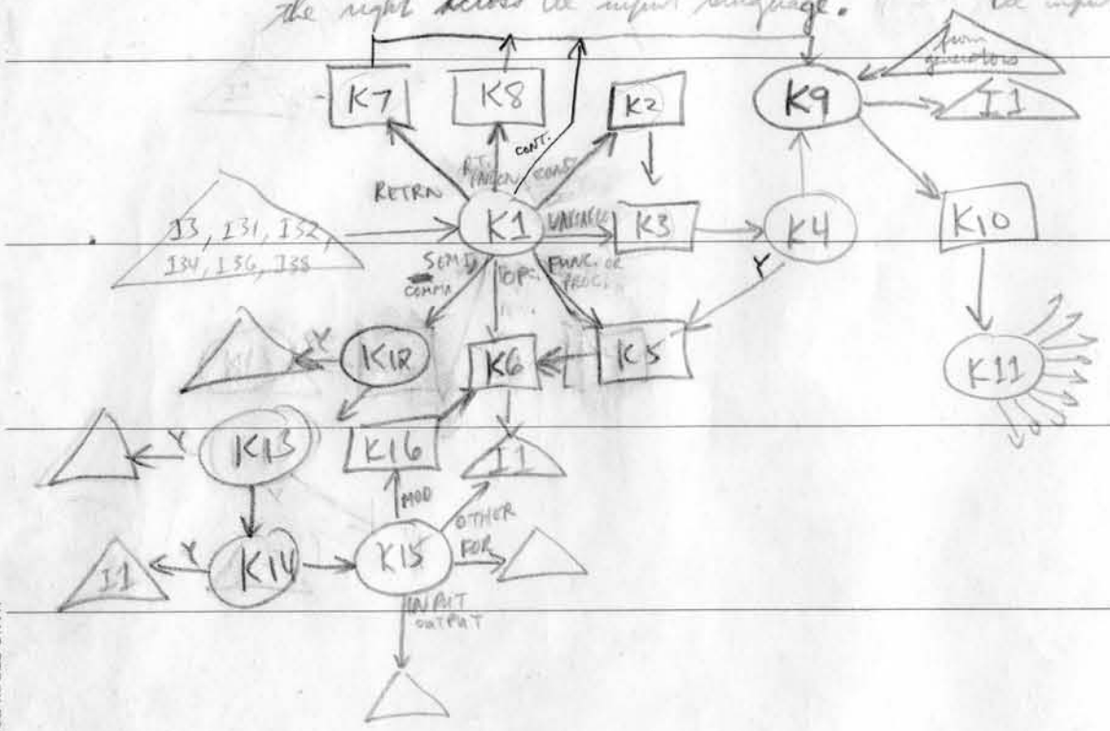
J. Constant condenser. This routine changes constants from the input string code to ZOS code. I1 is entered only from the Prescanner at a time when Q=0 and A2 is the first numeric character.



INITIALIZE	J1.	Set N to A2, ARITH to fixed, MU = 0.
A3 NXTCH	J2.	NXTCH into A3 [Subroutine A with Q = 1]
A3 NUMERIC?	J3.	If A3 is not numeric go to J5.
BUILD N	J4.	Set N to 10N+A3. To J2.
A3 DEC. PT.?	J5.	If A3 is not a decimal pt go to J12
A3 NXTCH	J6.	NXTCH into A3 [Subroutine A with Q = 1].
A3 NUMERIC?	J7.	If A3 is not numeric, the decimal point was a multiplication symbol which will be reinserted later because of implied multiplication. In this case go to J13.
BUILD N	J8.	N = 10N + A3, MU = MU+1.
A3 NXTCH	J9.	NXTCH into A3
A3 NUMERIC?	J10.	If A3 is numeric, go to J8.
FLOAT	J11.	Set ARITH floating. Convert N to floating point notation (divide by 10**MU).
A3 * ?	J12.	If A3 is an asterisk go to J14
CODE A2.	J13.	Set A2 to constant code. To I26.
A4 NXTCH	J14.	NXTCH into A4 [Subroutine A with Q = 2]
A3 * ?	J15.	If A3 is not an asterisk go to J13.
FLOAT	J16.	If N hasn't yet been converted to floating point, set ARITH floating and convert N. Set SIGN plus, MU zero
A3 NXTCH	J17.	NXTCH into A3.
A3 NUMERIC?	J18.	If A3 is numeric go to J20.
+ OR − EXP.	J19.	Set SIGN plus or minus according to A3. To J17
BUILD MU	J20.	MU = 10MU + A3.
A3 NXTCH	J21.	NXTCH into A3
A3 NUMERIC?	J22.	If A3 is numeric go to J20.
ADJUST EXP.	J23.	Multiply N by 10**±MU acc. to SIGN. To J13.

Coding Details: Temp. Storage Used: TEMP5, TEMP6, TEMP7, N
Subroutines Used: NXTCH

K. Arithmetic Scanner. This consists mostly of tests on the type of A¢. The scanner has two main functions: a) to control what goes onto the operator and operand stacks; (b) to detect when an operation is ready to be compiled for and in each an event to go to the proper generator. Between every two times the scanner is entered, the symbol-pair A¢, A1 has been moved to the right across the input language as seen by the scanner, A1 has been edited by the prescanner into a convenient form; a few non-arithmetic things come through but not many.



TEST A¢ K1. If A0 is an operator, a left parenthesis or BEGIN, go to K6
If A0 is a semicolon or comma, go to K12
If A0 = RETURN, go to K7
If A0 = CONTINUE, go to K9.
If A0 is function or procedure name, go to K5
If A0 is a number or "subscripted", go to K3
If A0 is a constant, go to K2
Otherwise A0 is a right parenthesis or END. Go to K8.

STACK CONSTANT	K2.	Put constant onto constant stack
STACK NAME	K3.	Put A¢ onto operand stack.
SUBSC. VAR.?	K4.	If A¢ is not a subscripted, go to K9
CHECK A1 (	K5.	ALARM if A1 is not a left parenthesis
STACK OP	K6.	PUT A¢ onto operator stack, go to I1
RUN EXIT	K7.	Compile RUN(EXIT)[Subroutine D, entry W7]. Go to K9.
UNSTACK (	K8.	ALARM if top of operator stack does not pair with A¢. Remove top of operator stack. Go to K9
HIERARCHY RIGHT?	K9.	If the hierarchy of the top of the operator stack is not greater than the hierarchy of A1, go to I1. (In this test we associate "hierarchies" with left and right parentheses, comma and semicolon, as well as with normal operators, but the "hierarchy" of a left parenthesis is tested only from the right, and the "hierarchy" of right parenthesis, semicolon, and comma are tested only from the left.) The hierarchy table is as follows:

(,), BEGIN, END, FOR	00
SEMICOLON,	02
UNTIL, IF, DO	04
STOP, =	06
COMMA	08
SWITCH	10
OR	12
AND	14
NOT	16
LSS, EQL, etc	18
slash in MOD	19
PLUS	20
DIVIDE	22
TIMES	24
POWER, MINUS	26
all others	∞

Coding Details: After entry to the generator, rA = 0, rB = CNAME - 1,
rR:88 = OPcode:08, COUNT = 0. These conditions were found to be most useful in the generators

Temp. Storage used: COP, OPST table, A0, CN0, NHMST table, CNAME, NSTAK table

Subroutines used: OUTP7.

UNSTACK OP TO GENERATOR	K10.	Remove top of operator stack
	K11.	Go to generator for this operation. Generators are in section P-4

FOR ~~OR PROC.~~ CALL? K12 If top of OPST = FOR go to ——.

~~PROC. CALL?~~ K13 If next from top of operator stack is a procedure name, go to Z1.

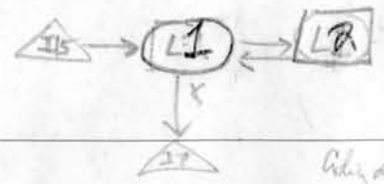
Δφ SEMICOLON? K14. If Δφ is a semicolon, go to I1.

TEST OUTSIDE OP. K15. If next from top of operator stack is INPUT or OUTPT, go to ——.
If it is MOD, go to K16
If it is FOR, go to ——
Otherwise go to I1.

Δφ = DIVIDE K16 Set Δφ = divide with hierarchy 19. Go to K6.

12
11
10
9
8
7
6
5
4
3
2

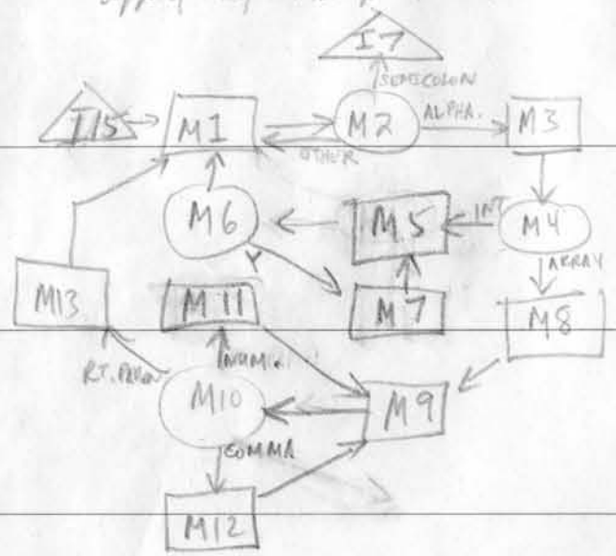
L. COMMENT condenser. This routine is so trivial it requires no COMMENT.



A3 NXTCH L2. NXTCH into A3 [Subroutine A with Δ=1]. To L1.
A3 SEMICOLON? L1. If A3 is a semicolon, go to I7.

Coding details: Subroutines used: NXTCH.

M. INTEGER AND ARRAY DECLARATION condenser. Entry is only from Precursor when A2 = INTEGER or ARRAY, A3 = blank. Remarks: Prefix applies only to first simple variable which appears only after the prefix

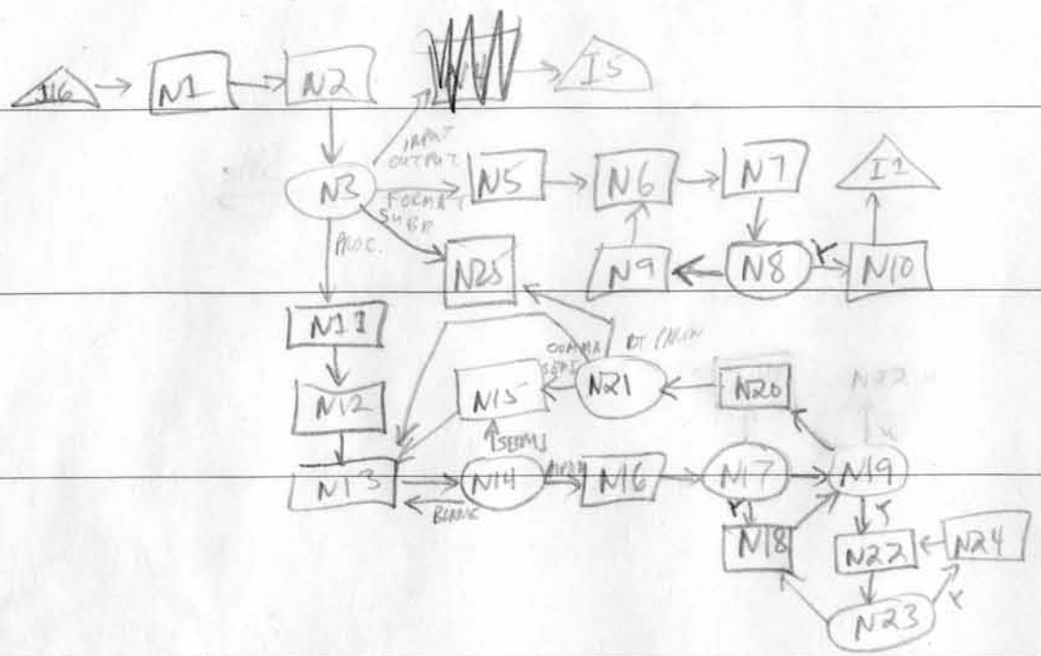


A3 NXTCH M1. NXTCH into A3 [Subroutine A with Δ=1].
A3 SEMICOLON? M2. If A3 is a semicolon, go to I7. If A3 is alphabetic, go to M3. Else to M1.
BUILD IDENT M3. BUILD identifier [Subroutine B].
A2 ? M4. If A2 is ARRAY, go to M8. Otherwise A2 is INTEGER.
FIX IDENT M5. Make this name (prefix) have integer meaning in compiler code
A3 PERIOD? M6. If A3 is not a period, go to M1,
PREFIX M7. Set A3 = blank and go to the prefix routine [Subroutine B, entry IN1]. To M5.
INITIALIZE M8. Set dimension to 1, MU = 0
A3 NXTCH M9. NXTCH into A3 ignoring all blanks [Subroutine A, with "last character" set to blank]
A3 ? M10. If A3 is minus, go to M11, If A3 is a comma, go to M12. Else to M13.
M11. MU = 10MU + A5. Go to M9.
M12. Set dimension to 2, NU = MU, MU = 0. Go to M9
M13. If dimension is one, reserve MU locations. If dimension is two, reserve MU·NU locations. Set compiler code for this name with proper base address and dimensionality, retaining INTEGER status if previously declared. Go to M1.

Coding Details: Temp Storage used: A3, TEMP5, TEMP6, TEMP7, HIGHL, PART2-table, LSTCH
Subroutines used: NXTCH, BUILD, BL01.

The "middle phase" is handled by the normal scanner.

N1. DECLARATIONS. This routine processes the "beginning phase" of INPUT, OUTPUT, FORMAT, SUBROUTINE, and PROCEDURE declarations. The "ending phase" of these declarations is done by generators. N5 - N10 is the FORMAT condenser, N12-24 is the procedure input string condenser.



COMPILE SKIP — N1. Compile CAB (around) [Subroutine D, entry INT]. This is done so that the coding produced by the declaration is bypassed at the time it appears.

N2 LABEL — N2. Define N2 to be a label located at NEXAL.

A1? — N3. If A1 is INPUT or OUTPUT, go to N4
If A1 is FORMAT, go to N5
If A1 is SUBROUTINE, go to N25
Otherwise A1 is PROCEDURE, go to N11

MAKE EXIT LINE — N4. Step LOOPL by one to make room for an exit line. To I5

INITIALIZE — N5. Set A2 = left parenthesis, set ~~STAR = 0~~, FWORD = blank. [Subroutine A with A2 = 1 and A3 = 1 and will find a comma or a left parenthesis. We are inside a procedure or subroutine declaration. Subroutine A will set STAR = 1 if a comma is found, and will leave STAR unchanged if a left parenthesis is found.]

A3 NXTCH — N6. NXTCH into A3 [Subroutine A with A2 = 0 and A3 = 1]

BOOKKEEPING — N7. If A3 is an asterisk and STAR = 0, set STAR = 1. If A3 is a right parenthesis and STAR = 0, go to N8 and set LEVEL = LEVEL - 1. If A3 is an asterisk and STAR ≠ 0 set STAR = 0. Otherwise do nothing.

END OF STRING? — N8. If LEVEL ≠ 0 go to N6.

RECORD A3 — N9. Add A3 to FWORD. If FWORD is full, punch it [and Subroutine C] and set it to blank. Go to N6.

FINISH UP. — N10. Punch FWORD [Subroutine C]. Go to I1.

A2 STATUS — N11. Set temp. storage counter so as not to overlay any which occurred previously. Define A2 to be a procedure label with global significance. Set up procedure heading.

INITIALIZE — N12. Set I = 0, J = 0.

A3 NXTCH — N13. NXCH into A3 [Subroutine A with A2 = 1]

A3? — N14. If A3 is a semicolon go to N15. If A3 is alphabetic, go to N16. Otherwise A3 ought to be blank; go to N13.

STEP I — N15. Set I = I + 3, J = I. Go to N13.

BUILD NAME — N16. BUILD an identifier [Subroutine B]. [If A3 = 0, N17 ≠ 0, go to N14.]

A3 NXTCH — N17. Put next nonzero character into A3 [Subroutine A, A2 = 1]

A3 LEFT PAREN? — N18. If A3 is a left parenthesis, go to N21. CLEAR if A3 is alphabetic or numeric.

MAKE STATUS — N19. Step LOOPL by one. Set the identifier last encountered to have the meaning specified by J:

J = 1 simple variable; 2 procedure dummy vector; 3 procedure dummy notype; 4 procedure output switch; 5 procedure dummy vector; 6 procedure dummy notype; 7 procedure label; 8 procedure dummy procedure

Coding details: Temp storage used: A1/A2, NEXT table, DECIML, LOOPL, TEMP5, TEMP6, TEMP7, THISE, PROCC, PROCT
Subroutines used: NXTCH, DFINE, PUNCH, OUTPT,

N19 (continued) Set J = I.

A3 ? — N20. If A3 is a semicolon go to N15. If A3 is a comma go to N25. Otherwise A3 is a right parenthesis; go to N25.

A3 NXTCH — N21. J = J + 1. Set A3 = next non-blank character [Subroutine A, A2 = 1, last character = blank]

A3 COMMA? — N22. If A3 is a comma, go to N24. Otherwise A3 is a right parenthesis; go to N18.

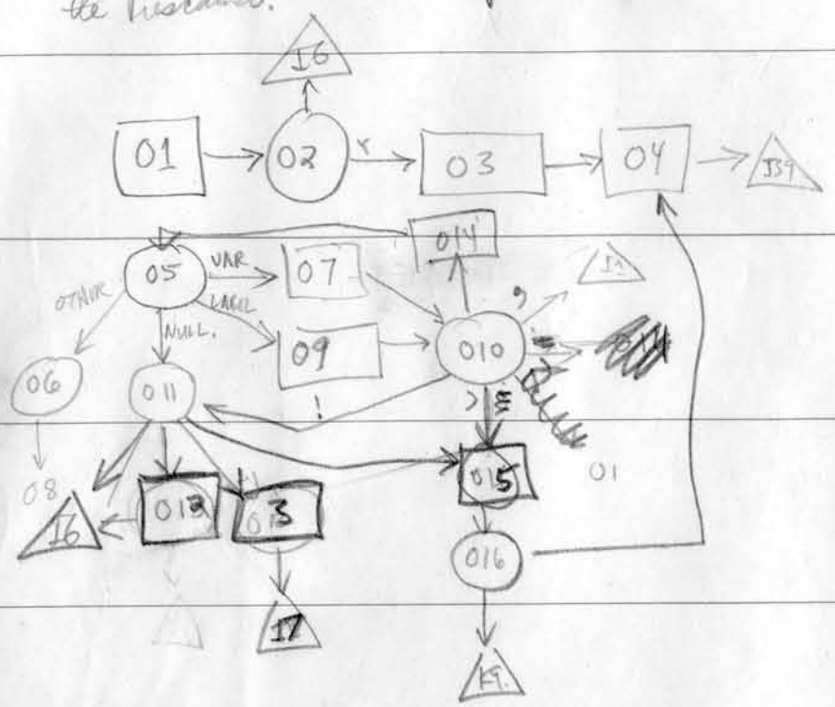
STEP LOOPL — N24. Step LOOPL by one. To N22

STORE EXIT — N25. Compile STA epil (for either procedure or subroutine) [Subroutine D]. Go to I7.

Three scanners alter the normal mode of compilation wildly.

0. Switch/procedure call scanner. Read the SWITCH and PROCEDURE generators before this, because they do the preliminary work for the second phase (from the Scan) is made to the generator P1; the second phase enters at 05 from the Prescanner.

Remarks: The entrance to the procedure call scanner in its first phase enters at 01 from the Prescanner. The switch scanner enters at 01 from the Prescanner.



COMPILE CUB — 01. Set PRNML = TEMP7 = TEMP7+1. Compile label code for A1 [Subroutine D].

A2 RIGHT PAREN? — 02. If A2 is a right parentheses, go to 03. Else to I6.

CLEAN UP — 03. Fix up forward reference from (A0) [Subroutine C].

RESET SWITCHES — 04. Reset I27 switch and label switch to normal position. Adjust A0 so it will be bypassed at next entrance of SCAN. To I39.

TEST A1? — 05. If A1 = nullstring, go to 011
If A1 is a label code for a non-procedure label, go to 09
If A1 is a simple variable, go to 07.
Otherwise go to 06 (testing A1)

DUMMY ARRAY? — 06 If it is a procedure dummy array, go to 08

CALL BY NAME — 07 Here we have an ordinary call by name which is to be entered as input to a procedure. Compile CAD(const) [Subroutines D, F], STA (procedure input storage). To a matrix, do this twice. To 010

TRANSFER NAME CALL — 08. Compile CAD(current procedure input storage) [Subroutine D], STA (called procedure input storage). For a matrix, do this twice. To 010.

CALL BY LOCATION — 09. Compile CAD 2+7, BUN 2+7, label code, [Subroutine F, inc. for 4 locations in loop 7 buffer & Subroutine D], STA (called procedure input) [Subroutine D].

A0? — 010. If A0 is a right parenthesis, go to 016
If A0 is a comma, go to I?.
If A0 is a semicolon, go to 014
(Otherwise A0 is an exclamation mark).

A2? — 011. If A2 is a right parenthesis, go to 015
If A2 is a left parenthesis, go to 013.
If A2 is a semicolon, go to 012
Otherwise A2 is a comma; go to I6

LABEL SWITCH ON, SCAN TO ")" — 012. Set label switch on, go to I6.

013. NXTCH into A3 until it is a right parenthesis [Subroutine A will Q=4]. Then set A1 = nullstring. To I7.

SET FLAGS — 014. Set A0 = exclamation mark. Set I27 switch to jump to 05. To 05.

ACALL — 015. Compile to ACALL the procedure. [Subroutine E]

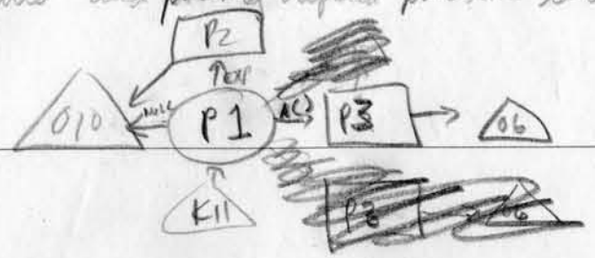
FUNCTION? — 016. If A0 = right parenthesis, we are using the procedure as a function; go to K9. Otherwise A0 is an exclamation point and we go to 04.

Coding Details: TEMP Storage Used. OPST table, CNAME, CMTX, A0, A1, A2, TEMP1, TEMP6, TEMP7, PROSW, SWITCH, DRUML, etc.

Subroutines Used; NXTCH, CONOT, OUTPT, PRSUB, FULL, GET, LABLE, ACALL.

Generators. The remaining routines are the special processes which handle their own feature of the language.

P. Procedure Generator. Used for library procedure calls and personally defined procedure calls.

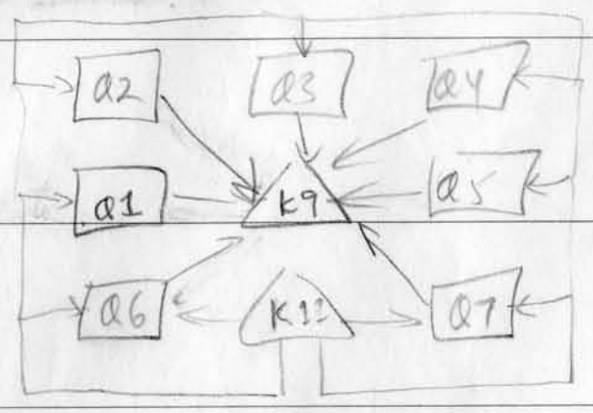


TEST TYPE — P1. If the top of the name stack is an array and the top of the subscript stack is a nullstring, go to P3.
If not, go to 010.

CALL BY VALUE — P2. Get it [Subroutine G], and compile STA (procedure input storage). To 010.

PULL NULLSTRINGS — P3. Remove all "nullstrings" from subscript stack, go to 06 (testing top of name stack).

Q. Arithmetic Generators. Addition, Multiplication, Division, Exponentiation, Replacement go to Q1. Minus goes to Q2. ABS goes to Q3. MOD goes to Q4. Library functions go to Q5. Arrays go to Q6. "STOP" goes to Q7.



+ · / * = — Q1. If it is a simple variable or a temp storage location raised to the second power, change it to a multiply. Compile for the desired operation [Subroutine H, entry IN1]. To K9.

MINUS — Q2. Reverse sign of top of operand stack. To K9

ABS — Q3. Set absolute val of top of operand stack on, set it positive. To K9

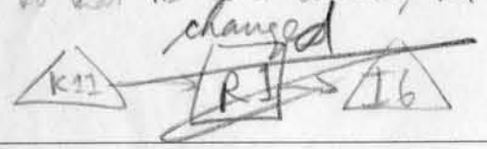
MOD — Q4. Compile SET 10 [Subroutine D]. To K9.

LIBRARY CALL — Q5. GET top of operand stack (ARGEN if first) [Subroutine G]. BCALL the library routine. [Subroutine E]. To K9.

ARRAYS — Q6. Move subscripts off operand stack to subscript stack. If they are wrap, GET them first [Subroutine G] and replace them with temp storage symbols. Take care not to introduce any holes in subscript or constant stacks! To K9.

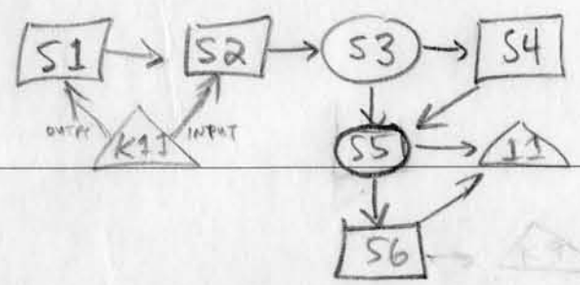
STOP — Q7. If top of operand stack is not a nullstring, GET it [Subroutine G]. Remove it from stack. Compile STOP 0137 [Subroutine D]. To K9.

R. Switch Generator. When R1 is entered the hierarch has been fixed so that A1 is a comma, A2 a left parenthesis.



SET UP FOR SWITCH — R1. GET top of operand stack [Subroutine G]. If it is floating, compile FAD (5810000000) [Subroutine D]. Compile STA 4001, CIB 4001 [Subroutine D]. Compile -BUN 11; Set TEMP7 = ORUML; Compile CMB (next). [Subroutine D] Set I7 switch to exit to 01. To I6.

S. Input-Output Generators. S1 is OUTPUT entry, S2 the INPUT entry.



S1. GET top of operand stack [Subroutine G].

S2. BCALL the coroutine line [Subroutine E].

S3. If INPUT, go to S4; if OUTPUT go to S5

S4. Set RIGHT = temp storage code will save arithmetic at top of operand stack. Compile to store [Subroutine H, OP= replace].

S5. If A0 is a comma, go to I1; else it is a semicolon, go to

S6. Compile CAD 0+7, CIRA 4 [Subroutine D]. BCALL the coroutine line [Subroutine E]. To I1.