```
        LICENSED MATERIAL - PROPERTY OF DATA GENERAL CORPORATION
0001 HANDO    MACRO REV 06.30          22:11:12 10/12/80
                      .TITLE HANDOUT01
02                    ; ************************************************************
                      ; ************************************************************
                      ; ************************************************************
05                    ; ************************************************************
06                    ; ************************************************************
07                    ; ************************************************************
08                    ; ************************************************************
09                    ; ************************************************************
10                    ; ************************************************************
11                    ; ************************************************************
12                    ; ****************                        ****************
13                    ; **************** S105 INSTRUCTION SET HANDOUT ****************
14                    ; ****************                        ****************
15                    ; ************************************************************
16                    ; ************************************************************
17                    ; ************************************************************
18                    ; ************************************************************
19                    ; ************************************************************
20                    ; ************************************************************
21                    ; ************************************************************
22                    ; ************************************************************
23                    ; ************************************************************
24                    ; ************************************************************
25                            .EJEC
```

0002.HANDO

```
01                ;       ***** THE FOLLOWING SETS UP ZREL MEMORY SPACE
02                        .ZREL

                  ;       ***** THE FOLLOWING SETS UP NREL MEMORY SPACE
                          .NREL
06
07
08                ; NOTE: THE FOLLOWING FOUR LINES ARE LABEL EQUATES.
09                ;       NOTICE THAT THESE LABELS GENERATE NO ASSEMBLER CODE,
10                ;       AND AS A RESULT TAKE UP NO MEMORY SPACE.
11                ;       LABEL EQUATES SIMPLY ASSIGN A VALUE TO THE LABEL.
12                ;       UPON ENCOUNTERING EACH LABEL, THE MACRO-ASSEMBLER
13                ;       REPLACES IT WITH THAT VALUE.
14        000000 AC0=0            ;THESE 4 LINES
15        000001 AC1=1            ;ARE USEFUL FOR GENERATING
16        000002 AC2=2            ;MEANINGFUL NAMES FOR
17        000003 AC3=3            ;THE ACCUMULATORS IN A.L.C INSTS.
18
19                ; NOTE: THE FOLLOWING LINE IS ILLUSTRATION OF THE .BLK STATEMENT
20                ;       .BLK IS USED TO ALLOCATE (OR DEFINE) AN AREA OF
21                ;       STORAGE, BUT DOES NOT ASSIGN ANY SPECIFIC VALUE.
22                ;       CONTRASTED WITH THIS IS THE .TXT STATEMENT WHICH ASSIGNS
23                ;       A SPECIFIC VALUE TO THE BLOCK OF MEMORY RESERVED.
24                ;       THE .BLK & .TXT ARE SYNONYMOUS WITH 'DS' AND 'DC'
25                ;       STATEMENTS RESPECTIVELY IN OTHER ASSEMBLER LANGUAGES.
26
27 00000'000065           .BLK    65
28
   00065'067516           .TXT    _NOTICE THE DIFFERENCE IN MACHINE CODE GENERATED
         064564
         062543
32       072040
33       062550
34       062040
35       063151
36       062546
37       062562
38       061556
39       020145
40       067151
41       066440
42       061541
43       064550
44       062556
45       061440
46       062157
47       020145
48       062547
49       062556
50       060562
51       062564
52       000144
53 00115'062542           .TXT    _BETWEEN .BLK & .TXT._
         073564
         062545
         020156
57       041056
58       045514
59       023040
60       027040
```

```
 0003 HANDO
01        054124
02        027124
          000000
     0130'062524          .TXT    _TEXT STMTS REQUIRE BEGINING & ENDING DELIMITERS
     0160'020111          .TXT    _I CHOSE THE UNDERSCORE BECAUSE IT'S RARELY USED
06        000000          .NOLOC  0       ;RESTORE OBJECT CODE LISTING CAPABILITY
07  00210'072111          .TXT    _IT'S EASY TO  FORGET THE ENDING DELIMITER.
08        071447
09        062440
10        071541
11        020171
12        067564
13        020040
14        067546
15        063562
16        072145
17        072040
18        062550
19        062440
20        062156
21        067151
22        020147
23        062544
24        064554
25        064555
26        062564
27        027162
2A  00235'027011          .TXT      IN WHICH CASE, THE REMAINDER OF THE SOURCE TEXT
          054124
          004524
3.        064440
32        020156
33        064167
34        061551
35        020150
36        060543
37        062563
38        020054
39        064164
40        020145
41        062562
42        060555
43        067151
44        062544
45        020162
46        063157
47        072040
48        062550
49        071440
50        072557
51        061562
52        020145
53        062564
5-        072170
     0270'027011          .TXT      IS TAKEN AS PART OF THE TEXT DATA_
          054124
57        004524
58        064440
59        020163
60        060564
```

```
 0004 HANDO
01        062553
02        020156
          071541
          070040
          071141
06        020164
07        063157
08        072040
09        062550
10        072040
11        074145
12        020164
13        060544
14        060564
15        000000
16        000000 .NOLOC  0        ;THIS RESTORES MACHINE CODE LISTING
17
18                ; NOTE: THE FOLLOWING LINE ILLUSTRATES USE OF A LABEL TO DEFINE
19                ;       WORD OF STORAGE.  THE LABEL MUST BE GIVEN A VALUE; OTHER
20                ;       ACTUAL STORAGE SPACE IS GENERATED.
21  00315'000000 ADER1:0
22  00316'000001          1                       ;THIS IS AT ADER1+1
23  00317'000002          2                       ;THIS IS AT ADER1+2
24  00320'000016          16                      ;THIS IS AT ADER1+3
25                              ;NOTE THE DEFAULT RADIX IS OCTAL
26  00321'000020          16.                     ;THIS DEFINES DECIMAL 16 AT ADER
27                ;               ^<====== NOTE THE DECIMAL POINT DENOTING DECIMAL
28        000012          .RDX    10              ;THE DEFAULT RADIX CAN ALSO BE C
                                                  ;AS IN THIS EXAMPLE.
          00322'000020          16                ;NOW NOTE THE VALUE.
          000010          .RDX    8               ;WHEN I CHANGE IT BACK TO 8
32  00323'000016          16                      ;STORAGE DEFINITIONS ARE IN OCTA
33
34                ; NOTE: THE FOLLOWING LINES ILLUSTRATE THE USE OF THE
35                ;       .TXTM AND .TXTN STATEMENTS.
36                ;       .TXTM MODIFIES THE DEFAULT PACKING OF TEXT DATA
37                ;       IN .TXT STATEMENTS.
38                ;       .TXTN SUPPRESSES THE WORD OF NULLS FOLLOWING AN
39                ;       EVEN NUMBER OF; TEXT CHARACTERS.
40                ;       FOR RDOS, DEFAULT PACKING IS R-L. FOR AOS, IT IS L-R.
41        000001          .TXTM   1               ;THIS SETS TEXT PACKING
42                                                ;TO THE OPPOSITE OF THE DEFAULT.
43  00324'040563          .TXT    _AS THIS HANDOUT WAS DONE ON AN RDOS SYSTEM,_
44        020164
45        064151
46        071440
47        064141
48        067144
49        067565
50        072040
51        073541
52        071440
53        062157
          067145
          020157
          067040
          060556
58        020122
59        042117
60        051440
```

```
 0005 HANDO
01          071571
02          071564
            062555
            026000
00352'027124            .TXT    _.TXTM 1 SETS PACKING L-R._
06          054124
07          046440
08          030440
09          071545
10          072163
11          020160
12          060543
13          065551
14          067147
15          020114
16          026522
17          027000
18          000000      .TXTM   0       ;NOW WHEN I REVERT BACK TO THE DEFAULT..
19 00367'062527         .TXT    _WE THEN GET PACKING R-L_
20          072040
21          062550
22          020156
23          062547
24          020164
25          060560
26          065543
27          067151
28          020147
            026522
            000114
31 00403'062524         .TXT    _TEXT STREAMS_  ;NOTICE THE NULL WORD?
32          072170
33          051440
34          071164
35          060545
36          071555
37          000000
38          000001      .TXTN   1               ;NOW WATCH IT DISAPPEAR.
39 00412'062524         .TXT    _TEXT STREAMS_  ;SEE?  AIN'T THAT SLICK?
40          072170
41          051440
42          071164
43          060545
44          071555
45          000000      .TXTN   0               ;AND NOW.......
46                                              ;BE-4 YOU CAN SAY "TEXT STATEMEN
47
48 00420'062524         .TXT    _TEXT STREAMS_  ;THE NULL WORD MAGICALLY APPEARS
49          072170
50          051440
51          071164
52          060545
53          071555
            000000

00427'067502 TEXT:      .TXT    _BOTH .TXT & .BLK CAN HAVE LABELS_
57 00450'000002 BLAWK:  .BLK    2               ;HOW'S THAT FOR BEING SNEAKY.
58          000000      .NOLOC  0       ;RENABLE THE B.S.
59
60                      .EJEC
```

```
 0006 HANDO
01                ;                  ***** LOCATION COUNTER *****
02
                  ;          THE LOCATION COUNTER IS AN ADDRESS POINTER USED DURING
                  ;              ASSEMBLY PROCESS TO KEEP TRACK OF WHERE THINGS
                  ;              THE PROGRAM SHOULD GO.  THIS COUNTER IS SYNONYM
06                ;              (AT RUN TIME) WITH THE P.C. (PROGRAM COUNTER).
07                ;              IT CAN BE SET BACK TO A PREVIOUS ADDRESS FOR
08                ;              REDEFINITION OF STORAGE.  IT CAN ALSO BE SAVED
09                ;              FOR PURPOSES OF RESTORATION BEFORE TEMPORARILY
10                ;              ALTERING ITS VALUE
11                ;              FOR EXAMPLE:
12        000452'                .PUSH    .LOC              ;SAVES THE CURRENT L.C.
13        000317'                .LOC     ADER1+2
14 00317'000002 ADER3:           .BLK     2
15        000452'.A=             .POP                ;OBTAIN THE SAVED LOCATION COUNTER
16        000452'                .LOC     .A         ;RESTORE THE LOCATION COUNTER
17 00452'000000 ADDR1:           0
18
19
20                ;          ********** NOVA INSTRUCTION SET **********
21                ;                              M.R.I.
22
23                ;          LDA                  LOAD ACCUMULATOR
24                ;          STA                  STORE ACCUMULATOR
25
26                ;          ISZ                  INCREMENT - SKIP IF ZERO
27                ;          DSZ                  DECREMENT - SKIP IF ZERO
28
                  ;          JMP                  JUMP
                  ;          JSR                  JUMP TO SUBROUTINE
                  ;                                    OR
32                ;                               JUMP & SAVE RETURN
```

```
!0007 HANDO
01                ;                  ********** NOVA INSTRUCTION SET **********
02                ;                                  A.L.C.

                  ;        ADD                  ADD   TWO ACCUMULATORS
                  ;        SUB               SUBTRACT TWO ACCUMULATORS
06
07                ;        COM               COMPUTE 1'S COMPLAMENT OF ACCUMULATOR
08                ;        NEG               COMPUTE 2'S COMPLAMENT OF ACCUMULATOR (1
09                ;        ADC               ADD THE COMPLAMENT OF AN ACCUMULATOR
10
11                ;        INC               ADD 1 (INCREMENT) TO ACCUMULATOR
12
13                ;        MOV               MOVE ONE ACCUMULATOR TO ANOTHER
14
15                ;        AND               AND TWO ACCUMULATORS TOGETHER
```

```
!0008 HANDO
01                ;                          LDA
02                ;                LOAD      ACCUMULATOR
                  ;                - -       -
                  ;                ----------------------

06                ;       LOADS THE CONTENTS OF A SPECIFIED MEMORY LOCATION
07                ;               INTO THE SPECIFIED ACCUMULATOR
08
09                ;       POSSIBLE ACCUMULATORS:  1 - AC1
10                ;                               2 - AC2
11                ;                               3 - AC3
12                ;                               0 - AC0
13
14                ;       FORMAT:
15                ;                LDA     ACS,ADDR
16                ;                    OR
17                ;                LDA     ACS,D,X
18
19                ;       WHERE:   ACS     IS THE ACCUMULATOR
20                ;                ADDR    IS THE ADDRESS LABEL
21                ;                D       IS THE DISPLACEMENT (0-377 OR -200 -- +1
22                ;                X       IS THE INDEX MODE (0,1,2,3)
23
24                ;       EXAMPLES:
25
26 00453'020777          LDA     AC0,ADDR1       ;LOAD  CONTENTS OF ADDR1 INTO AC
27
28 00454'024402          LDA     AC1,2,1         ;LOAD  CONTENTS OF PC+2

   00455'031005          LDA     AC2,5,2         ;LOAD AC2 W/ CONTENTS OF AC2

32 00456'025410          LDA     AC1,10,3        ;LOAD AC1 W/ CONTENTS OF AC3+10
33
34 00457'022773          LDA@    AC0,ADDR1       ;LOAD AC0 THRU ADDR1
```

```
!0009 HANDO
01                ;                                   STA
02                ;                             STORE   ACCUMULATOR
                  ;                             --      -
                  ;                             ----------------
06                ;       STORE THE CONTENTS OF THE SPECIFIED ACCUMULATOR INTO
07                ;               INTO THE INDICATED ADDRESS LOCATION
08
09                ;       POSSIBLE ACCUMULATORS:  1 - AC1
10                ;                               2 - AC2
11                ;                               3 - AC3
12                ;                               0 - AC0
13
14                ;       FORMAT:
15                ;               STA     ACS,ADDR
16                ;               STA     ACS,D,X
17                ;
18                ;       WHERE:
19                ;               ACS     IS THE ACCUMULATOR
20                ;               ADDR    IS THE ADDRESS LABEL
21                ;               D       IS THE DISPLACEMENT  (0-377 OR -200 -- +
22                ;               X       IS THE INDEX MODE  (0,1,2,3)
23
24                ;       EXAMPLES:
25
26 00460'040772           STA     AC0,ADDR1       ;STORE AC0 IN ADDR1
27
28 00461'044405           STA     AC1,5,1         ;STORE AC1 @ 5+PC  dangerous
   00462'051010           STA     AC2,10,2        ;STORE AC2 @ 10+AC2
32 00463'045415           STA     AC1,15,3        ;STORE AC1 @ 15+AC3
33
34 00464'042766           STA@    AC0,ADDR1       ;STORE AC0 THRU ADDR1
```

100 AC1 : Ø
101 AC0 : @AC1
↑
100

```
!0010 HANDO
01                ;                              ISZ
02                ;                    INCREMENT    SKIP IF ZERO
                  ;                       -            -       -
                  ;                    ------------------------
06                ;       INCREMENT THE CONTENTS OF A MEMORY LOCATION BY 1
07                ;               AND SKIP THE NEXT INSTRUCTION IF
08                ;                   THE LOCATION BECOMES ZERO.
09
10                ;       FORMAT:
11                ;                ISZ      ADDR
12                ;                     OR
13                ;                ISZ      D,X
14                ;
15                ;       WHERE:
16                ;                ADDR     IS THE ADDRESS LABEL
17                ;                D        IS THE DISPLACEMENT (0-377 OR -200 -- +1
18                ;                X        IS THE INDEX MODE    (0,1,2,3)
19
20                ;       EXAMPLES:
21
22  00465'010765          ISZ      ADDR1            ;INCREMENT ADDR1
23
24  00466'010405          ISZ      5,1              ;INCREMENT 5+PC
25                                                   MODE
26  00467'011010          ISZ      10,2             ;INCREMENT 10+AC2
27
28  00470'011415          ISZ      15,3             ;INCREMENT 15+AC3

?   00471'012761          ISZ@     ADDR1            ;INCREMENT THRU ADDR1
```

```
!0011 HANDO
01                ;                                DSZ
02                ;                        DECREMENT   SKIP IF ZERO
                  ;                           -           -       -
                  ;                        ------------------------
06                ;       DECREMENT THE CONTENTS OF THE SPECIFIED MEMORY LOCATION
07                ;               AND SKIP THE NEXT INSTRUCTION IF THE
08                ;                    LOCATION BECOMES ZERO.
09
10                ;       FORMAT:
11                ;               DS7     ADDR
12                ;                    OR
13                ;               DSZ     D,X
14
15                ;       WHERE:
16                ;               ADDR    IS THE ADDRESS LABEL
17                ;               D       IS THE DISPLACEMENT  (0-377 OR -200 -- +
18                ;               M       IS THE INDEX MODE    (0,1,2,3)
19
20                ;       EXAMPLES:
21
22 00472'014760           DSZ     ADDR1           ;DECREMENT ADDR1
23
24 00473'014405           DSZ     5,1             ;DECREMENT PC+5
25
26 00474'015010           DSZ     10,2            ;DECREMENT 10+AC2
27
2A 00475'015415           DS7     15,3            ;DECREMENT 15+AC3

   00476'016754           DSZ@    ADDR1           ;DECREMENT THRU ADDR1
```

```
01                  ;                                   JMP
02                  ;                            JUMP   (SYNONYMOUS WITH GO TO)
                    ;
                    ;
                    ;       JUMP TO SPECIFIED MEMORY LOCATION. KNOWN AS A BRANC
06                  ;              OTHER SYSTEMS.  SYNONOMOUS WITH  GO TO
07                  ;                  USED IN BASIC, FORTRAN & COBOL.
08
09                  ;       FORMAT:
10                  ;                JMP      ADDR
11                  ;                     OR
12                  ;                JMP      D,X
13
14                  ;       WHERE:
15                  ;                ADDR     IS THE ADDPESS LABEL
16                  ;                D        IS THE DISPLACEMENT (0-377 OR -200 -- +1
17                  ;                X        IS THE INDEX MODE   (0,1,2,3)
18
19                  ;       EXAMPLES:
20
21  00477'000753            JMP      ADDR1              ;JUMP TO ADDR1
22
23  00500'000405            JMP      5,1                ;JUMP TO PC+5
24
25  00501'001010            JMP      10,2               ;JUMP TO 10+AC2
26
27  00502'001415            JMP      15,3               ;JUMP TO 15+AC3
28
    00503'002747            JMP@     ADDR1              ;JUMP THRU ADDR1
```

JMP @ADDR1

```
!0013 HANDO
01                ;                                  JSR
02                ;                            JUMP   TO  SUBROUTINE
                  ;                            -     OR    -  -
                  ;                            JUMP    AND SAVE RETURN
                  ;                            -           -    -
06                ;                            ----------------------
07
08                ;       JUMP TO SPECIFIED MEMORY ADDRESS SAVING THE RETURN ADDRE
09                ;            (ADDR OF N.S.I.) IN AC3.
10
11                ;       FORMAT:
12                ;                JSR     ADDR
13                ;                     OR
14                ;                JSR     D,X
15
16                ;       WHERE:
17                ;                ADDR    IS THE ADDRESS LABEL
18                ;                D       IS THE DISPLACEMENT    (0-377 OR -200 --
19                ;                X       IS THE INDEX MODE      (0,1,2,3)
20
21                ;       EXAMPLES:
22
23 00504'004746          JSR     ADDR1              ;JUMP TO ADDR1
24
25 00505'004405          JSR     5,1 or .+5         ;JUMP TO PC+5
26
27 00506'005010          JSR     10,2               ;JUMP TO 10+AC2
2R
   00507'005415          JSR     15,3               ;JUMP TO 15+AC3

3. 00510'006742          JSR@    ADDR1              ;JUMP THRU ADDR1
```

```
!0014 HANDO
01                ;                                  ADD
02                ;                           ADD    ACCUMULATORS
                  ;                           ---
                  ;                           ----------------------

06                ;       ADD THE CONTENTS OF ACS TO THE CONTENTS OF ACD - THE
07                ;               RESULT IS PLACED IN ACD.
08                ;               ACS REMAINS UNCHANGED.
09
10                ;       FORMAT:
11                ;               ADD     ACS,ACD
12
13                ;       EXAMPLES:
14
15 00511'133000           ADD     AC1,AC2          ;ADD AC1 TO AC2
16
17 00512'157120           ADDZL   AC2,AC3          ;ADD AC2 TO AC3 & LEFT SHIFT
18                                                 ;NOTE: CARRY SET TO 0 FIRST
19
20 00513'117223           ADDZR   AC0,AC3 SNC      ;ADD AC0 TO AC3 & RIGHT SHIFT
21                                                 ;SKIP IF NON-ZERO CARRY
22
23 00514'113002           ADD     AC0,AC2 SZC      ;ADD AC0 TO AC2 SKIPPING NSI
24                                                 ;IF OVERFLOW OCCURS
```

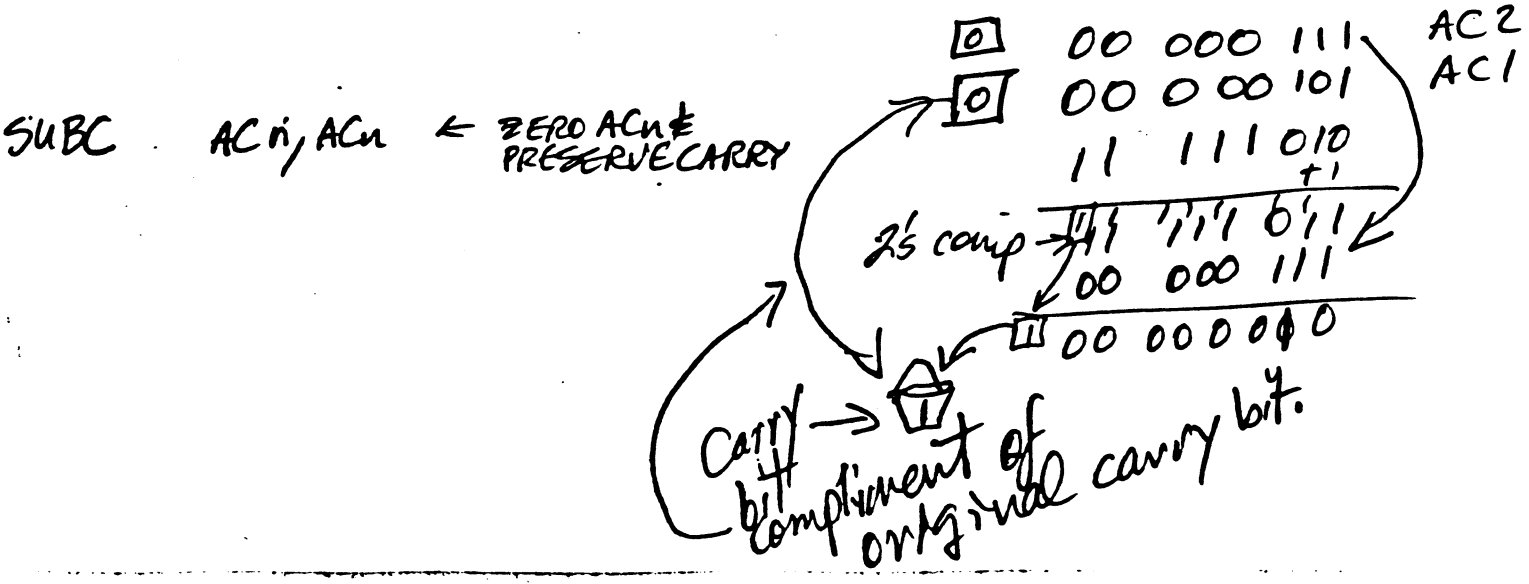
compliment carry on overflow.

```
10015 HANDO
01                ;                              SUB
02                ;                    SUBTRACT   REGISTERS
                  ;                    ---
                  ;                    ------------------
06                ;       SUBTRACT THE CONTENTS OF ACS FROM ACD LEAVING THE
07                ;              RESULT IN ACD. ACS REMAINS UNCHANGED.
08                ;              IF ACS IS < OR = ACD, CARRY WILL BE
09                ;                   COMPLEMENTED.
10
11                ;       FORMAT:
12                ;                SUB      ACS,ACD
13
14                ;       EXAMPLES:
15
16 00515'106400          SUB      AC0,AC1          ;SUBTRACT AC0 FROM AC1
17
18 00516'152540          SUBOL    AC2,AC2          ;SUBTRACT AC2 FROM IT SELF
19                                                 ;(ZERO IT OUT)
20                                                 ;WHAT WOULD HAPPEN IF CARRY WERE
21                                                 ;SET TO 1 FIRST?
22
23 00517'176520          SUBZL    AC3,AC3          ;SUBTRACT AC3 FROM ITSELF -
24                                                 ;SHIFT COMPLEMENTED CARRY
25                                                 ;INTO BIT 15 (CST-1)
26
27 00520'172422          SUBZ     AC3,AC2 SZC      ;SUBTRACT AC3 FROM AC2 -
28                                                 ;SKIP IF AC3>AC2
```

SUBTRACT

ACD = 7 = AC2
ACS = 5 = AC1

SUBZ AC1, AC2



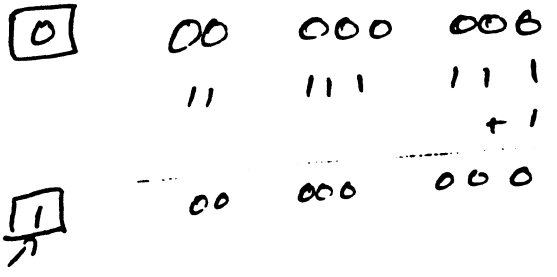
```
!0016 HANDO
01                  ;                                   COM
02.                 ;                    COMPUTE   1'S COMPLAMENT
                    ;                                   ---
                    ;                    ------------------------

06                  ;       COMPUTE THE 1'S COMPLIMENT OF ACS AND PLACE THE RESULTAN
07                  ;               VALUE IN ACD.  ACS REMAINS UNCHANGED.
08
09                  ;       FORMAT:
10                  ;               COM     ACS,ACD
11
12                  ;       EXAMPLES:
13
14 00521'130000             COM     AC1,AC2         ;PLACE 1'S COMP OF AC1 IN AC2
15
16 00522'100200             COMR    AC0,AC0         ;PLACE 1'S COMP OF AC0 IN
17                                                  ;ITSELF AND SHIFT RIGHT
```

```
!0017 HANDO
01                ;                                   NEG
02                ;                             NEGATE    ACCUMULATOR
                  ;                             ---
                  ;                             ____________________

06                ;       NEGATE THE CONTENTS OF ACS PLACING IT IN ACD.  IF
07                ;              A BIT IS CARRIED OUT OF THE HIGH ORDER
08                ;              THE CARRY BIT IS COMPLIMENTED.
09                ;       (NOTICE THE SIMILARITY TO COM)
10
11                ;       FORMAT:
12                ;              NEG      ACS,ACD
13
14                ;       EXAMPLES:
15
16 00523'100400           NEG      AC0,AC0          ;NEGATE AC0
17
18 00524'130520           NEGZL    AC1,AC2          ;ZERO CARRY - NEGATE AC1 INTO
19                                                  ;AC2 SHIFTING LEFT FIRST
```

Negate 0



```
!0018 HANDO
01                ;                              INC
02                ;                    INCREMENT   ACCUMULATOR
                  ;                    ---
                  ;                    ----------------------

06                ;       INCREMENT THE CONTENTS OF ACS (BY 1) AND PLACE THE RESUL
07                ;               IN ACD.  ACS REMAINS UNAFFECTED.  IF OVERFLOW
08                ;               OCCURS, CARRY IS COMPLEMENTED.
09
10                ;       FORMAT:
11                ;               INC     ACS,ACD
12
13                ;       EXAMPLES:
14
15 00525'101400           INC     AC0,AC0         ;INCREMENT AC0
16
17 00526'131422           INCZ    AC1,AC2 SZC     ;INCREMENT AC1 INTO AC2 -
18                                                ;SKIP NSI IF CARRY NOT SET
19
20 00527'175433           INCZ#   AC3,AC3 SNC     ;INCREMENT (DON'T ACTUALLY
21                                                ;MODIFY)AC3 - SKIP IF CARRY
22                                                ;IS NON-ZERO.
```

```
10019 HANDO
01                ;                                  MOV
02                ;                               MOVE   ACCUMULATOR
                  ;                               ---
                  ;                               ------------------

06                ;       MOVE THE CONTENTS OF ACS INTO ACD.
07                ;              THIS IS ONLY A COPY FUNCTION -
08                ;              THE CONTENTS OF ACS REMAIN UNCHANGED.
09
10                ;       FORMAT:
11                ;              MOV      ACS,ACD
12
13                ;       EXAMPLES:
14
15 00530'101003           MOV      AC0,AC0 SNC      ;MOVE AC0 INTO ITSELF TO
16                                                  ;TO TEST CARRY.
17
18 00531'131120           MOVZL    AC1,AC2          ;MOVE AC1 TO AC2 AND
19                                                  ;MULTIPLY BY 2
20
21 00532'155220           MOVZR    AC2,AC3          ;MOVE AC2 TO AC3 AND
22                                                  ;DIVIDE BY 2
```

```
!0020 HANDO
01                ;                                  ADC
02                ;                               ADD   COMPLAMENT
                  ;                               --    -
                  ;                               ----------------

06                ;       ADD THE COMPLIMENT OF ACS TU ACD LEAVING THE RESULT
07                ;               IN ACD.  ACS REMAINS UNCHANGED.
08
09                ;       FORMAT:
10                ;               ADC     ACS,ACD
11
12                ;       EXAMPLES:
13
14 00533'106000           ADC     AC0,AC1         ;ADD THE COMPL. OF AC0 TO AC1
15
16 00534'152140           ADCOL   AC2,AC2         ;ADD THE COMPL. OF AC2 TO ITSELF
17                                                ;SETTING CARRY TO 1 - THEN
18                                                ; LEFT SHIFT GENERATING CST-1
```

```
10021 HANDO
01                ;                                   AND
                  ;                          LOGICAL  AND TWO ACCUMULATORS
                  ;                                   ---
                  ;
05                ;       LOGICALLY AND ACS AGAINST ACD LEAVING THE RESULT
06                ;               IN ACD.  ACS REMAINS UNAFFECTED.
07
08                ;       FORMAT:
09                ;               AND     ACS,ACD
10
11                ;       EXAMPLES:
12
13 00535'107400           AND     AC0,AC1          ;AND AC0 AGAINST AC1
14
15 00536'153620           ANDZR   AC2,AC2          ;AND AC2 AGAINST ITSELF
16                                                 ;ZEROING CARRY FIRST
17                                                 ;AND THEN SHIFT RIGHT.
18
19 00537'167503           ANDL    AC3,AC1 SNC      ;AND AC3 AGAINST AC1 AND
20                                                 ;SKIP IF CARRY IS SET AFTER
21                                                 ;DOING A LEFT SHIFT
```

```
        LICENSED MATERIAL - PROPERTY OF DATA GENERAL CORPORATION
!0022 HANDO
01                ;       THE FOLLOWING EXAMPLE ILLUSTRATES A NUMBER OF DIFFERENT
02                ;               INVERSE MOVE A TABLE USING BASE DISPLACEMENT ADD
                  ;               INDIRECT ADDRESSING AND AUTO INDEX LOCATIONS.

                  ;       ********** BASE DISPLACEMENT ADDRESSING **********
06
07 00540'030435           LDA     AC2,INTAB       ;GET ADDRESS OF INPUT TABLE
08 00541'034435           LDA     AC3,OUTAB       ;GET ADDRESS OF OUTPUT TABLE
09 00542'126000           ADC     AC1,AC1         ;GENERATE -1 FOR DECREMENT
10                LOOP1:                                          .
11 00543'021000           LDA     AC0,ZERO,AC2    ;GET THE INPUT TABLE VALUE
12 00544'041400           STA     AC0,ZERO,AC3    ;STORE REVERSE IN OUTPUT TABLE
13 00545'151400           INC     AC2,AC2         ;BUMP  TO NEXT INPUT  TABLE ADDR
14 00546'137000           ADD     AC1,AC3         ;DECR. TO NEXT OUTPUT TABLE ADDR
15 00547'014513           DSZ     MVCNT           ;ALL DONE ?
16 00550'000773           JMP     LOOP1           ;REPEAT TIL TABLE MOVED
17
18
19                ;       ********** INDIRECT ADDRESSING **********
20
21 00551'024424           LDA     AC1,INTAB       ;GET ADDRESS OF INPUT  TABLE
22 00552'044421           STA     AC1,WORK1       ;STORE IN INDIRECT WORK AREA
23 00553'024423           LDA     AC1,OUTAB       ;GET ADDRESS OF OUTPUT TABLE
24 00554'044420           STA     AC1,WORK2       ;STORE IN INDIRECT WORK AREA
25                LOOP2:
26 00555'022416           LDA@    AC0,WORK1       ;GET INPUT  TABLE VALUE THRU WOR
27 00556'042416           STA@    AC0,WORK2       ;GET OUTPUT TABLE VALUE THRU WOR
28 00557'010414           ISZ     WORK1           ;BUMP INPUT  ADDRESS
   00560'010414           DSZ     WORK2           ;BUMP OUTPUT ADDRESS
   00561'014501           DSZ     MVCNT           ;ALL DONE ?
   00562'000773           JMP     LOOP2           ;REPEAT TIL TABLE MOVED
32
33
34                ;       ********** AUTO INDEX  ADDRESSING **********
35
36 00563'024414           LDA     AC1,INTBA       ;GET INPUT TABLE ADDRESS -1
37 00564'044020           STA     AC1,INPUT       ;STORE IN AUTO INDEX LOCATION
38 00565'024413           LDA     AC1,OUTBA       ;PUT OUTPUT TABLE ADDRESS -1
39 00566'044030           STA     AC1,OUTPT       ;STORE IN AUTO INDEX LOCATION
40                LOOP3:
41 00567'022020           LDA@    AC0,INPUT       ;GET INPUT  TABLE VALUE
42 00570'042030           STA@    AC0,OUTPT       ;PUT OUTPUT TABLE VALUE
43 00571'014471           DSZ     MVCNT           ;ALL DONE ?
44 00572'000775           JMP     LOOP3           ;REPEAT TIL TABLE MOVED
45
46
47
48 00573'000000 WORK1:    0                       ;INDIRECT WORK AREA
49 00574'000000 WORK2:    0                       ;    "     ""   ""
50
51       000000 ZERO=     0
52       000020 INPUT=    20                      ;AUTO INDEX LOC. 20
53       000030 OUTPT=    30                      ;AUTO INDEX LOC. 30

   00575'000601'INTAB:    INPTB                   ;ADDRESS OF INPUT  TABLE
   00576'000634'OUTAB:    OUTTB+                  ;ADDRESS OF OUTPUT TABLE
57 00577'000600'INTBA:    INPTB-1                 ;ADDRESS OF INPUT  TABLE -1
58 00600'000635'OUTBA:    OUTTB+1                 ;ADDRESS OF OUTPUT TABLE +1
59
60 00601'040501 INPTB:    .TXT    ._AABBCCDDEEFFGGHHIIJJKKLLMMNNOOPPQQRRSSTTUUVVWWX
```

```
 0023 HANDO
01          041102
02          041503
            042104
            042505
            043106
06          043507
07          044110
08          044511
09          045112
10          045513
11          046114
12          046515
13          047116
14          047517
15          050120
16          050521
17          051122
18          051523
19          052124
20          052525
21          053126
22          053527
23          054130
24          054531
25          055132
26          000000
27 00634'000026 OUTTB:  .BLK    26
28
   00662'000033 MVCNT:  OUTTB-INPTB                ;LENGTH OF MOVE
```

```
10024 HANDO
01                ;                    *** RELOCATABILITY DOCUMENTATION ***
02
                  ;CONSIDER THE FOLLOWING DESIRABLE CHARACTERISTICS OF PROGRAM
                  ;DEVELOPMENT:

06                ;1. SUBROUTINES CREATED FOR PREVIOUS PROGRAMS SHOULD NOT HAVE
07                ;   TO BE RE-WRITTEN TO ACCOMODATE A NEW PROGRAM ENVIRONMENT, AN
08
09                ;2. DIFFERENT PROGRAMMERS SHOULD BE ABLE TO WORK INDEPENDENTLY
10                ;   ON SEPARATE MODULES WITHOUT FEAR OF DUPLICATION OF USER SYMBO
11
12                ;CHARACTERISTIC (2) IS EASILY ACCOMPLISHED BY WRITING SUBROUTINE
13                ;AS INDIVIDUAL DISK FILES WHICH CAN BE ASSEMBLED SEPARATELY. TO
14                ;OBTAIN AN EXECUTABLE FILE WOULD ONLY REQUIRE THE EXTRA STEP OF
15                ;COMBINING THE ASSEMBLED VERSIONS SOMEHOW (E.G. THROUGH THE USE
16                ;AN ABSOLUTE LOADER).
17
18                ;HOWEVER A DISADVANTAGE OF THIS SCHEME IS THAT THE POSITION DEP-
19                ;ENDENCE OF THE .LOC PSEUDO OP REQUIRES AGREEMENT AMONG THE PROG
20                ;RAMMERS AS TO THE ABSOLUTE LOCATION OF EACH MODULE SO THAT MODU
21                ;INTERCOMMUNICATION REQUIREMENTS CAN BE SATISFIED. FURTHERMORE,
22                ;IF ONE OF THE MODULES IS TO BE REUSABLE AS PER (1), THE ABSOLUT
23                ;NATURE OF THE MODULE MUST BE TAKEN INTO CONSIDERATION AND THE N
24                ;MODULES MUST LITERALLY BE "WRITTEN AROUND" THE OLD ONE.
25
26                ;IN ORDER TO ACHIEVE THE OBJECTIVES IN (1) AND (2) WITHOUT THE
27                ;INHERENT DIFFICULTIES IMMEDIATELY IMPOSED BY THE .LOC PSEUDO OP
28                ;WE SCRAP .LOC ENTIRELY AND INTRODUCE THE CONCEPT OF RELOCATABIL

                  ;RELOCATABLE MODULES:

32                ;       1. ARE ASSEMBLED FROM INDIVIDUAL SOURCE FILES
33                ;       2. ARE IN MACHINE LANGUAGE FORMAT AFTER THE ASSEMBLY,
34                ;          BUT ABSOLUTE ADDRESSES ARE NOT ASSIGNED INSTEAD,
35                ;          EACH ASSEMBLED WORD IS GIVEN A RELATIVE LOCATION
36                ;          IN THE MODULE WITH RESPECT TO THE FIRST WORD
37                ;       3. REQUIRE THE CONSTRUCTION OF ONE EXECUTABLE FILE
38                ;          FROM SEVERAL DISTINCT ASSEMBLED SOURCE FILES
39                ;          THROUGH THE USE OF A BINDER CALLED THE "RELOCATABLE
40                ;          LOADER".
41                ;       4. OVERCOME THE MODULE INTERCOMMUNICATION PROBLEM THROUG
42                ;          THE USE OF "GLOBAL SYMBOLS".
```

```
10025 MANDO
01                ;               ******** EXAMPLES OF CLI TO INVOKE ASM ********
02
                  ;       ASSEMBLE SOURCE PROGRAM FOR SYNTAX ONLY - NO .RB FILE NE
05                ;               XREF LISTING REQUESTED - LIST FILE IS THE SOURCE
06                ;               FILE NAME W/ .LS EXTENSION.
07
08                ;       ASM/X/L/N   SOURCE
09
10
11                ;       ASSEMBLE SOURCE FILE W/ LISTING TO LINE PRINTER, INCLUDE
12                ;               DEFINED SYMBOLS, XREF LISTING, GIVE THE .RB FILE
13                ;               A NAME DIFFERENT THAN THAT OF SOURCE.
14
15                ;       ASM/X/U   SOURCE OBJECT.RB/B $LPT/L
16
17
18                ;       ASSEMBLE SOURCE FILE.  INCLUDE FILE W/ SYMBOL DEFINITION
19                ;               PASS 1 ONLY.    LIST FILE IS SAME AS SOURCE FILE
20                ;               WITH THE .LS EXTENSION - INCLUDE XREF
21
22                ;       ASM/X/L SYMBOLS/S SOURCE
23                        .EJECT
```

```
 0026 HANDO
01
02                ;           ******** EXAMPLES OF CLI TO INVOKE MAC ********
                  ;       ASSEMBLE SOURCE FOR SYNTAX ONLY - NO .RB FILE NEEDED
06                ;               LIST FILE IS THE SOURCE FILE NAME W/ THE
07                ;               .LS EXTENSION.
08
09                ;       MAC/L/N SOURCE
10
11
12                ;       ASSEMBLE SOURCE FILE W/ LISTING TO LINE PRINTER, INCLUDE
13                ;               DEFINED SYMBOLS & GIVE THE .RB FILE A NAME
14                ;               DIFFERENT THAN THAT OF SOURCE.
15
16                ;       MAC/U   SOURCE OBJECT.RB/B $LPT/L
17
18
19                ;       ASSEMBLE SOURCE FILE.   INCLUDE FILE W/ SYMBOL DEFINITIO
20                ;               PASS1 ONLY.     LIST FILE IS SAME AS SOURCE FILE
21                ;               WITH THE .LS EXTENSION.
22
23                ;       MAC/L SYMBOL/S SOURCE
24                        .EJECT
```

```
01
02                ;           ******** EXAMPLES OF CL. TO INVOKE MASM ********


                  ;       ASSEMBLE SOURCE FOR SYNTAX ONLY - NO .OB FILE NEEDED
06                ;               LIST FILE IS THE SOURCE FILE NAME W/ THE
07                ;               .LS EXTENSION.
08
09                ;       X MASM/N/L=SOURCE.LS    SOURCE
10
11
12                ;       ASSEMBLE SOURCE FILE W/ LISTING TO LINE PRINTER, INCLUDE
13                ;               DEFINED SYMBOLS & GIVE THE .OB FILE A NAME
14                ;               DIFFERENT THAN THAT OF SOURCE.
15
16                ;       X MASM/U/L=@LPT SOURCE OBJECT.OB/B
17
18
19                ;       ASSEMBLE SOURCE FILE.   INCLUDE FILE W/ SYMBOL DEFINITIO
20                ;               PASS1 ONLY.     LIST FILE IS SAME AS SOURCE FILE
21                ;               WITH THE .LS EXTENSION.
22
23                ;       X MASM/L=SOURCE.LS SYMBOL/S SOURCE
24
25                        .EJECT
```

```
 0028 HANDO
01                ;                ******** EXAMPLES OF CLI TO INVOKE RLDR ********
02
.                 ;        RELOCATABLE LOAD A SINGLE TASK PROGRAM, INCLUDE THE     J
                  ;                REQUEST A LOAD MAP AND APPEND LISTING TO A PREVI
06                ;                .LS FILE FROM AN ASSEMBLY, THE .SV FILE NAME IS
FUU00663'000000                    THE SAME AS THAT OF THE.RB FILE.
08
09                ;        RLDR/D/P  PROG1.<RB,LS/L>
10
11                ;        RELOCATABLE LOAD A SINGLE TASK PROGRAM WITH OVERLAYS, CH
12                ;                THE .SV & .OL FILES TO ANOTHER NAME.  NO LOAD MA
13                ;                IS BEING PRODUCED.
14
15                ;        RLDR     PROG1  NEWPROG/S
16
17                ;        RELOCATABLE LOAD A MAIN PROGRAM AND 4 TASKS.  THIS IS A
18                ;                MULTI-TASKING PROGRAM REQUIRING 22(OCTAL) TASKS,
19                ;                16(OCTAL) CHANNELS, AND DEFINES A VIRTUAL
20                ;                OVERLAY AREA, REQUEST A LOAD MAP APPENDED TO
21                ;                THE MAIN PROGRAM'S LISTING FILE.
22
23
24                ;        RLDR/P   MAIN.<RB,LS/L> S1 S2 S3 S4 [OV<1,2,3 4,5>] 22/K
25                         .EJECT
```

```
01
02                ;                 ******** EXAMPLES OF CLI TO INVOKE BIND ********


05                ;                 BIND A SINGLE TASK PROGRAM,
06                ;                 REQUEST A LOAD MAP AND APPEND LISTING TO A PREVI
07                ;                 .LS FILE FROM AN ASSEMBLY, THE .PR FILE NAME IS
FUU00664'000000                     THE SAME AS THAT OF THE.OB FILE.
09
10                ;         X BIND/L=PROG1.LS  PROG1
11
12
13                ;         BIND A MAIN PROGRAM AND 4 TASKS.  THIS IS A
14                ;                 MULTI-TASKING PROGRAM REQUIRING 22(OCTAL) TASKS,
15                ;                 REQUESTING THE LOAD MAP APPENDED
16                ;                 TO THE MAIN PROGRAM'S LISTING FILE.
17
18
19                ;         X BIND/L=MAIN.LS/K=22 MAIN S1 S2 S3 S4
20
21                          .EJECT
```

```
0030 HANDO
01                ;                ********** ECLIPSE INSTRUCTION SET **********
02
                  ;
                  ;                *** FIXED POINT ARITH. & MANIPULATION ***
                  ;                *** AND EXTENDED JUMPS  &  SKIP TESTS ***
06                ;                ----------------------------------------
07
08                ;        ELDA                    ;LOAD  ACCUMULATOR
09                ;        ESTA                    ;STORE ACCUMULATOR
10
11                ;        ELEF                    ;LOAD EFFECTIVE ADDRESS
12
FF                         EJMP                    ;EXTENDED JMP
14                         EJSR                    ;EXTENDED JSR
FF                         EISZ                    ;EXTENDED ISZ
16                         EDSZ                    ;EXTENDED DSZ
17                ;        DAD                     ;DECIMAL ADD
18                ;        DSB                     ;DECIMAL SUBTRACT
19
20                ;        ADI                     ;ADD IMMEDIATE
21                ;        ADDI                    ;EXTENDED ADD IMMEDIATE
22                ;        SBI                     ;SUBTRACT IMMEDIATE
23
24                ;        MUL                     ;UNSIGNED MULTIPLY
25                ;        MULS                    ;SIGNED   MULTIPLY
26                ;        DIV                     ;UNSIGNED DIVIDE
27                ;        DIVS                    ;SIGNED   DIVIDE
28                ;        DIVX                    ;SIGN EXTEND & DIVIDE
                  ;        HLV                     ;HALVE
                  ;        XCH                     ;EXCHANGE ACCUMULATORS
32
33
34                ;                *** LOGICAL INSTRUCTIONS ***
35                ;                ----------------------------
36
37                ;        ANDI                    ;AND IMMEDIATE
38                ;        IOR                     ;INCLUSIVE OR
39                ;        IORI                    ;INCLUSIVE OR IMMEDIATE
40                ;        XOR                     ;EXCLUSIVE OR
41                ;        XORI                    ;EXCLUSIVE OR IMMEDIATE
42                ;        ANC                     ;AND W/COMPLAMENTED SOURCE
43
44                ;        LSH                     ;LOGICAL SHIFT
45                ;        DLSH                    ;DOUBLE LOGICAL SHIFT
46                ;        HXL                     ;HEX SHIFT LEFT
47                ;        DHXL                    ;DOUBLE HEX SHIFT LEFT
48                ;        HXR                     ;HEX SHIFT RIGHT
49
```

```
10031 HANDO
01                ;              *** BYTE MANIPULATION ***
02                ;              ------------------------

                  ;       LDB                     ;LOAD BYTE
                  ;       STB                     ;STORE BYTE
06
07
08                ;              *** BIT MANIPULATION ***
09                ;              ------------------------
10
11                ;       BTO                     ;SET BIT TO ONE
12                ;       BT7                     ;SET BIT TO ZERO
13                ;       SZB                     ;SKIP ON   ZERO   BIT
14                ;       SNB                     ;SKIP ON NON-ZERO BIT
15                ;       SZBO                    ;SKIP ON ZERO BIT & SET TO ONE
16                ;       LOB                     ;LOCATE LEAD BIT
17                ;       LRB                     ;LOCATE & RESET LEAD BIT
18                ;       COB                     ;COUNT BITS
19
20
21                ;              *** DATA MOVEMENT ***
22                ;              --------------------
23
24                ;       BAM                     ;BLOCK ADD & MOVE  Interruptable
25                ;       BLM                     ;BLOCK MOVE
26
27
28                ;              *** STACK MANIPULATION ***
                  ;              -------------------------

31                ;       PSH,POP                 ;PUSH & POP ACCUMULATORS
32                ;       PSHR                    ;PUSH RETURN ADDRESS
33                ;       SAVE                    ;SAVE
34                ;       MSP                     ;MODIFY STACK POINTER
35                ;       POPB                    ;POP BLOCK
36                ;       RTN                     ;RETURN
37                ;       RSTR                    ;RESTORE
38                ;       PSHJ                    ;PUSH P.C. & JUMP
39                ;       POPJ                    ;POP  P.C. & JUMP
40
41
42                ;              *** PROGRAM FLOW ALTERATION ***
43                ;              ------------------------------
44
45                ;       SGT                     ;SKIP IF ACS > ACD
46                ;       SGE                     ;SKIP IF ACS >= ACD
47
48
49                ;              *** SPECIAL EXECUTIONS ***
50                ;              -------------------------
51
52                ;       SYC                     ;SYSTEM CALL
53                ;       XCT                     ;EXECUTE ACCUMULATOR (CONTENTS)
```

```
10032 HANDO
01                      ;                       ELDA --- ESTA --- EJMP --- EJSR
02                      ;               EXTENDED LOAD & STORE ACCUM.  JUMP & JUMP SUBROU
                        ;               ------------------------------------------------

                        ;       LOAD AND STORE THE SYSTEM ACCUMULATORS FROM AND TO
06                      ;               INSTRUCTIONS IDENTICAL TO THEIR NOVA
07                      ;               COUNTERPARTS EXCEPT NO ADDRESS LIMIT
08                      ;               (WITHIN 32K).
09
10                      ;       FORMAT:
11                      ;               ELDA    AC,ADDR
12                      ;               ESTA    AC,ADDR
13                      ;               ELDA    AC,D,X
14                      ;               ESTA    AC,D,X
15                      ;               EISZ    ADDR
 UU00665'112070                         EISZ    D,X
17          000000
18 00667'116070                         EDSZ    ADDR
19          003200
 UU00671'116070                         EDSZ    D,X
21          000000
22                      ;               EJMP    ADDR
23                      ;               EJSR    ADDR
24                      ;               EJMP    D,X
25                      ;               EJSR    D,X
26
27                      ;       WHERE:
28                      ;               AC      IS THE ACCUMULATOR
                        ;               ADDR    IS THE ADDRESS LABEL
                        ;               D       IS THE DISPLACEMENT   (0 - 07777)
                        ;               X       IS THE INDEX MODE     (0,1,2,3)
32
33                      ;       EXAMPLES:
34
35 00673'122470                 ELDA    AC0,ADDR1               ;LOAD AC0 W/CONTENTS OF
36          077556
37 00675'146470                 ESTA    AC1,ADDR1               ;STORE AC0 IN ADDR1
38          077554
39 00677'132470                 ELDA    AC2,5,1                 ;LOAD AC2 WITH CONTENTS
40          000005
41 00701'157070                 ESTA    AC3,10,2                ;STORE AC3 @ 10+AC2
42          000010
43 00703'133470                 ELDA    AC2,15,3                ;LOAD AC3 W/CONTENTS OF
44          000015
45
46 00705'122470                 ELDA@   AC0,ADDR1               ;LOAD AC0 THRU ADDR1
47          177544
48
49
50 00707'112470                 EISZ    ADDR1                   ;INCREMENT ADDR1 SKIP IF
51          077542
52 00711'116470                 EDSZ    ADDR1                   ;DECREMENT ADDR1 SKIP IF
53          077540

   00713'102470                 EJMP    ADDR1                   ;EXTENDED JUMP TO ADDR
            077536
57 00715'106470                 EJSR    ADDR1                   ;EXTENDED JSR  TO ADDR1
58          077534
59 00717'102470                 EJMP    5,1                     ;EXTENDED JUMP 5+P.C.
60          000005
```

```
 0033 HANDO
01 00721'107070          EJSR     10,2                    ;EXTENDED JSR 10+AC2
02       000010
   00723'102470          EJMP@    ADDR1                   ;EXTENDED JUMP THRU ADDR
         177526
   00725'107070          EJSR@    0,2                     ;EXTENDED JSR  THRU AC2
06       100000
```

```
!0034 HANDO
01                ;                    *** EXTENDED MRI INSTRUCTIONS ***
02
                  ;EXTENDED MEMORY REFERENCE INSTRUCTIONS ALLOW FOR THE FULL R    E
                  ;OF ADDRESSABILITY (I.E. 32KW) WITHOUT IMPLEMENTING INDIRFC
                  ;ADDRESSING.
06
07                ;EXTENDED MEMORY REFERENCE INSTRUCTIONS TAKE TWO WORDS OF MEMORY
08                ;STORAGE VS. ONE WORD FOR NORMAL MRI'S. JUST AS MRI'S ARE BROKEN
09                ;DOWN INTO TWO CATERGORIES (I.E. WITH AC'S AND WITHOUT AC'S) THE
10                ;EXTENDED MRI'S ALSO FALL INTO THE SAME TWO GROUPS.
11
12                ;THOSE REQUIRING AC'S ARE: ELEF, ELDA, ESTA
13
14                ;THOSE WITHOUT AC'S ARE: EJMP, EJSR, EDSZ, EISZ
15
16                ;THERE ARE TWO FORMATS FOR EXTENDED MRI'S:
17
18                ;FORMAT 1: INST [@] DISP INDEX    -OR-    INST [@] AC DISP INDEX
19
20                ;FORMAT 2: INST [@] ADDRESS       -OR-    INST [@] AC ADDRESS
21
22                ;THE FIRST FORMAT IS USED WITH THE SPECIFIED MODE OF INDEXING
23                ;(0,1,2,3). WHEN THE SECOND FORMAT IS USED, THE ASSEMBLER MUST
24                ;ATTEMPT TO FORM THE CORRECT MODE OF INDEXING. THE ASSEMBLED
25                ;MODE WILL ALWAYS BE EITHER 0 OR 1. RULES ARE AS FOLLOWS:
26
27                ;MODE IS 1 IF THE CURRENT PC AND ADDRESSED LOCATION HAVE THE
28                ;SAME ADDRESS TYPE. THE ADDRESSES MUST BE BOTH NREL, ZREL,
                  ;OR ABSOLUTE.
                  ;MODE IS 0 IF THE CURRENT PC AND THE ADDRESSED LOCATION DO
32                ;NOT HAVE MATCHING ADDRESS TYPES.
33
34                ;MODE IS 1 IF THE ADDRESSED LOCATION IS EXTERNAL TO THE
35                ;ASSEMBLY (THIS TOPIC COVERED IN INTER PROGRAM COMMUNICATION).
```

```
10035 HANDO
01                  ;                                   ELEF
02                  ;                          LOAD EFFECTIVE ADDRESS
                    ;                          -    --
                    ;                          ----------------------

06                  ;       LOAD THE EFFECTIVE ADDRESS (NOT THE CONTENTS) IN TO
07                  ;              THE INDICATED ACCUMULATOR.
08
09                  ;       POSSIBLE ACCUMULATORS:  1 - AC1
10                  ;                               2 - AC2
11                  ;                               3 - AC3
12                  ;                               0 - AC0
13
14                  ;       FORMAT:
15                  ;              ELEF     AC,ADDR
16                  ;              ELEF     AC,D,X
17
18                  ;       WHERE:
19                  ;              AC       IS THE ACCUMULATOR
20                  ;              ADDR     IS THE ADDRESS LABEL
21                  ;              D        IS THE DISPLACEMENT   (0 - 17777)
22                  ;              X        IS THE INDEX MODE     (0,1,2,3)
23
24                  ;       EXAMPLES:
25
26 00727'162470             ELEF     AC0,ADDR1                ;LOAD A(ADDR1) IN AC0
27        077522
28 00731'166470             ELEF     AC1,5,1                  ;LOAD A(PC+5) IN AC1
          000005
   00733'177070             ELEF     AC3,10,2                 ;LOAD A(10+AC2)IN AC3
31        000010
32 00735'173470             ELEF     AC2,15,3                 ;LOAD A(15+AC3)IN AC2
33        000015
34
35 00737'162470             ELEF@    AC0,ADDR1                ;LOAD INDIRECT AN ADDRES
36        177512
37                                                           ;THRU ADDR1
```

```
!0036 HANDO
01                ;                                  ADI
02                ;                               ADD   IMMEDIATE
                  ;                               --    -
                  ;                               ----------------

06                ;       THE CONTENTS OF THE IMMEDIATE PARAMETER IN THIS INSTRUCT
07                ;               ARE ADDED TO THE UNSIGNED 16 BIT NUMBER IN THE
08                ;               DESIGNATED ACCUMULATOR.  THE CARRY BIT REMAINS
09                ;               UNCHANGED.  THE IMMEDIATE PARAMETER MUST BE
10                ;               IN THE RANGE OF 1 - 4.
11
12                ;       FORMAT:
13                ;               ADI     N,AC
14
15                ;       WHERE:
16                ;               N       IS THE IMMEDIATE VALUE  (1 - 4)
17                ;               AC      IS THE DESIGNATED ACCUMULATOR
18
19                ;       EXAMPLES:
20
21 00741'100010           ADI     1,AC0                   ;ADD ONE TO AC0
22
23 00742'150010           ADI     3,AC2                   ;ADD  3  TO AC2
24
25 00743'114010           ADI     CST1,AC3                ;ADD CST1 TO AC3
26
27        000001 CST1=    1
```

```
!0037 HANDO
01                ;                                ADDI
02                ;                  EXTENDED ADDI     IMMEDIATE
                  ;                  ----------------------------

                  ;       THE CONTENTS OF THE IMMEDIATE FIELD (IN THIS CASE THE 2ND
06                ;              WORD OF THE INST.) ARE ADDED TO THE DESIGNATED
07                ;              ACCUMULATOR.  THE CARRY BIT REMAINS UNAFFECTED.
08                ;              NOTE: THE IMMEDIATE VALUE IS NO LONGER LIMITED
09                ;                    TO  1 - 4.
10
11                ;       FORMAT:
12                ;              ADDI     I,AC
13
14                ;       WHERE:
15                ;              I        IS THE 16 BIT SIGNED IMMEDIATE FIELD
16                ;              AC       IS THE DESIGNATED ACCUMULATOR
17
18                ;       EXAMPLES
19
20 00744'163770          ADDI     500,AC0                  ;ADD 500 TO AC0
21        000500
22 00746'173770          ADDI     377,AC2                  ;ADD 377 TO AC2
23        000377
24 00750'177770          ADDI     776,AC3                  ;ADD 776 TO AC3
25        000776
```

```
!0038 HANDO
01                ;                                SBI
02                ;                      SUBTRACT   IMMEDIATE
                  ;                      - -        -
                  ;                      ---------------------

06                ;       THE CONTENTS OF THE IMMEDIATE FIELD ARE SUBTRACTED TO TH
07                ;             NUMBER CONTAINED IN THE DESIGNATED ACCUMULATOR.
08                ;             THE IMMEDIATE VALUF IS LIMITED FROM 1 - 4.
09                ;             THE CARRY BIT REMAINS UNCHANGED.
10                ;       NOTE: THERE IS NO EXTENDED SUBTRACT IMMEDIATE (SBBI?).
11                ;             TO ACCOMPLISH THIS, USE AN ADDI W/ A NEGATIVE
12                ;                NUMBER.
13
14                ;       FORMAT:
15                ;                SBI      N,AC
16
17                ;       WHERE:
18                ;                N        IS THE IMMEDIATE VALUE FROM 1 - 4
19                ;                AC       IS THE DESIGNATED ACCUMULATOR
20
21                ;       EXAMPLES:
22
23 00752'120110           SBI      2,AC0                    ;SUBTRACT 2 FROM AC0
24
25 00753'174110           SBI      4,AC3                    ;     "     4 FROM AC3
26
27 00754'110110           SBI      CST1,AC2                 ;     "     1 FROM AC2
```

```
!0039 HANDO
01                ;                                MUL
02                ;                    UNSIGNED    MULTIPLY
                  ;                                ---
                  ;                    ____________________

06                ;       THE 16-BIT UNSIGNED NUMBER IN AC1 IS MULTIPLIED BY THE 16
07                ;           UNSIGNED NUMBER IN AC2 TO YIELD A 32-BIT INTER-
08                ;           MEDIATE RESULT.THE 16-BIT NUMBER IN AC0 IS ADDED
09                ;           TO THIS PRODUCING THE FINAL RESULT.  THAT RESULT
10                ;           IS A 32-BIT UNSIGNED NUMBER OCCUPYING AC0 & AC1.
11                ;           BIT 0 OF AC0 IS THE HIGH-ORDER & BIT 15 OF AC1
12                ;           BECOMES THE LOW ORDER PORTION OF THE RESULT. AC2
13                ;           REMAINS UNCHANGED (AS DOES AC3).  BECAUSE THE
14                ;           RESULT IS DOUBLE-WORD VALUE, OVERFLOW CANNOT
15                ;           OCCUR.
16
17                ;       FORMAT:
18                ;               MUL
19
20                ;       NOTE: THERE ARE NO PARAMETERS ON THIS INSTRUCTION.  ALL
21                ;           REQUIRED REGISTERS SHOULD BE PRELOADED BEFORE
22                ;           EXECUTING THIS INSTRUCTION.  IF NOTHING IS TO
23                ;           BE ADDED FROM AC0, IT SHOULD BE ZEROED OUT:
24                ;           SUBO    AC0,AC0.
25
26                ;       EXAMPLES:       SELF-EXPLANITORY
27
28 00755'143710          MUL
```

```
!0040 HANDO
01                ;                                MULS
02                ;                          SIGNED     MULTIPLY
                  ;                            -          ---
                  ;                          ----------------

06                ;       THE 16-BIT 2'S COMPLIMENT NUMBER IN AC1 IS MULTIPLIED BY
07                ;             THE 16-BIT 2'S COMPLIMENT NUMBER IN AC2 YIELDING
08                ;             A 32-BIT INTERMEDIATE RESULT.  THE 16-BIT SIGNED
09                ;             2'S COMPLIMENT NUMBER IN AC0 IS ADDED TO THIS
10                ;             GIVING THE FINAL RESULT WHICH OCCUPIES AC0 & AC1
11                ;             BIT 0 OF AC0 IS THE SIGN BIT AND BIT 15 OF AC1
12                ;             IS THE LOW ORDER BIT OF THE RESULT.
13
14                ;       FORMAT:
15                ;             MULS
16
17                ;       NOTE:    AS WITH MUL, THERE ARE NO PARAMETERS.  ACCUMULAT
18                ;             AC0,AC1,AC2  ARE ASSUMMED TO BE LOADED PRIOR TO
19                ;             THE EXECUTION OF THIS INSTRUCTION.
20
21                ;       EXAMPLES:        SELF-EXPLANITORY
22 00756'077201           MULS
```

```
!0041 HANDO
01                ;                              DIV
02                ;                    UNSIGNED   DIVIDE
                  ;                               ---
                  ;
                  ;  ------------------------------------

06                ;       THE 32-BIT UNSIGNED NUMBER CONTAINED IN AC0 & AC1 IS
07                ;               DIVIDED BY THE 16-BIT UNSIGNED NUMBER IN
08                ;               AC2.  THE QUOTIENT AND REMAINDER ARE 16-BIT
09                ;               NUMBERS RESIDING IN AC1 & AC0 RESPECTIVELY.
10                ;               THE CARRY BIT IS SET TO 0.  THE CONTENTS
11                ;               OF AC2 REMAINS UNCHANGED.
12
13                ;       FORMAT:
14                ;               DIV
15
16                ;       NOTE:   THERE ARE NO PARAMETERS FOR THIS INSTRUCTION.  A
17                ;               ACCUMULATORS ARE ASSUMED PRELOADED BEFORE EXECUT
18                ;               OF THIS INSTRUCTION.  BEFORE THE DIVIDE, AC0 IS
19                ;               COMPARED TO AC2.  IF AC0 > AC2, AN OVERFLOW
20                ;               CONDITION IS INDICATED.  THE CARRY BIT IS SET
21                ;               TO 1 AND THE OPERATION IS TERMINATED.
22
23                ;       EXAMPLES:       SELF-EXPLANITORY
24
25 00757'153710           DIV
```

```
10042 HANDO
01                ;                                DIVS
02                ;                          SIGNED    DIVIDE
                  ;                                    ---
                  ;                          ----------------

06                ;       THE 32-BIT SIGNED 2'S COMPLIMENT NUMBER CONTAINED IN ACO
07                ;               AC1 IS DIVIDED BY THE 16-BIT SIGNED 2'S COMPLIME
08                ;               NUMBER IN AC2; THE QUOTIENT & REMAINDER ARE SIGN
09                ;               16-BIT NUMBERS OCCUPYING AC1 & ACO RESPECTIVELY.
10                ;               SIGN OF THE QUOTIENT ARE DETERMINED BY ALGEBRAIC
11                ;               RULES;  SIGN OF THE REMAINDER IS ALWAYS THE SAME
12                ;               AS THE SIGN OF THE QUOTIENT EXCEPT A ZERO VALUE
13                ;               IS ALWAYS POSITIVE.
14
15
16                ;       FORMAT:
17                ;               DIVS
18
19                ;       NOTE:   THERE ARE NO PARAMETERS FOR THIS INSTRUCTION.  A
20                ;               ACCUMULATORS ARE ASSUMMED PRELOADED PRIOR TO THE
21                ;               EXECUTION OF THIS INSTRUCTION.  IF THE MAGNITUDE
22                ;               OF THE QUOTIENT IS SUCH THAT IT WILL NOT FIT INT
23                ;               AC1, OVERFLOW IS INDICATED BY THE CARRY BIT BEIN
24                ;               SET TO 1 AND THE OPERATION BEING TERMINATED.  TH
25                ;               CONTENTS OF AC0 & AC1 ARE UNPREDICTABLE.
26
27                ;       EXAMPLES:       SELF-EXPLANITORY
28 00760'077001           DIVS
```

```
!0043 HANDO
01                ;                              DIVX
02                ;                    SIGN EXTEND    AND DIVIDE
                  ;                         -              ---
                  ;                    ____________________________

                  ;        THE SIGN OF THE NUMBER IN AC1 IS EXTENDED INTO AC0
07                ;              BY PLACING A COPY OF BIT 0 IN AC1 INTO
08                ;              EACH BIT OF AC0.  AFTER SIGN EXTENSION A
09                ;              SIGNED DIVIDE IS PERFORMED.
10
11                ;        FORMAT:
12                ;              DIVX
13
14                ;        NOTE: THERE ARE NO PARAMETERS FOR THIS INSTRUCTION.  AFT
15                ;              EXECUTION OF THIS INSTRUCTION, THE CONTENTS OF
16                ;              AC0 & AC1 WILL CONTAIN THE RESULT AS IN DIVS.
17
18                ;        EXAMPLES:       SELF-EXPLANITORY
19 00761'137710          DIVX
```

```
!0044 HANDO
01                ;                                  HLV
02                ;                            HALVE   ACCUMULATOR
                  ;                            - --
                  ;                            ____________________

06                ;       THE SIGNED 2'S COMPLIMENT NUMBER CONTAINED IN THE DESIGN
07                ;             ACCUMULATOR IS DIVIDED BY 2 AND ROUNDED TOWARD 0
08                ;             THE DESIGNATED ACCUMULATOR CONTAINS THE RESULT.
09
10                ;       FORMAT:
11                ;             HLV     AC
12
13                ;       WHERE:  AC      IS THE DESIGNATED ACCUMULATOR
14
15                ;       NOTE: IF THE NUMBER IS POSITIVE, DIVISION IS ACCOMPLISHE
16                ;             BY SHIFTING THE NUMBER RIGHT 1 BIT.  IF NEGATIVE
17                ;             DIVISION IS ACCOMPLISHED BY NEGATING IT, SHIFTIN
18                ;             IT RIGHT 1 BIT AND NEGATING IT BACK.
19
20                ;       EXAMPLES:
21
22 00762'143370           HLV     AC0                     ;DIVIDE AC0 BY 2
23
24 00763'147370           HLV     AC1                     ;DIVIDE AC1 BY 2
25
26 00764'153370           HLV     AC2                     ;DIVIDE AC2 BY 2
```

```
10045 HANDO
01                ;                                  ANDI
02                ;                               AND     IMMEDIATE
                  ;                               ---     -
04                ;                               ----------------

                  ;       THE DESIGNATED ACCUMULATOR IS ANDED BY THE IMMEDIATE FIE
07                ;               WHICH IS TREATED AS A 16-BIT QUANTITY.  THIS
08                ;               RESULT IS PLACED IN THE DESIGNATED ACCUMULATOR.
09
10                ;       FORMAT:
11                ;               ANDI    I,AC
12
13                ;       WHERE:
14                ;               I       IS THE IMMEDIATE VALUE (1 WORD)
15                ;               AC      IS THE DESIGNATED ACCUMULATOR
16
17                ;       EXAMPLES:
18
19 00765'143770           ANDI    177,AC0                 ;AND 177 AGAINST AC0
20       000177
21 00767'147770           ANDI    377,AC1                 ;AND 377 AGAINST AC1
22       000377
23 00771'157770           ANDI    OCT70,AC3               ;AND VALUE OF OCT70
24       000070
25                                                        ;AGAINST AC3
26
27       000070 OCT70=    70                              ;OCTAL 70
```

```
!0046 HANDO
01                  ;                                   IOR
02                  ;                          INCLUSIVE   OR
                    ;                          -           --
                    ;                          ----------------

06                  ;       THE CONTENTS OF ACS ARE INCLUSIVELY OR'D WITH THE CONTEN
07                  ;              OF ACD WITH THE RESULT PLACED IN ACD.
08
09                  ;       FORMAT:
10                  ;              IOR      ACS,ACD
11
12                  ;       WHERE:
13                  ;              ACS      IS THE SOURCE ACCUMULATOR
14                  ;              ACD      IS THE DESTINATION ACCUMULATOR
15
16                  ;       EXAMPLES:
17
18 00773'104410             IOR      AC0,AC1                    ;IOR AC0 AGAINST AC1
19
20 00774'130410             IOR      AC1,AC2                    ;IOR AC1 AGAINST AC2
21
22 00775'154410             IOR      AC2,AC3                    ;IOR AC2 AGAINST AC3
```

```
!0047 HANDO
01                ;
02                ;                              IORI
                  ;                    INCLUSIVE    OR IMMDEIATE
                  ;                    -            -- -
                  ;                    ------------------------

                  ;       THE CONTENTS OF THE IMMEDIATE FIELD ARE TREATED AS A 16-
07                ;             VALUE AND INCLUSIVELY OR'D AGAINST THE
08                ;             DESIGNATED ACCUMULATOR; THE RESULT IS
09                ;             RETAINED IN THE DESIGNATED ACCUMULATOR.
10
11                ;       FORMAT:
12                ;             IORI    I,AC
13
14                ;       WHERE:
15                ;             I       IS THE IMMEDIATE VALUE
16                ;             AC      IS THE DESIGNATED ACCUMULATOR
17
18                ;       EXAMPLES:
19
20 00776'103770         IORI    377,AC0         ;IOR 377 AGAINST AC0
21       000377
22 01000'107770         IORI    776,AC1         ;IOR 776 AGAINST AC1
23       000776
24 01002'117770         IORI    OCT70,AC3       ;IOR THE VALUE OF OCT70
25       000070
26                                               ;AGAINST AC3
```

```
!0048 HANDO
01                ;                                XOR
02                ;                          EXCLUSIVE   OR
                  ;                            -           --
                  ;                          ------------------

06
07                ;       THE CONTENTS OF ACS ARE EXCLUSIVELY OR'D AGAINST THE
08                ;               THE CONTENTS OF ACD WITH THE RESULT
09                ;               BEING PLACED IN ACD.
10
11                ;       FORMAT:
12                ;               XOR     ACS,ACD
13
14                ;       WHERE:
15                ;               ACS     IS THE SOURCE ACCUMULATOR
16                ;               ACD     IS THE DESTINATION ACCUMULATOR
17
18                ;       EXAMPLES:
19
20 01004'104510           XOR     AC0,AC1         ;XOR AC0 AGAINST AC1
21
22 01005'130510           XOR     AC1,AC2         ;XOR AC1 AGAINST AC2
23
24 01006'154510           XOR     AC2,AC3         ;XOR AC2 AGAINST AC3
```

```
!0049 HANDO
01                  ;                                   XORI
02                  ;                        EXCLUSIVE    OR IMMEDIATE
                    ;                          -          -- -
                    ;                        ------------------------
06                  ;       THE CONTENTS OF THE IMMEDIATE VALUE ARE TREATED AS A 16-
07                  ;               QUANTITY AND EXCLUSIVELY OR'D WITH THE CONTENTS
08                  ;               OF THE DESIGNATED ACCUMULATOR WITH THE RESULT
09                  ;               BEING PLACED IN THAT ACCUMULATOR.
10
11                  ;       FORMAT:
12                  ;               XORI    I,AC
13
14                  ;       WHERE:
15                  ;               I       IS THE IMMEDIATE VALUE (1 WORD)
16                  ;               AC      IS THE DESIGNATED ACCUMULATOR
17
18                  ;       EXAMPLES:
19
20 01007'123770             XORI    377,AC0         ;XOR 377 AGAINST AC0
21        000377
22 01011'127770             XORI    177,AC1         ;XOR 177 AGAINST AC1
23        000177
24 01013'137770             XORI    OCT70,AC3       ;XOR THE VALUE OF OCT70
25        000070
26                                                  ;AGAINST AC3
```

```
!0050 HANDO
01                ;                              ANC
02                ;                    AND WITH   COMPLAMENTED SOURCE
                  ;                    --         -
                  ;                    ------------------------------

06                ;       THE LOGICAL 1'S COMPLAMENT OF ACS IS AND'D AGAINST ACD W
07                ;               THE RESULT PLACED IN ACD.  IN THIS CASE, IF
08                ;               CORRESPONDING POSITIONS IN ACS & ACD ARE SET
09                ;               TO 0 & 1 RESPECTIVELY, THE RESULT WILL BE
10                ;               A 1.  ACS REMAINS UNCHANGED.
11
12                ;       FORMAT:
13                ;               ANC     ACS,ACD
14
15                ;       WHERE:
16                ;               ACS     IS THE SOURCE ACCUMULATOR
17                ;               ACD     IS THE DESTINATION ACCUMULATOR
18
19                ;       EXAMPLES:
20
21 01015'104610           ANC     AC0,AC1         ;ANC AC0 AGAINST AC1
22
23 01016'130610           ANC     AC1,AC2         ;ANC AC1 AGAINST AC2
24
25 01017'154610           ANC     AC2,AC3         ;ANC AC2 AGAINST AC3
```

```
!0051 HANDO
01                ;                                LSH
02                ;                     LOGICAL     SHIFT
                  ;                     -           --
                  ;                     ------------------

06                ;        THE CONTENTS OF THE DESTINATION ACCUMULATOR ARE SHIFTED
07                ;            LEFT OR RIGHT DEPENDING ON THE SIGNED 2'S COMPLIMENT
08                ;                NUMBER CONTAINED IN BITS 8-15 OF ACS.  IF THE
09                ;                NUMBER CONTAINED IN ACS IS POSITIVE, SHIFTING
10                ;                OCCURS TO THE LEFT; IF NEGATIVE SHIFTING IS TO
11                ;                THE RIGHT; IF ZERO, NO SHIFTING TAKES PLACE.
12                ;                IF THE VALUE IS > 15, NO SHIFTING TAKES PLACE.
13                ;                BITS SHGIFTED OUT ARE LOST; VACATED POSITIONS
14                ;                ARE FILLED WITH ZEROES.  BITS 0-7 OF ACS ARE
15                ;                IGNORED AND ACS IS LEFT UNCHANGED.
16
17                ;        FORMAT:
18                ;                LSH     ACS,ACD
19
20                ;        WHERE:
21                ;                ACS     CONTAINS THE SHIFTING DIRECTIONS
22                ;                ACD     IS THE ACCUMULATOR TO BE SHIFTED
23
24                ;        EXAMPLES:
25
26 01020'121210           LSH     AC1,AC0          ;SHIFT AC0 BASED ON AC1
27
28 01021'155210           LSH     AC2,AC3          ;SHIFT AC3 BASED ON AC2

   01022'135210           LSH     AC1,AC3          ;SHIFT AC3 BASED ON AC1
```

```
10052 HANDO
01                ;                                DLSH
02                ;                  DOUBLE LOGICAL    SHIFT
                  ;                  -      -          --
                  ;                  ------------------------

06
07                ;        THIS SHIFT IS THE SAME AS WITH LSH EXCEPT THAT A 32-BIT
08                ;                ACCUMULATOR IS SHIFTED.  THIS 32-BIT ACCUMULATOR
09                ;                IS ACTUALLY ACD & ACD+1. THE SHIFT VALUE IN ACS
10                ;                CAN BE 1-31.
11                ;                NOTE:   IF ACD = AC1    THEN ACD+1 = AC2
12                ;                        IF ACD = AC3    THEN ACD+1 = AC0
13
14                ;        FORMAT:
15                ;                DLSH    ACS,ACD
16
17                ;        WHERE:
18                ;                ACS     CONTAINS THE SHIFTING DIRECTIONS
19                ;                ACD     IS THE ACCUMULATOR PAIRS AS FOLLOWS:
20                ;                                AC0 = AC0,AC1
21                ;                                AC1 = AC1,AC2
22                ;                                AC2 = AC2,AC3
23                ;                                AC3 = AC3,AC0
24
25                ;        EXAMPLES:
26
27 01023'131310           DLSH    AC1,AC2         ;AC2+AC3 ARE SHIFTED BY AC1
28
   01024'105310           DLSH    AC0,AC1         ;AC1+AC2 ARE SHIFTED BY AC0

   01025'155310           DLSH    AC2,AC3         ;AC3+AC0 ARE SHIFTED BY AC2
32
33 01026'115310           DLSH    AC0,AC3         ;AC3+AC0 ARE SHIFTED BY AC0
34                                                ;NOTICE ACCUMULATOR OVERLAP HERE
```

```
10053 HANDO
01                  ;                                   HXL
02                  ;                                 HEX   SHIFT LEFT
                    ;                                 - -         -
                    ;                                 ---------------

06                  ;       THE CONTENTS OF THE DESIGNATED ACCUMULATOR ARE SHIFTED BY
07                  ;               THE VALUE OF THE IMMEDIATE FIELD.  THIS VALUE
08                  ;               CAN BE FROM 1 - 4.  BITS SHIFTED OUT ARE LOST;
09                  ;               VACATED POSITIONS ARE FILLED WITH ZEROES.
10                  ;               BECAUSE THIS IS A HEX SHIFT, 4 BITS ARE SHIFTED
11                  ;               FOR EACH VALUE OF THE IMMEDIATE FIELD.
12                  ;               IE: 1= 4 BITS, 2= 8 BITS, 3= 12 BITS, 4= 16 BITS
13
14                  ;       FORMAT:
15                  ;               HXL     N,AC
16
17                  ;       WHERE:
18                  ;               N       IS THE OF BITS (X 4) TO BE SHIFTED
19                  ;               AC      IS THE ACCUMULATOR TO BE SHIFTED
20
21                  ;       EXAMPLES:
22
23 01027'105410             HXL     (1) AC1         ;AC1 IS SHIFTED LEFT 4 BITS
24
25 01030'155410             HXL     3,AC3           ;AC3 IS LEFT SHIFTED 12 BITS
26
27 01031'161410             HXL     4,AC0           ;AC0 IS SHIFTED OUT (CLEARED)
```

IMMEDIATE

```
!0054 HANDO
01                  ;                                   DHXL
02                  ;                        DOUBLE HEX     SHIFT LEFT
                    ;                        -      - -           -
                    ;                        ------------------------

06                  ;       THIS INSTRUCTION FUNCTIONS THE SAME AS THE PREVIOUS ONE
07                  ;               EXCEPT THE ACCUMULATOR IS 32 BITS BY USING
08                  ;               AC & AC+1.
09
10                  ;       FORMAT:
11                  ;               DHXL    N,AC
12
13                  ;       WHERE:
14                  ;               N       IS THE NUMBER OF BITS (X 4) TO BE SHIFTE
15                  ;               AC      IS THE REGISTER PAIR AS FOLLOWS:
16                  ;                               AC0 = AC0,AC1
17                  ;                               AC1 = AC1,AC2
18                  ;                               AC2 = AC2,AC3
19                  ;                               AC3 = AC3,AC0
20
21                  ;       EXAMPLES:
22
23 01032'105610             DHXL    1,AC1           ;AC1+AC2 ARE SHIFTED 4 BITS
24
25 01033'131610             DHXL    2,AC2           ;AC2+AC3 ARE SHIFTED 8 BITS
26
27 01034'155610             DHXL    3,AC3           ;AC3+AC0 ARE SHIFTED 12 BITS
28
   01035'161610             DHXL    4,AC0           ;AC0+AC1 ARE SHIFTED 16 BITS
                                                    ;IE. AC0=AC1 - AC1 IS CLEARED
```

```
10055 HANDO
01                ;                                  HXR
02                ;                               HEX   SHIFT RIGHT
                  ;                               - -         -
                  ;                               ----------------

06                ;       THIS INSTRUCTION FUNCTIONS IDENTICALLY TO HXL EXCEPT THA
07                ;            THE SHIFTING OCCURS TO THE RIGHT INSTEAD.
08
09                ;       FORMAT:
10                ;                HXR      N,AC
11
12                ;       WHERE:
13                ;                N        IS THE NUMBER OF BITS (X 4) TO BE SHIFTE
14                ;                AC       IS THE ACCUMULATOR TO BE SHIFTED
15
16                ;       EXAMPLES:
17
18 01036'105510           HXR      1,AC1          ;AC1 IS SHIFTED 4 BITS RIGHT
19
20 01037'131510           HXR      2,AC2          ;AC2 IS SHIFTED 8 BITS RIGHT
21
22 01040'155510           HXR      3,AC3          ;AC3 IS SHIFTED 12 BITS RIGHT
23
24 01041'161510           HXR      4,AC0          ;AC0 IS CLEARED
```

```
!0056 HANDO
01                ;                                DHXR
02                ;                  DOUBLE HEX    SHIFT RIGHT
                  ;                  -      - -    -     -
                  ;                  ------------------------

06                ;       THIS INSTRUCTION FUNCTIONS IDENTICALLY TO THE DHXL EXCEP
07                ;              THAT THE SHIFTING OCCURS TO THE RIGHT.
08
09                ;       FORMAT:
10                ;              DHXR    N,AC
11
12                ;       WHERE:
13                ;              N       IS THE NUMBER OF BITS (X 4) TO BE SHIFTE
14                ;              AC      IS THE ACCUMULATOR PAIR AS FOLLOWS:
15                ;                              AC0 = AC0,AC1
16                ;                              AC1 = AC1,AC2
17                ;                              AC2 = AC2,AC3
18                ;                              AC3 = AC3,AC0
19
20                ;       EXAMPLES:
21
22 01042'105710           DHXR    1,AC1           ;AC1+AC2 ARE SHIFTED 4 BITS RIGH
23
24 01043'131710           DHXR    2,AC2           ;AC2+AC3 ARE SHIFTED 8 BITS RIGH
25
26 01044'155710           DHXR    3,AC3           ;AC3+AC0 ARE SHIFTED 12 BITS RIG
27
28 01045'161710           DHXR    4,AC0           ;AC0+AC1 ARE SHIFTED 16 BITS RIG
                                                  ;AC0 IS CLEARED AND AC1=AC0
```

```
!0057 HANDO
01                ;                                LDB
02                ;                             LOAD   BYTE
                  ;                             -  -   -
                  ;                             ---------

06                ;       THE 8-BIT BYTE ADDRESSED BY THE BYTE POINTER (POINTER)
07                ;              IN ACS IS LOADED INTO BITS 8-15 OF ACD; BITS
08                ;              0-7 ARE SET TO ZERO.  ACS REMAINS UNCHANGED.
09
10                ;       FORMAT:
11                ;              LDB      ACS,ACD
12
13                ;       WHERE:
14                ;              ACS      CONTAINS THE BYTE POINTER (POINTER) TO
15                ;              ACD      IS THE ACCUMULATOR TO RECEIVE THE BYTE A
16
17                ;       EXAMPLES:
18
19 01046'132710          LDB      AC1,AC2          ;AC2 IS LOADED FROM THE BYTE
20                                                 ;ADDRESSED BY AC1
21
22 01047'162710          LDB      AC3,AC0          ;AC0 IS LOADED FROM THE BYTE
23                                                 ;ADDRESSED BY AC3
24
25 01050'152710          LDB      AC2,AC2          ;AC2 IS LOADED FROM THE BYTE
26                                                 ;THAT IT NOW POINTS TO
```

```
10058 HANDO
01                ;                                   STB
02                ;                              STORE   BYTE
                  ;                              --      -
                  ;                              ----------

06                ;       BITS 8-15 OF ACD ARE STORED IN THE BYTE ADDRESSED BY THE
07                ;              POINTER (BOINTER) IN ACS.  THE CONTENTS OF ACS &
08                ;              REMAIN UNALTERED.
09
10                ;       FORMAT:
11                ;              STB      ACS,ACD
12
13                ;       WHERE:
14                ;              ACS      IS THE BYTE POINTER (BOINTER) TO MEMORY
15                ;              ACD      IS THE ACCUMULATOR TO BE STORED
16
17                ;       EXAMPLES:
18
19 01051'133010          STB      AC1,AC2          ;BITS 8-15 OF AC2 TO BE STORED
20                                                  ;AT ADDRESS CONTAINED IN AC1
21
22 01052'163010          STB      AC3,AC0          ;BITS 8-15 OF AC0 TO BE STORED
23                                                  ;AT ADDRESS CONTAINED IN AC3
24
25 01053'147010          STB      AC2,AC1          ;BITS 8-15 OF AC1 TO BE STORED
26                                                  ;AT ADDRESS CONTAINED IN AC2
```

```
10059 HANDO
01                ;                                BTO
02                ;                          SET BIT   TO ONE
                  ;                           - -      -
                  ;                          ------------------

06                ;       THE BIT IN MEMORY ADDRESSED BY THE 32-BIT BIT POINTER SP
07                ;             VIA ACS & ACD IS SET TO 1.  ACS CONTAINS THE HIG
08                ;             ORDER 16 BITS AND ACD CONTAINS THE LOW-ORDER. IF
09                ;             ACS AND ACD ARE THE SAME, THEY REPRESENT THE LOW
10                ;             ORDER BITS AND THE HIGH-ORDER ARE ASSUMED 0.
11
12                ;       FORMAT:
13                ;             BTO     ACS,ACD
14
15                ;       WHERE:
16                ;             ACS     IS THE HIGH-ORDER 16 BITS IN THE ADDRESS
17                ;             ACD     IS THE  LOW-ORDER 16 BITS IN THE ADDRESS
18
19                ;       EXAMPLES:
20
21 01054'132010           BTO     AC1,AC2          ;THE BIT ADDRESSED BY AC1-AC2 IS
22
23 01055'172010           BTO     AC3,AC2          ;THE BIT ADDRESSED BY AC3-AC2 IS
24
25 01056'142010           BTO     AC2,AC0          ;THE BIT ADDRESSED BY AC2-AC0 IS
26
27 01057'102010           BTO     AC0,AC0          ;THE BIT ADDRESSED BY AC0     IS
```

```
!0060 HANDO
01                ;                                BTZ
02                ;                          SET BIT   TO ZERO
                  ;                            -    -  -
                  ;                          ---------------

06                ;       THIS INSTRUCTION FUNCTIONS IDENTICALLY TO THE PREVIOUS O
07                ;               EXCEPT THE BIT IS SET TO ZERO INSTEAD.
08
09                ;       FORMAT:
10                ;               BTZ     ACS,ACD
11
12                ;       WHERE:
13                ;               ACS     IS THE HIGH-ORDER 16 BITS OF ADDRESS
14                ;               ACD     IS THE  LOW-ORDER 16 BITS OF ADDRESS
15
16                ;       EXAMPLES:
17
18 01060'132110           BTZ     AC1,AC2         ;THE BIT ADDRESSED BY AC1-AC2 IS
19
20 01061'142110           BTZ     AC2,AC0         ;THE BIT ADDRESSED BY AC2-AC0 IS
21
22 01062'166110           BTZ     AC3,AC1         ;THE BIT ADDRESSED BY AC3-AC1 IS
23
24 01063'102110           BTZ     AC0,AC0         ;THE BIT ADDRESSED BY AC0     IS
```

```
10061 HANDO
01                ;                                SZB
02                ;                         SKIP ON   ZERO BIT
                  ;                         -         -    -
                  ;                         ------------------
                  ;
06                ;       THE 32 BIT POINTER FORMED BY ACS+ACD IS USED TO ADDRESS
07                ;               TO TEST FOR A ZERO BIT.  IF THE BIT IS ZERO, THE
08                ;               NEXT WORD IS SKIPPED.  IF THE BIT IS ONE, CONTRO
09                ;               DROPS TO THE NEXT SEQUENTIAL WORD.
10
11                ;       FORMAT:
12                ;               SZB     ACS,ACD
13
14                ;       WHERE:
15                ;               ACS     IS THE HIGH-ORDER 16 BITS OF THE MEM. AD
16                ;               ACD     IS THE  LOW-ORDER 16 BITS OF THE MEM. AD
17
18                ;       EXAMPLES:
19
20 01064'132210           SZB     AC1,AC2         ;SKIP NSW IF A(AC1+AC2) IS 0
21
22 01065'162210           SZB     AC3,AC0         ;SKIP NSW IF A(AC3+AC0) IS 0
23
24 01066'146210           SZB     AC2,AC1         ;SKIP NSW IF A(AC2+AC1) IS 0
```

```
!0062 HANDO
01                ;                                SNB
02                ;                         SKIP ON   NON-ZERO BIT
                  ;                          -         -        -
                  ;                         ------------------------

06                ;       THE LOCATION IN MEMORY ADDRESS BY THE 32-BIT BIT POINTER
07                ;              OBTAINED FROM ACS+ACD IS TESTED FOR A NON-ZERO
08                ;              BIT.  IF NON-ZERO, THE NEXT SEQUENTIAL WORD IS
09                ;              SKIPPED; ELSE CONTROL PASSES TO THE NSW.
10
11                ;       FORMAT:
12                ;              SNB      ACS,ACD
13
14                ;       WHERE:
15                ;              ACS      IS THE HIGH-ORDER 16 BITS OF THE MEM. AD
16                ;              ACD      IS THE  LOW-ORDER 16 BITS OF THE MEM. AD
17
18                ;       EXAMPLES:
19
20 01067'132770           SNB      AC1,AC2          ;SKIP NSW IF A(AC1+AC2) = 1
21
22 01070'172770           SNB      AC3,AC2          ;SKIP NSW IF A(AC3+AC2) = 1
23
24 01071'116770           SNB      AC0,AC3          ;SKIP NSW IF A(AC0+AC3) = 1
25
26 01072'126770           SNB      AC1,AC1          ;SKIP NSW IF A(AC1)     = 1
```

```
10063 HANDO
01                ;                              SZBO
02                ;              SKIP ON ZERO BIT    AND SET TO ONE
                  ;              -       -   -                   -
                  ;              ------------------------------
05
06                ;        THIS INSTRUCTION WORKS IDENTICALLY TO THE SZB EXCEPT THA
07                ;             IF THE BIT IS FOUND TO BE ZERO, IT IS SET TO ONE
08
09                ;        FORMAT:
10                ;                SZBO    ACS,ACD
11
12                ;        WHERE:
13                ;                ACS     IS THE HIGH-ORDER 16-BITS OF THE MEMORY
14                ;                ACD     IS THE  LOW-ORDER 16-BITS OF THE MEMORY
15
16                ;        EXAMPLES:
17
18 01073'132310           SZBO    AC1,AC2          ;A(AC1+AC2) IS TESTED AND SET
19
20 01074'162310           SZBO    AC3,AC0          ;A(AC3+AC0) IS TESTED AND SET
21
22 01075'102310           SZBO    AC0,AC0          ;A(AC0)     IS TESTED AND SET
```

```
!0064 HANDO
01                ;                                   LOB
02                ;                                LOCATE   LEAD BIT
                  ;                                --          -
                  ;                                ------------------

06                ;       THE CONTENTS OF ACS IS SCANNED FOR HIGH-ORDER ZEROES AND
07                ;               A NUMBER EQUAL TO THAT FOUND IS ADDED TO THE
08                ;               SIGNED 2'S COMPLIMENT VALUE IN ACD.
09                ;               ACS REMAINS UNCHANGED UNLESS ACS & ACD
10                ;                       THE SAME.
11
12                ;       FORMAT:
13                ;               LOB     ACS,ACD
14
15                ;       WHERE:
16                ;               ACS     IS THE SOURCE ACCUMULATOR TO BE INSPECTE
17                ;               ACD     IS THE DESTINATION ACCUMULATOR TO BE ADD
18
19                ;       EXAMPLES:
20
21 01076'132410           LOB     AC1,AC2         ;AC1 IS SCANNED AND THE
22                                                ;COUNT ADDED IN AC2
23
24 01077'162410           LOB     AC3,AC0         ;AC3 IS SCANNED AND THE
25                                                ;COUNT ADDED TO AC0
26
27 01100'126410           LOB     AC1,AC1         ;AC1 IS SCANNED AND THE
28                                                ;COUNT IS ADDED TO AC1
                                                  ;CHANGING ITS INITIAL VALUE
```

```
10065 HANDO
01                ;                                COB
02                ;                             COUNT   BITS
                  ;                             --      -
                  ;                             ----------

06                ;       THE CONTENTS OF ACS ARE SCANNED FOR ONES AND A NUMBER EQ
07                ;               TO THE AMOUNT FOUND IS ADDED TO THE SIGNED 2'S
08                ;               COMPLIMENT VALUE IN ACD.  ACS REMAINS UNCHANGED
09                ;               UNLESS ACS & ACD ARE THE SAME.
10
11                ;       FORMAT:
12                ;               COB     ACS,ACD
13
14                ;       WHERE:
15                ;               ACS     IS THE SOURCE ACCUMULATOR TO BE SCANNED
16                ;               ACD     IS THE DESINATION ACCUMULATOR TO BE ADDE
17
18                ;       EXAMPLES:
19
20 01101'132610           COB     AC1,AC2         ;COUNT THE BITS IN AC1
21                                                ;AND ADD RESULT TO AC2
22
23 01102'162610           COB     AC3,AC0         ;COUNT THE BITS IN AC3
24                                                ;AND ADD RESULT TO AC0
25
26 01103'126610           COB     AC1,AC1         ;COUNT THE BITS IN AC1
27                                                ;AND ADD RESULT TO AC0
28                                                ;CHANGING ITS INITIAL VALUE
```

```
!0066 HANDO
01                ;                              LRB
02                ;                  LOCATE AND RESET LEAD BIT
                  ;                  -       -       -
                  ;                  ------------------------

06                ;        THIS INSTRUCTION FUNCTIONS IDENTICALLY TO LOB EXCEPT THA
07                ;                THE LEAD BIT ('1') IS SET TO ZERO.  THE ONE
08                ;                EXCEPTION IS THAT IF ACS & ACD ARE THE SAME,
09                ;                ONLY THE LEAD BIT IS RESET, NO COUNT IS TAKEN.
10
11                ;        FORMAT:
12                ;                LRB     ACS,ACD
13
14                ;        WHERE:
15                ;                ACS     IS THE SOURCE ACCUMULATOR TO BE SCANNED
16                ;                ACD     IS THE DESTINATION ACCUMULATOR TO BE ADD
17
18                ;        EXAMPLES:
19
20 01104'132510          LRB     AC1,AC2         ;AC1 IS SCANNED AND RESET
21                                               ;THE COUNT IS ADDED TO AC2
22
23 01105'116510          LRB     AC0,AC3         ;AC0 IS SCANNED AND RESET
24                                               ;THE COUNT IS ADDED TO AC3
25
26 01106'126510          LRB     AC1,AC1         ;AC1 IS SCANNED AND RESET
27                                               ;NO COUNT IS ADDED TO AC1
```

```
!0067 HANDO
01                ;                                BAM
02                ;                        BLOCK ADD   AND MOVE
                  ;                        -     -         -
                  ;  Interrupt able        ------------------

06                ;       WORDS IN MEMORY ARE MOVED FROM ONE LOCATION TO ANOTHER A
07                ;               AN EXTREMELY FAST RATE WITH A CONSTANT FROM
08                ;               AC0 ADDED TO EACH WORD MOVED (IF DESIRED).
09                ;               UP TO 32,768 WORDS CAN BE MOVED IN ONE OPERATION
10                ;               ACCUMULATORS 0-3 ARE ALL USED AS FOLLOWS:
11
12                ;                       AC0     CONTAINS CONSTANT TO BE ADDED
13                ;                       AC1     NUMBER OF WORDS TO BE MOVED
14                ;                       AC2     SOURCE STARTING ADDRESS
15                ;                       AC3     DESTINATION STARTING ADDRESS
16
17                ;               IF AC1 IS GREATER THAN 32,768 (100000 OCTAL)
18                ;               THE MOVE IS ABORTED AND NO ACCUMULATORS
19                ;               ARE MODIFIED.  FOR EACH WORD MOVED AC1 IS
20                ;               DECREMENTED BY 1 AND AC2 & AC3 ARE INCREMENTED
21                ;               BY 1.  AT THE END OF THE MOVE AC1=0 AND AC2 & AC3
22                ;               POINT TO THE WORD FOLLOWING THEIR RESPECTIVE FIELD
23                ;               ADDRESSES IN AC2 & AC3 CAN BE INDIRECT.
24
25                ;       FORMAT:
26                ;               BAM
27
28                ;       EXAMPLES:       SELF-EXPLANETORY

   01107'113710             BAM
30.
32
33                ;       PROGRAM EXAMPLE:
34
35 01110'162070                   ELEF    AC0,CST60       ;PUT OCTAL 60 FOR AC0 AD -
36       000060
37 01112'166070                   ELEF    AC1,CST10       ;MOVE LENGTH = 10
38       000012
39 01114'172470                   ELEF    AC2,NMTAB       ;POINT TO BEGINING OF IN
40       000004
41 01116'176470                   ELEF    AC3,OCTAB       ;POINT TO BEGINING OF OU
42       000014
43 01120'113710                   BAM                     ;DO THE MOVE
44
45 01121'000000 NMTAB:    0
46 01122'000001           1
47 01123'000002           2
48 01124'000003           3
49 01125'000004           4
50 01126'000005           5
51 01127'000006           6
52 01130'000007           7
53 01131'000010           8.
   01132'000011           9.
   01133'000012 OCTAB:    .BLK    10.
!        000060 CST60=     60
57       000012 CST10=     10.
```

```
!0068 HANDO
01                ;                                   BLM
02                ;                              BLOCK    MOVE
                  ;                  Interruptible --       -
                  ;       THE BLOCK MOVE IS IDENTICAL TO THE BLOCK ADD AND MOV
06                ;            EXCEPT THAT NO ADDITION IS DONE TO EACH WORD AND
07                ;            THEREFORE AC0 IS NOT USED.
08
09                ;       FORMAT:
10                ;               BLM
11
12                ;       EXAMPLES:       SELF-EXPLANETORY
13
14 01145'133710         BLM
```

```
!0069 HANDO
01                ;                                PSH
02                ;                    PUSH MULTIPLE ACCUMULATORS
                  ;                    - --
                  ;                    ------------------------
L
06                ;        ACCUMULATORS ARE PUSHED ONTO THE STACK STARTING WITH ACS
07                ;                AND ENDING WITH ACD IN ASCENDING ORDER.
08                ;                IF ACS & ACD ARE THE SAME, ONLY ONE ACCUMULATOR
FUU01146'000000                    IS ACTUALLY PUSHED.
10
11                ;        FORMAT:
12                ;                PSH     ACS,ACD
13
14                ;        WHERE:
15                ;                ACS     IS THE STARTING ACCUMULATOR TO PUSH
16                ;                ACD     IS THE  ENDING  ACCUMULATOR TO PUSH
17                ;                        AS FOLLOWS:
18                ;                                AC1,AC3     PUSHES AC1,AC2,AC3
19                ;                                AC2,AC0     PUSHES AC2,AC3,AC0
20                ;                                AC1,AC0     PUSHES AC1,AC2,AC3,AC0
21                ;                                AC0,AC0     PUSHES AC0
22
23                ;        EXAMPLES:
24
25 01147'123110          PSH     AC1,AC0          ;PUSH AC1 - AC0 ONTO STACK
26
27 01150'113110          PSH     AC0,AC2          ;PUSH AC0 - AC2 ONTO STACK
28
   01151'177110          PSH     AC3,AC3          ;PUSH AC3       ONTO STACK
```

PSH AC0, AC3
POP AC3, AC0

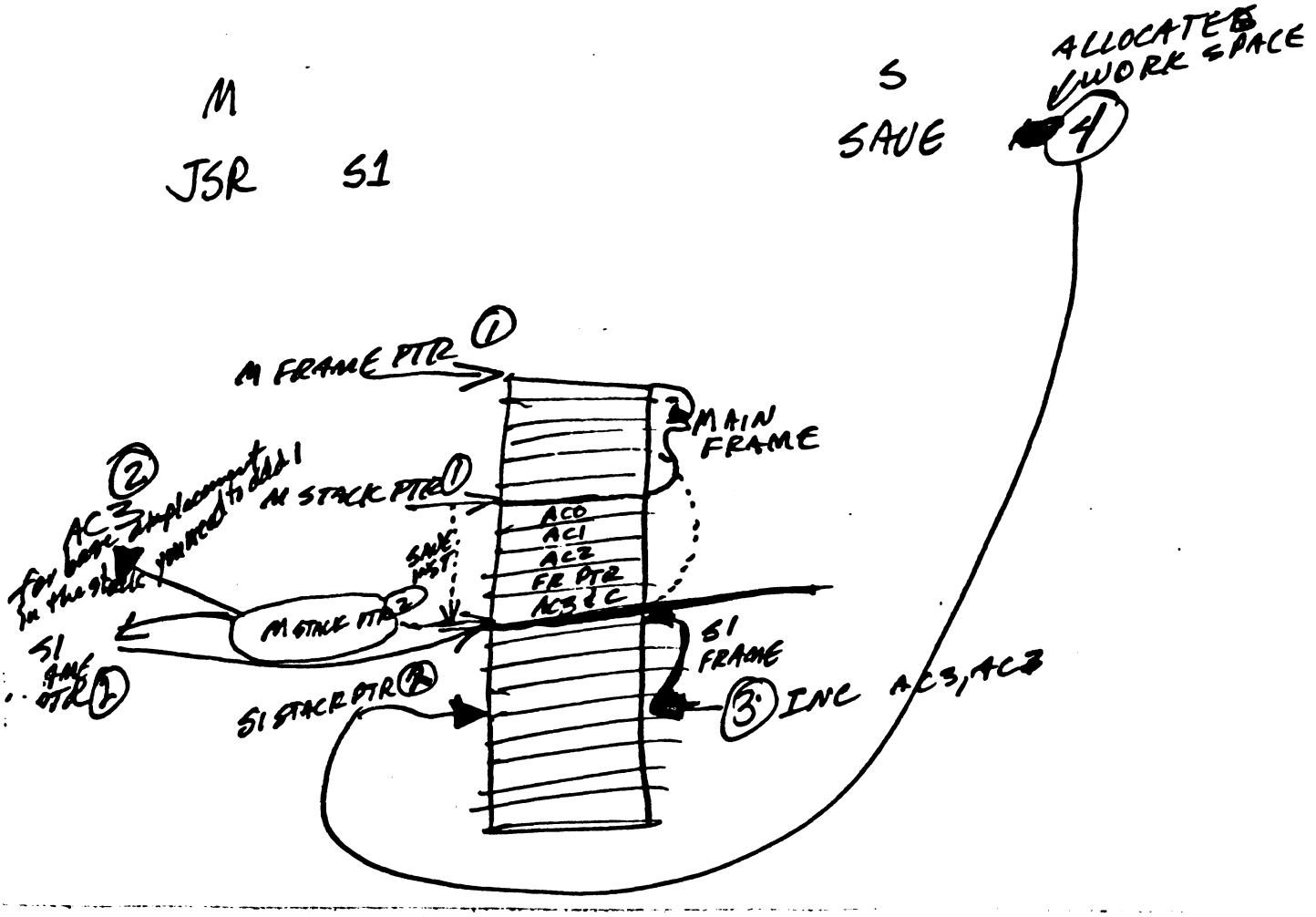
```
10070 HANDO
01                ;                                  POP
02                ;                  POP MULTIPLE  ACCUMULATORS
                  ;                  ---
                  ;                  ------------------------

06                ;        THE SET OF ACCUMULATORS STARTING WITH ACS AND ENDING WIT
07                ;                ACD ARE FILLED WITH WORDS POPPED FROM THE STACK
08                ;                IN DESCENDING ORDER.  IF ACS & ACD ARE THE SAME,
09                ;                ONLY ONE ACCUMULATOR IS POPPED.  ACCUMULATORS
10                ;                ARE SPECIFIED, AS IN THE PSH INSTRUCTION,
11                ;                EXCEPT THEY ARE POPPED IN DESCENDING ORDER.
12
13                ;        FORMAT:
14                ;                POP     ACS,ACD
15
16                ;        WHERE:
17                ;                ACS     IS THE STARTING ACCUMULATOR TO POP
18                ;                ACD     IS THE  ENDING  ACCUMULATOR TO POP
19
20                ;        EXAMPLES:
21
22 01152'137210           POP     AC1,AC3          ;AC1,AC0,AC3 ARE POPPED
23
24 01153'167210           POP     AC3,AC1          ;AC3,AC2,AC1 ARE POPPED
25
26 01154'147210           POP     AC2,AC1          ;AC2,AC1     ARE POPPED
27
28 01155'103210           POP     AC0,AC0          ;AC0         IS  POPPED
```

```
!0071 HANDO
01                ;                                PSHR
02                ;                     PUSH RETURN     ADDRESS
                  ;                     - -- -
                  ;                     ----------------------

06                ;          TWO IS ADDED TO THE PRESENT VALUE OF THE PROGRAM COUNTER
07                ;                AND IT IS PUSHED ONTO THE STACK.
08
09                ;          FORMAT:
10                ;                    PSHR
11
12                ;          EXAMPLES:        SELF-EXPLANITORY
13
14 01156°103710              PSHR
```

```
!0072 HANDO
01                ;                                   SAVE
02                ;                                   ----
                  ;                                   ----
                  ;
                  ;       A RETURN BLOCK (DESCRIBED BELOW) IS PUSHED ONTO THE STACK;
06                ;               THE VALUE OF THE STACK POINTER IS PLACED
07                ;               IN THE FRAME POINTER AND IN AC3.
08
09                ;       FORMAT:
10                ;               SAVE    I
11
12                ;       WHERE:  I       IS A 16-BIT UNSIGNED INTEGER ADDED
13                ;                       TO THE STACK POINTER.
14
15                ;       A RETURN BLOCK IS AS FOLLOWS
16
17                ;               WORD#   1 - AC0
18                ;               PUSHED  2 - AC1
19                ;                       3 - AC2
20                ;                       4 - FRAME POINTER BEFORE SAVE (AC3 on Return)
21                ;                       5 - BIT 0 = CARRY BIT
22                ;                           BITS 1-15 ARE BITS 1-15 OF AC3
23
24                ;       EXAMPLES:       SELF-EXPLANITORY
25
26 01157'163710           SAVE    0
27        000000
```



```
10073 HANDO
01                ;                              MSP
02                ;                       MODIFY   STACK POINTER
                  ;                         -        -     -
                  ;                       ----------------------

06                ;       THE CONTENTS OF THE SPECIFIED ACCUMULATOR ARE ADDED TO
07                ;             THE CONTENTS OF THE STACK POINTER AND THE RESULT
08                ;             COMPARED TO THE STACK LIMIT. IF GREATER THAN THE
09                ;             STACK LIMIT, A STACK PROTECTION FAULT IS PERFORM
10                ;             AS A RESULT, THE PC IN THE FAULT RETURN BLOCK
11                ;             IS THE ADDRESS OF THE MSP INSTRUCTION AND THE
12                ;             STACK POINTER IS LEFT UNCHANGED.
13
14                ;       FORMAT:
15                ;             MSP     AC
16
17                ;       WHERE:
18                ;             AC      IS THE ACC TO ADD TO THE STACK POINTER
19
20                ;       EXAMPLES:
21
22 01161'103370          MSP     AC0             ;ADD AC0 TO STACK POINTER
23
24 01162'107370          MSP     AC1             ;ADD AC1 TO STACK POINTER
```

Save the SP if you dont want it modified.

```
10074 HANDO
01                  ;                                   POPB
02                  ;                                 POP     BLOCK
                    ;                                 ---     -
                    ;                                 ---------------
06
07                  ;       FIVE WORDS ARE POPPED OFF THE STACK (AFTER BEING SAVED)
08                  ;               AND ARE PLACED IN THEIR APPROPRIATE DESTINATIONS
09                  ;               (SEE SAVE).
10
11                  ;       FORMAT:
12                  ;               POPB
13
14                  ;       EXAMPLES:       SELF-EXPLANITORY
15
16 01163'107710             POPB
```

DOES NOT RESTORE FRAME POINTER!

```
10075 HANDO
01                ;                                  RTN
02                ;                                 RETURN
                  ;                                  - - -
                  ;                                 ------
05
06
07                ;       THE CONTENTS OF THE FRAME POINTER ARE PLACED IN THE STAC
08                ;             POINTER AND A POPB INSTRUCTION IS EXECUTED.
09                ;             THE POPPED VALUE OF AC3 IS PLACED IN THE
10                ;             FRAME POINTER.
11
12                ;       FORMAT:
13                ;             RTN
14
15                ;       EXAMPLES:       SELF-EXPLANITORY
16
17 01164'127710           RTN
```

```
!0076 HANDO
01                ;                                  RSTR
02                ;                                 RESTORE
                  ;                                 - -- -
                  ;                                 ------

06                ;       NINE WORDS ARE POPPED OF THE STACK AND PLACED IN THEIR
07                ;               APPROPRIATE LOCATIONS.
08                ;       FOR MORE DETAILED EXPLANATION SEE THE ECLIPSE PROGRAMMER
09                ;               GUIDE - PAGE 3-26.
10
11                ;       FORMAT:
12                ;               RSTR
13
14                ;       EXAMPLES:       SELF-EXPLANITORY
15
16 01165'167710           RSTR
```

```
10077 HANDO
01                ;                              SGT
02                ;                    SKIP IF ACS GREATER THAN ACD
                  ;                    -           -       -
                  ;                    ------------------------

06                ;        THE CONTENTS OF ACS AND ACD ARE COMPARED ALGEBRAICLY AS
07                ;                SIGNED 2'S COMPLIMENT INTEGERS.  IF ACS IS GREAT
08                ;                THAN ACD, THE NEXT SEQUENTIAL WORD IS SKIPPED;
09                ;                OTHERWISE, CONTROL PASSES TO THE NSW.
10                ;                ACS AND ACD REMAIN UNCHANGED.
11
12                ;        FORMAT:
13                ;                SGT     ACS,ACD
14
15                ;        WHERE:
16                ;                ACS     IS THE SOURCE ACCUMULATOR FOR COMPARISON
17                ;                ACD     IS THE DESTINATION ACCUMULATOR
18                ;                                FOR COMPARISON.
19
20                ;        EXAMPLES:
21
22 01166'131010          SGT     AC1,AC2         ;AC1 IS COMPARED AGAINST AC2
23
24 01167'161010          SGT     AC3,AC0         ;AC3 IS COMPARED AGAINST AC0
25
26 01170'125010          SGT     AC1,AC1         ;AC1 IS COMPARED AGAINST AC1
27                                               ; (FRIVOLOUS)
```

```
!0078 HANDO
01                ;                              SGE
02                ;              SKIP IF ACS GREATER THAN OR EQUAL ACD
                  ;              -           -                -
                  ;              ------------------------------------
06                ;       THE SIGNED 2'S COMPL. NUMBERS IN ACS AND ACD ARE COMPARE
07                ;              IF ACS IS GREATER THAN OR EQUAL TO ACD, THE NEXT
08                ;              SEQUENTIAL WORD IS SKIPPED; ELSE CONTROL PASSES
09                ;              TO THE NSW.
10
11                ;       FORMAT:
12                ;              SGE     ACS,ACD
13
14                ;       WHERE:
15                ;              ACS     IS THE SOURCE ACCUMULATOR FOR COMPARISON
16                ;              ACD     IS THE DESTINATION ACCUMULATOR
17                ;                              FOR COMPARISON
18
19                ;       EXAMPLES:
20
21 01171'131110           SGE     AC1,AC2         ;AC1 IS COMPARED AGAINST AC2
22
23 01172'141110           SGE     AC2,AC0         ;AC2 IS COMPARED AGAINST AC0
24
25 01173'175110           SGE     AC3,AC3         ;AC3 IS COMPARED AGAINST AC3
26                                                ; (FRIVOLOUS)
```

```
!0079 HANDO
01                ;                                CLM
02                ;                          COMPARE   TO LIMITS
                  ;                          -         - -
                  ;                          ------------------

06                ;        THE SIGNED 2'S COMPLIMENT NUMBER IN ACS IS COMPARED WITH
07                ;                LIMITS "L" & "H".  IF ACS IS GE "L" AND LE "H",
08                ;                THE NEXT SEQUENTIAL WORD (NSW) IS SKIPPED.  IF
09                ;                LT "L" OR GT "H" THE NSW IS EXECUTED.
10                ;                IF ACS & ACD ARE DIFFERENT THE ADDRESS OF LIMIT
11                ;                VALUE "L" WILL BE IN ACD AND LIMIT
12                ;                VALUE "H" WILL BE IN THE WORD FOLLOWING "L".
13                ;                IF ACS & ACD ARE THE SAME, THE LIMIT VALUES "L"
14                ;                "H" WILL BE FOUND IN THE TWO WORDS FOLLOWING THE
15                ;                CLM INST. RESPECTIVELY: NSW WILL THEN BE THE
16                ;                3RD WORD FOLLOWING THE CLM.
17
18                ;        FORMAT:
19                ;                CLM      ACS,ACD
20
21                ;        WHERE:
22                ;                ACS      IS THE ACCUMULATOR TO BE COMPARED
23                ;                ACD      IF DIFFERENT FROM ACS CONTAINS THE ADDR
24                ;                         OF THE LIMITS VALUES.
25
26                ;        EXAMPLES:
27
28 01174'132370            CLM      AC1,AC2          ;COMPARE AC1 AGAINST THE
                                                     ;LIMITS ADDRESSED BY AC2
31 01175'102370            CLM      AC0,AC0          ;COMPARE AC0 AGAINST THE
32                                                   ;CONTAINED IN THE NEXT
33                                                   ;TWO WORDS
34 01176'000050 LIMT1:     50
35 01177'000100 LIMT2:     100
```

```
!0080 MANDO
01                ;                                XCT
02                ;                              EXECUTE
                  ;                              - - -
                  ;                              -------

06                ;       THE INSTRUCTION CONTAINED IN THE SPECIFIED ACCUMULATOR I
07                ;               EXECUTED AS THO IT WERE IN MAIN MEMORY AT THE
08                ;               LOCATION OF THE XCT INST.  THE ACCUMULATOR WAS
09                ;               PROBABLY LOADED OR MODIFIED PRIOR TO THE EXECUTI
10                ;               OF THIS INSTRUCTION.  IF THE ACC CONTAINS AN XCT
11                ;               INSTRUCTION, A 1 INSTRUCTION LOOP IS GENERATED.
12                ;               THIS COULD BE USEFUL AS I/O WAITS BECAUSE THE
13                ;               INSTRUCTION IS INTERRUPTABLE.
14
15                ;       FORMAT:
16                ;               XCT     AC
17
18                ;       WHERE:
19                ;                       AC      IS THE ACCUMULATOR CONTAINING TH
20                ;                               INSTRUCTION TO BE EXECUTED
21
22                ;       EXAMPLES:
23
24 01200'127370           XCT     AC1             ;EXECUTE THE CONTENTS OF AC1
25
26 01201'137370           XCT     AC3             ;EXECUTE THE CONTENTS OF AC3
```

```
!0081 HANDO
01                  ;                                   PSHJ
02                  ;                                 PUSH     JUMP
                    ;                                 - --     -
                    ;                                 ------------
                    ;
                    ;       THE ADDRESS OF THE NEXT SEQUENTIAL INSTRUCTION (NSI) IS
07                  ;            PUSHED ONTO THE STACK AND A JUMP IS MADE.
08                                                                  (JUMP
09                  ;       FORMAT:
10                  ;                PSHJ     ADDR
11                  ;                     OR
12                  ;                PSHJ     D,X
13
14                  ;       WHERE:
15                  ;                ADDR     IS AN ADDRESS LABEL IN THE PROGRAM
16                  ;                D        IS THE DISPLACEMENT  (0-17777)
17                  ;                X        IS THE INDEX MODE    (0,1,2,3)
18
19                  ;       EXAMPLES:
20
21 01202'102670             PSHJ     ADDR1           ;PUSH NSI & JUMP TO ADDR1
22         077247
23
24 01204'102670             PSHJ     5,1             ;PUSH NSI & JUMP TO 5+PC
25         000005
26
27 01206'103270             PSHJ     10,2            ;PUSH NSI & JUMP TO 10+AC2
28         000010

?  01210'103670             PSHJ     15,3            ;PUSH NSI & JUMP TO 15+AC3
-          000015
3
33 01212'102670             PSHJ@    ADDR1           ;PUSH NSI & JUMP THRU ADDR1
34         177237
35
36 01214'103270             PSHJ@    25,2            ;PUSH NSI & JUMP THRU 25+AC2
37         100025
```

```
!0082 HANDO
01                ;                                  POPJ
02                ;                        POP  PC      AND JUMP
                  ;                        ---                -
                  ;                        ___________________

06                ;       THE TOP WORD OF THE STACK IS POPPED AND PLACED INTO THE
07                ;               EFFECTING A JUMP TO THE ADDRESS THAT WAS JUST
08                ;               POPPED OFF THE STACK.
09
10                ;       FORMAT:
11                ;               POPJ
12
13                ;       EXAMPLES:       SELF-EXPLANITORY
14
15 01216'117710           POPJ
```

```
10083 HANDO
01              ;*****************************************************************
02              ;* THE FOLLOWING TABLE IS TO BE USED BY RDOS USERS RUNNING ON  *
                ;* AN AOS SYSTEM. BELOW IS DEFINED THE CONVERSION BETWEEN AOS  *
                ;* SPEED AND RDOS TEXT EDITOR. ALSO OUTLINED ARE THE BASIC     *
                ;* CLI COMMANDS WHICH WILL BE NEEDED.                          *
06              ;*****************************************************************
07
08              ;       TEXT EDITOR                     AOS SPEED
09              ;       -----------                     ---------
10
11              ;           UY                            FO
12              ;           UE                            FU
13              ;           S                             S
14              ;           C                             C
15              ;           I                             I
16              ;           B                             J
17              ;         (N)T                          (N)T      (N) = SOME NUMBE
18              ;           T                            #T
19              ;          1T                             T
20              ;         (N)L                          (N)L      (N) = SOME NUMBE
21              ;         (N)K                          (N)K      (N) = SOME NUMBE
22              ;           H                             H
23              ;           US                            FB
24              ;           GW                            FW
25              ;           GR                            FR
26              ;           $$ (ESC ESC)                  CTRL D
27
28              ;NOTE:  UNDER RDOS, THE SYSTEM ALWAYS APPENDS A NEW LINE TO ALL
                ;       CARRIAGE RETURNS, UNDER AOS NO APPENDING IS PERFORMED,
                ;       THEREFORE IT IS NECESSARY TO END EACH LINE OF YOUR TEXT
3.              ;       WITH A NEW LINE CHARACTER RATHER THAN A CARRIAGE RETURN.
32              ;-----------------------------------------------------------------
33
34              ;       RDOS BASIC CLI COMMANDS         AOS BASIC CLI COMMANDS
35
36              ;            LIST/A/E/S                       F/AS/S
37              ;            DELETE/V/C                       DEL/V/C
38              ;            TYPE                             TYPE
39              ;            PRINT                            QPRINT
40              ;-----------------------------------------------------------------
41
42              ;       RDOS PROGRAM ASSEMBLY           AOS PROGRAM ASSEMBLY
43              ;            R
44              ;            MAC FILENAME                ) X MASM FILENAME
45
46              ;       RDOS PROGRAM LOADING            AOS PROGRAM BINDING
47              ;            R
48              ;            RLDR FILENAME               ) X BIND FILENAME
49
50              ;       RDOS FILE EXECUTION             AOS FILE EXECUTION
51              ;            R
52              ;            FILENAME (CR)               ) X FILENAME (NL)
```

```
!0084 HANDO
01                  ;       ******************************************
02                  ;       * THE FOLLOWING MACROS SHOW TWO WAYS OF *
                    ;       * WRITING A FILL ROUTINE. AC0 CONTAINS  *
                    ;       * THE FILLING VALUE, AC1 CONTAINS THE # *
                    ;       * OF WORDS TO FILL, AND AC2 IS THE STA- *
06                  ;       * RTING ADDRESS TO BEGIN FILLING.       *
07                  ;       ******************************************
08
09                          .MACRO  ZFILL
10                          SBI     1,2         ; DECREMENT START BY 1
11                          STA     2,20        ; STORE STARTING ADDRESS
12                          STA     1,16        ; STORE COUNTER
13                          STA@    0,20        ; FILL THE LOCATION
14                          DSZ     16          ; DECREMENT COUNTER
15                          JMP     .-2         ; NOT FINISHED, LOOP BACK
16                          %
17
18
19                          .MACRO  FILL
20                          STA     0,0,2       ; STORE FILLER VALUE
21                          MOV     2,3         ; SET UP DESTINATION MINUS ONE
22                          ADI     1,3         ; ADD ONE TO DESTINATION
23                          BLM                 ; MOVE IT !!!!!
24                          %
25
26                  ;       NOTE: THE FIRST MACRO TAKES 2 PAGE 0 LOCATIONS AND
27                  ;             6 INSTRUCTIONS WHERE AS THE SECOND ONE ONLY
28                  ;             TAKES 4 INSTRUCTIONS AND NO PAGE 0 LOCATIONS.

                    ;       NOTE:  FOLLOWING ARE SOME EXAMPLES OF MACRO EXPANSIO
32
33
34                  FILL1:  ZFILL
35 01217'110110             SBI     1,2         ; DECREMENT START BY 1
36 01220'050020             STA     2,20        ; STORE STARTING ADDRESS
37 01221'044016             STA     1,16        ; STORE COUNTER
38 01222'042020             STA@    0,20        ; FILL THE LOCATION
39 01223'014016             DSZ     16          ; DECREMENT COUNTER
40 01224'000776             JMP     .-2         ; NOT FINISHED, LOOP BACK
41
42                  FILL2:  FILL
43 01225'041000             STA     0,0,2       ; STORE FILLER VALUE
44 01226'155000             MOV     2,3         ; SET UP DESTINATION MINUS ONE
45 01227'114010             ADI     1,3         ; ADD ONE TO DESTINATION
46 01230'133710             BLM                 ; MOVE IT !!!!!
47
48        000001            .NOMAC  1           ;INHIBIT MACRO EXPANSION
49
50                  FILL3:  ZFILL
 FU01237'000000     FIL3A   JMP     .+1         ;NOTICE THE ADDRESS OF THIS INST.
52
53                  FILL4:  FILL
 U01244'000000      FIL4A   JMP     .+1         ;NOTICE THE ADDRESS OF THIS INST.

        000000              .NOMAC  0           ;REENABLE MACRO EXPANSION
```

```
10085 HANDO
01                ;       ******************************************
02                ;       * THE FOLLOWING MACRO PERFORMS A BLOCK *
                  ;       * MOVEMENT OF CHARACTERS FROM ONE AREA *
                  ;       * OF MEMORY TO ANOTHER. THE FIRST ARG. *
                  ;       * IS THE STARTING LOCATION OF SOURCE,  *
06                ;       * THE SECOND ARG. IS THE STARTING LOC- *
07                ;       * ATION OF DESTINATION, AND THE THIRD  *
08                ;       * ARG. IS THE NUMBER OF WORDS TO MOVE. *
09                ;       ******************************************
10
11                        .MACRO  BLOCK
12                        ELEF    0,^1      ; GET SOURCE ADDRESS
13                        ELEF    1,^2      ; GET DESTINATION ADDRESS
14                        ELEF    2,^3      ; GET NUMBER OF WORDS TO MOVE
15                        SBI     1,0       ; DECREMENT SOURCE BY ONE (AUTO INC.)
16                        SBI     1,1       ; DECREMENT DESTINATION BY ONE (AUTO INC
17                        STA     2,16      ; STORE THE COUNTER
18                        STA     0,20      ; STORE SOURCE ADDRESS
19                        STA     1,21      ; STORE DESTINATION ADDRESS
20                        LDA@    0,20      ; GET A SOURCE WORD
21                        STA@    0,21      ; MOVE WORD TO DESTINATION
22                        DSZ     16        ; DECREMENT COUNTER
23                        JMP     .-3       ; NOT FINISHED, LOOP BACK
24                        %
25                ;       NOTE: THE FOLLOWING ILLUSTRATES PASSING OF PARAMETERS
26                ;             TO A MACRO
27                        BLOCK   TABL1,TABL2,10
28 01245'162470           ELEF    0,TABL1 ; GET SOURCE ADDRESS
       000016
   01247'166470           ELEF    1,TABL2 ; GET DESTINATION ADDRESS
       000026
32 01251'172070           ELEF    2,10    ; GET NUMBER OF WORDS TO MOVE
33         000010
34 01253'100110           SBI     1,0     ; DECREMENT SOURCE BY ONE (AUTO INC.)
35 01254'104110           SBI     1,1     ; DECREMENT DESTINATION BY ONE (AUTO INC
36 01255'050016           STA     2,16    ; STORE THE COUNTER
37 01256'040020           STA     0,20    ; STORE SOURCE ADDRESS
38 01257'044021           STA     1,21    ; STORE DESTINATION ADDRESS
39 01260'022020           LDA@    0,20    ; GET A SOURCE WORD
40 01261'042021           STA@    0,21    ; MOVE WORD TO DESTINATION
41 01262'014016           DSZ     16      ; DECREMENT COUNTER
42 01263'000775           JMP     .-3     ; NOT FINISHED, LOOP BACK
43
44 01264'000012 TABL1:    .BLK    10.
45 01276'000012 TABL2:    .BLK    10.
```

```
        LICENSED MATERIAL - PROPERTY OF DATA GENERAL CORPORATION
10086 HANDO
01              ;MAC.PS - ECLIPSE RDOS SYSTEMS
02
                ;THE INITIAL ECLIPSE RDOS STARTER TAPE YOU RECEIVE FROM DATA
                ;GENERAL CONTAINS THE MACROASSEMBLER ALONG WITH ALL NECESSAR
                ;SUPPORTING FILES FOR THE MACROASSEMBLER.
06
07              ;BEFORE THE MACROASSEMBLER CAN BE USED, A FILE NAMED "MAC.PS"
08              ;MUST BE CREATED FROM OTHER SOURCE FILES AS DEFINED BELOW.
09              ;THIS FILE "MAC.PS" IS THE MACROASSEMBLER'S PERMANENT SYMBOL
10              ;FILE WHICH WILL CONTAIN ALL SYMBOLS YOU NORMALLY USE WHEN
11              ;CODING ASSEMBLER PROGRAMS (I.E. THE LDA, STA, ... INSTRUCTIONS,
12              ;SYSTEM CALLS - .PCHAR .... ETC.).
13
14              ;THE FOLLOWING FILES ARE USED DEPENDING ON YOUR APPLICATION
15              ;AND NEEDS TO BUILD MAC.PS:
16
17              ;NBID.SR                    NOVA BASIC INSTRUCTION DEFINITION
18
19              ;NCID.SR                    ECLIPSE COMMERCIAL INSTRUCTION DEFINITIO
20
21              ;NEID.SR                    NOVA EXTENDED INSTRUCTIONS FOR ECLIPSES
22
23              ;NFPID.SR       HARDWARE FLOATING POINT INSTRUCTION DEFINITION
24
25              ;OSID.SR                    SYSTEM CALLS DEFINITION
26
27              ;PARU.SR                    USER PARAMETER DEFINITION
28
                ;ARDOS.SR       MAPPED ECLIPSE SYSTEM

                ;ZRDOS.SR       BIG MAPPED ECLIPSE SYSTEM
32
33              ;TO BUILD MAC.PS WE USE THE FOLLOWING COMMAND:
34
35              ;MAC/S NBID OSID NEID [NCID] [NFPID] [PARU] SYSFILE
36
37              ;WHERE SYSFILE IS EITHER ARDOS.SR OR ZRDOS.SR

**00030 TOTAL ERRORS, 00000 PASS 1 ERRORS
```

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| AC0 | 000000 | | 2/14 | 8/26 | 8/34 | 9/26 | 9/34 | 14/20 | 14/23 |
| | | | 15/16 | 16/16 | 17/16 | 18/15 | 19/15 | 20/14 | 21/13 |
| | | | 22/11 | 22/12 | 22/26 | 22/27 | 22/41 | 22/42 | 32/35 |
| | | | 32/46 | 35/26 | 35/35 | 36/21 | 37/20 | 38/23 | 44/22 |
| | | | 45/19 | 46/18 | 47/20 | 48/20 | 49/20 | 50/21 | 51/26 |
| | | | 52/29 | 52/33 | 53/27 | 54/29 | 55/24 | 56/28 | 57/22 |
| | | | 58/22 | 59/25 | 59/27 | 60/20 | 60/24 | 61/22 | 62/24 |
| | | | 63/20 | 63/22 | 64/24 | 65/23 | 66/23 | 67/35 | 69/25 |
| | | | 69/27 | 70/28 | 73/22 | 77/24 | 78/23 | 79/31 | |
| AC1 | 000001 | | 2/15 | 8/28 | 8/32 | 9/28 | 9/32 | 14/15 | 15/16 |
| | | | 16/14 | 17/18 | 18/17 | 19/18 | 20/14 | 21/13 | 21/19 |
| | | | 22/09 | 22/14 | 22/21 | 22/22 | 22/23 | 22/24 | 22/36 |
| | | | 22/37 | 22/38 | 22/39 | 32/37 | 35/28 | 44/24 | 45/21 |
| | | | 46/18 | 46/20 | 47/22 | 48/20 | 48/22 | 49/22 | 50/21 |
| | | | 50/23 | 51/26 | 51/30 | 52/27 | 52/29 | 53/23 | 54/23 |
| | | | 55/18 | 56/22 | 57/19 | 58/19 | 58/25 | 59/21 | 60/18 |
| | | | 60/22 | 61/20 | 61/24 | 62/20 | 62/26 | 63/18 | 64/21 |
| | | | 64/27 | 65/20 | 65/26 | 66/20 | 66/26 | 67/37 | 69/25 |
| | | | 70/22 | 70/24 | 70/26 | 73/24 | 77/22 | 77/26 | 78/21 |
| | | | 79/28 | 80/24 | | | | | |
| AC2 | 000002 | | 2/16 | 8/30 | 9/30 | 14/15 | 14/17 | 14/23 | 15/18 |
| | | | 15/27 | 16/14 | 17/18 | 18/17 | 19/18 | 19/21 | 20/16 |
| | | | 21/15 | 22/07 | 22/11 | 22/13 | 32/39 | 32/43 | 35/32 |
| | | | 36/23 | 37/22 | 38/27 | 44/26 | 46/20 | 46/22 | 48/22 |
| | | | 48/24 | 50/23 | 50/25 | 51/28 | 52/27 | 52/31 | 54/25 |
| | | | 55/20 | 56/24 | 57/19 | 57/25 | 58/19 | 58/25 | 59/21 |
| | | | 59/23 | 59/25 | 60/18 | 60/20 | 61/20 | 61/24 | 62/20 |
| | | | 62/22 | 63/18 | 64/21 | 65/20 | 66/20 | 67/39 | 69/27 |
| | | | 70/26 | 77/22 | 78/21 | 78/23 | 79/28 | | |
| | 000003 | | 2/17 | 14/17 | 14/20 | 15/23 | 15/27 | 18/20 | 19/21 |
| | | | 21/19 | 22/08 | 22/12 | 22/14 | 32/41 | 35/30 | 36/25 |
| | | | 37/24 | 38/25 | 45/23 | 46/22 | 47/24 | 48/24 | 49/24 |
| | | | 50/25 | 51/28 | 51/30 | 52/31 | 52/33 | 53/25 | 54/27 |
| | | | 55/22 | 56/26 | 57/22 | 58/22 | 59/23 | 60/22 | 61/22 |
| | | | 62/22 | 62/24 | 63/20 | 64/24 | 65/23 | 66/23 | 67/41 |
| | | | 69/29 | 70/22 | 70/24 | 77/24 | 78/25 | 80/26 | |
| ACTUA | 000000U | | 69/09 | | | | | | |
| ADDR1 | 000452' | | 6/17 | 8/26 | 8/34 | 9/26 | 9/34 | 10/22 | 10/30 |
| | | | 11/22 | 11/30 | 12/21 | 12/29 | 13/23 | 13/31 | 32/35 |
| | | | 32/37 | 32/46 | 32/50 | 32/52 | 32/55 | 32/57 | 33/03 |
| | | | 35/26 | 35/35 | 81/21 | 81/33 | | | |
| ADDR1 | 000315' | | 4/21 | 6/13 | | | | | |
| ADDR3 | 000317' | | 6/14 | | | | | | |
| AS | 000000U | | 28/07 | 29/08 | | | | | |
| BLAWK | 000450' | | 5/57 | | | | | | |
| BLOCK | 001645 | MC | 85/11 | 85/27 | | | | | |
| BST1 | 000001 | | 36/25 | 36/27 | 38/27 | | | | |
| BST10 | 000012 | | 67/37 | 67/57 | | | | | |
| BST60 | 000060 | | 67/35 | 67/56 | | | | | |
| | 000000U | | 32/16 | 32/20 | | | | | |
| BUIL3A | 000000U | | 84/51 | | | | | | |
| BUIL4A | 000000U | | 84/54 | | | | | | |
| BUILE. | 000000U | | 28/07 | 29/08 | | | | | |
| BUIL | 001624 | MC | 84/19 | 84/42 | 84/53 | | | | |
| | 001217' | | 84/34 | | | | | | |
| BUL | 001225' | | 84/42 | | | | | | |
| BUILL3 | 001231' | | 84/50 | | | | | | |
| BUILL4 | 001240' | | 84/53 | | | | | | |
| BUNPTB | 000601' | | 22/55 | 22/57 | 22/60 | 23/29 | | | |

```
INPUT 000020        22/37    22/41    22/52
  TAB 000575'       22/07    22/21    22/55
  TBA 000577'       22/36    22/57
      000000U       69/09
LIMT1 001176'       79/34
LIMT2 001177'       79/35
LOOP1 000543'       22/10    22/16
LOOP2 000555'       22/25    22/31
LOOP3 000567'       22/40    22/44
MVCNT 000662'       22/15    22/30    22/43    23/29
NMTAB 001121'       67/39    67/45
OCT70 000070        45/23    45/27    47/24    49/24
OCTAB 001133'       67/41.   67/55
OUTAB 000576'       22/08    22/23    22/56
OUTBA 000600'       22/38    22/58
OUTPT 000030        22/39    22/42    22/53
OUTTB 000634'       22/56    22/58    23/27    23/29
PUSHE 000000U       69/09
SAME  000000U       28/07    29/08
TABL1 001264'       85/28    85/44
TABL2 001276'       85/30    85/45
TEXT  000427'        5/56
THAT  000000U       28/07    29/08
THE   000000U       28/07    29/08
THE.0 000000U       29/08
THE.R 000000U       28/07
WORK1 000573'       22/22    22/26    22/28    22/48
  RK2 000574'       22/24    22/27    22/29    22/49
      000000U       32/16    32/20
  RO  000000        22/11    22/12    22/51
ZFILL 001574  MC    84/09    84/34    84/50
.A    000452'        6/15     6/16
```

```
;
; COPYRIGHT (C) DATA GENERAL CORPORATION 1977, 1978
; ALL RIGHTS RESERVED.
; LICENSED MATERIAL-PROPERTY OF DATA GENERAL CORPORATION.
;

        .TITLE  LITMACS

;==================================
; FOR RUNNING WITH MAC VERSION 4.00
;==================================

; LITMACS.SR  -  MACROS FOR NON-ZREL LITERAL FACILITY
;
;       THIS MODULE CONTAINS MACROS WHICH PROVIDE A LITERAL FACILITY
; DIFFERENT FROM MAC'S IN THAT THE LITERAL WORDS ARE PLACED WHERE
; THE USER SPECIFIES, INSTEAD OF IN ZREL.  PC-RELATIVE ADDRESSING, INSTEAD OF
; PAGE ZERO ADDRESSING, IS USED TO ACCESS THE LITERAL WORDS.  THUS, NREL
; OR ABSOLUTE PROGRAMS CAN USE LITERALS WITHOUT USING UP ZREL STORAGE.
;
;       IN ADDITION, THIS LITERAL FACILITY IS MORE GENERAL: IT ACCEPTS
; EXTERNALS, INSTRUCTIONS, PSEUDO-OPS, AND MACROS, IN ADDITION TO CONSTANTS,
; LOCAL SYMBOLS, AND EXPRESSIONS OF THESE.
;
;       USE OF A LITERAL IS SPECIFIED WITH THE "LIT" MACRO CALL.
; CALLS ARE OF THE FORM:
;
;               LIT[<EXPRESSION>]
```

```
;
; WHERE <EXPRESSION> IS ONE OR MORE MACRO ARGUMENTS MAKING UP
; A VALID MAC EXPRESSION.  "LIT[<EXPRESSION>]" YIELDS THE ADDRESS OF
; A LOCALLY-ADDRESSABLE MEMORY LOCATION (OR LOCATIONS) CONTAINING
  THE VALUE FOR <EXPRESSION>.  FOR EXAMPLE:

        LDA     0,LIT[3]            ; LOAD AC0 WITH 3
;       LDA     0,LIT[TABLE]        ; LOAD AC0 WITH THE ADDRESS "TABLE".
;       JMP     @LIT[LABEL]         ; JUMP TO ADDRESS "LABEL"
;       LIT[10.]                    ; GENERATE THE ADDRESS OF A DECIMAL 10
;
;       NOTE!!  CALL TO "LIT" MUST BE THE LAST THING ON A LINE.
;               ALL ELSE ON THE LINE IS LOST INCLUDING THE COMMENT.
;               (THIS IS DUE TO CROCK IN MAC.)
;
;       LITERALS ARE PLACED IN CORE WITH THE "LPOOL" MACRO.  LPOOL
; GENERATES IN-LINE LITERAL WORDS FOR ALL LITERALS USED SINCE THE LAST
; PREVIOUS LPOOL.  YOU MUST SAY "LPOOL" ONLY WHERE CONTINUITY OF INSTRUCTIONS
; IS NOT REQUIRED.  YOU MUST SPRINKLE ENOUGH "LPOOL"'S THROUGHOUT YOUR
; PROGRAM TO AVOID ADDRESSING ERRORS.  (IF YOU DO GET AN ADDRESSING ERROR,
; JUST INSERT ANOTHER LPOOL AT AN APPROPRIATE PLACE.)  HOWEVER, DON'T
; OVERDO THE "LPOOL"'S; THE FEWER THE "LPOOL"'S, THE MORE LIKELY IT
; IS THAT YOUR LITERAL USES WILL BE COMPLETELY OPTIMIZED.
;
;       MULTIPLE REFERENCES TO THE SAME LITERAL VALUE ARE OPTIMIZED.
; A SUBSEQUENT REFERENCE TO THE SAME LITERAL VALUE WILL USE AN
; ALREADY GENERATED LITERAL IF IT IS WITHIN ADDRESSING RANGE.  THE
; LIMITATIONS TO THE OPTIMIZATION ARE AS FOLLOWS:
;
;       1) ALL FORWARD REFERENCES (I.E., SYMBOLS USED IN "LIT" BEFORE
;          THEY ARE DEFINED) WILL CAUSE SEPARATE LITERALS TO BE GENERATED.
;          (PLEASE NOTE THE RESTRICTIONS ON FORWARD REFERENCES BELOW.)
```

```
;       2) FOR ASSEMBLY EFFICIENCY, LPOOL ASSUMES THAT THE USER'S PROGRAM
;          USES AND DUMPS LITERALS IN ORDER, THAT IS, THAT BACKWARD
;          .LOC'S ARE NOT DONE.  IF THIS IS NOT TRUE, LITERALS WILL
;          STILL WORK, BUT MORE LITERALS THAN ARE REALLY NECESSARY
;          MAY BE GENERATED.

;       ALL CALLS TO "LIT" AND "LPOOL" MUST BE IN THE SAME ADDRESS
; SPACE (I.E., ALL DONE IN "NREL" OR ALL DONE IN "ZREL" OR ALL ABSOLUTE).
;
;       "LIT" CAN HANDLE NEARLY ANY EXPRESSION.  THE EXCEPTIONS ARE THAT
; IN PASSING DOWN THE ARGUMENT(S) TO "LIT", COMMAS, SPACES, AND TABS CHANGE
; TO BECOME A SINGLE SPACE (ACCORDING TO THE RULES FOR MACRO ARGUMENTS
; AS DESCRIBED IN THE MAC MANUAL).  THIS SOMEWHAT FOULS UP THE USE OF
; 'XXXX' AND "X BUT NOT VERY MUCH.  THE ARGUMENT TO "LIT" CAN ALSO HAVE
; MULTIPLE ARGUMENTS SUCH AS:
;
;       LDA     0,LIT[SUB 0,0]  ; LOAD AC0 WITH A "SUB 0,0" INSTRUCTION
;
;       IN RARE CASES IT MAY BE NECESSARY TO TELL "LIT" NOT TO TRY
; TO OPTIMIZE A LITERAL USE.  YOU CAN DO THIS BY PASSING TO LIT A NULL FIRST
; ARGUMENT, SINCE LIT WILL NOT TRY TO OPTIMIZE ANY LITERAL USE WHICH INVOLVES
; MULTIPLE ARGUMENTS.  PASSING DOWN A NULL FIRST ARGUMENT CAN BE DONE BY
; PUTTING A COMMA IMMEDIATELY FOLLOWING THE LEFT BRACKET.  (NOTE THAT A
; SPACE ALSO WORKS, BUT DOESN'T REMIND YOU AS WELL THAT THE FIRST ARGUMENT
; IS NULL!)
;
;       LITERAL USES WHICH ARE FORWARD REFERENCES, EXTERNALS, INSTRUCTIONS,
; PSEUDO-OPS, AND MACROS ARE AUTOMATICALLY RECOGNIZED BY LIT AS NON-
; OPTIMIZABLE, SO YOU NEED NOT PASS DOWN A NULL FIRST ARGUMENT.  HOWEVER,
; IF THE SINGLE ARGUMENT TO LIT INVOLVES A MACRO CALL, AND IF THAT MACRO'S
; PROPER OPERATION DEPENDS ON THE NUMBER OF TIMES IT IS CALLED, THE RESULTS
; WILL BE UNPREDICTABLE.  IN THIS CASE YOU SHOULD EXPLICITY PREVENT OPTIMIZA-
; WITH A NULL FIRST ARGUMENT.  FOR EXAMPLE, "LIT" ITSELF COULD BE USED
; IN THIS WAY:
;
;       LDA     0,LIT[,2*LIT[3_]]
;
; WILL LOAD AC0 WITH A BYTE POINTER TO A CONSTANT 3.  (THE "_" IS NECESSARY SO
; THAT MAC WILL NOT TAKE THE FIRST RIGHT BRACKET AS THE END OF THE OUTER
; LIT CALL.  (CROCK))
;
;       WHEN A LITERAL USE IS NOT OPTIMIZABLE, THE LITERAL IS
; EVALUATED ONLY AT LPOOL TIME, WHEN IT IS DUMPED IN LINE.  WHEN A LITERAL
; USE IS OPTIMIZABLE, ITS VALUE IS CALCULATED AT LIT TIME FOR USE IN
; COMPARING WITH OTHER OPTIMIZABLE LITERAL VALUES, SO THAT REDUNDANT LITERAL
; WORDS NEED NOT BE GENERATED.
;
;       IF FORWARD REFERENCE EXPRESSIONS (THOSE INVOLVING UNDEFINED SYMBOLS)
; ARE TO BE USED WITH LIT, THE FOLLOWING RULE MUST BE FOLLOWED:
; THE FORWARD REFERENCE SYMBOL MUST COME VERY FIRST IN THE EXPRESSION,
; EXCEPT FOR AN "@" SIGN, IF ONE IS PRESENT.  IN PARTICULAR, THE EXPRESSION
; MAY NOT START WITH "+", "-", OR "(".  (SORRY.)
;
;       THE USER MAY PLAY WITH HIS INPUT RADIX AS HE LIKES, AND EVERYTHING
; WILL COME OUT RIGHT.  THE RADIX IN FORCE AT LIT TIME WILL BE USED, EVEN
; THOUGH THE LITERAL WORD IS ACTUALLY GENERATED AT LPOOL TIME, WHERE THE
; INPUT RADIX MAY BE DIFFERENT.
```

```
;
;     - IF THE EXPRESSION FOR LIT HAS A CALCULABLE V LUE AT THE TIME OF
; THE LIT CALL, YET THE VALUE IS DIFFERENT AT LPOOL  IME (DUE TO A
; CHANGE IN THE VALUE OF ONE OR MORE OF THE COMPONENTS OF THE EXPRESSION
  BETWEEN THE LIT AND LPOOL CALLS), THE VALUE CALCULATED AT LIT TIME WILL
     LL BE USED, BUT THE LISTING WILL LOOK A LITTLE FUNNY.


;
;       SYMBOL USAGE IN LITMACS
;
; (HOPEFULLY-TO-BECOME) STANDARD MACRO SYMBOL CONVENTIONS
;
;       1. THE SYMBOLS ?0 THROUGH ?9 ARE RESERVED TO REPRESENT THE
;          NUMBERS 10. THROUGH 19. FOR USE IN REFERRING TO MACRO ARGUMENTS.
;       2. THE SYMBOLS ?A THROUGH ?Z ARE RESERVED FOR USE AS MACRO
;          TEMPORARIES.  MACROS USING THESE SYMBOLS ARE EXPECTED TO
;          SAVE THEIR VALUES BEFORE USING THEM AND RESTORE THEM BEFORE EXIT.
;          ANYONE (IN OR OUTSIDE OF MACROS) USING THESE SYMBOLS MUST USE
;          THEM IN A .DUSR MANNER ALWAYS, SINCE MAC DOES NOT ALLOW A
;          MIXTURE OF ASSIGNMENTS WITH A GIVEN SYMBOL.  FOR EXAMPLE:
;
;               .PUSH ?I                    ;TO SAVE
;               .DUSR ?I=WHATEVER
;               .DUSR ?I=?I+1               ;ALWAYS USE .DUSR
;               .DUSR ?I = .POP             ;TO RESTORE
;
;
; SYMBOL SPACE RESERVED BY LITMACS
;
;       LITMACS RESERVES FOR ITS EXCLUSIVE USE ALL SYMBOLS WHICH
  MATCH THE FOLLOWING TEMPLATES:

;       ?G---, ?L---, ?U---, ?V---, ?W---
;               WHERE --- IS THREE DECIMAL DIGITS;
;       ?L?--
;               WHERE -- IS TWO ALPHANUMERIC CHARACTERS.
;
;
; LISTING CONTROL
;
;       THE SYMBOL "?L?NS" CONTROLS THE LISTING FORMAT.  IF THIS IS
; SET TO ZERO (ITS DEFAULT VALUE IS ZERO) THEN ALL LITERALS ARE SHOWN
; AS THEY ARE STORED AT "LPOOL" TIME. IF SYMBOL IS NON-ZERO THEN THEY
; ARE SHOWN IN THE CROSS REF BUT NOT AT "LPOOL".
;       THE SYMBOL "?L?NI" CONTROLS THE LISTING OF "X" AND "XX" INSTRUCTION
; MACROS.  IF SYMBOL IS ZERO (ITS DEFAULT VALUE IS NON-ZERO) THEN THE
; INSTRUCTION GENERATED IS LISTED. IF SYMBOL IS NON-ZERO THEN NO
; LISTING IS DONE OF THE EXPANSION.
;
;
; RADIX CONTROL
;
;       IN ORDER TO EXPAND THE NUMBER OF LITERALS THAT CAN BE USED,
; LITMACS USES AN INTERNAL RADIX OF 10.  THE INTEGRITY OF THE USER'S
; RADIX IS PRESERVED BY SAVING IT ON ENTRY TO EACH USER-CALLABLE MACRO
  AND RESTORING IT ON EXIT.

;       ?L?IR           RADIX USED INTERNALLY, SET IN ?L?1X
;       ?L?UR           SAVE FOR USER RADIX
;
```

```
;
; NOTES:        THE INTERNAL RADIX IS IN EFFECT BY DEFAULT THROUGHOUT
;       THE MACROS.  THE INTERNAL RADIX MUST BE IN EFFECT WHEN USING
;       THE BACKSLASH ("\") FACILITY.  THE USER RADIX MUST BE IN EFFECT
;       WHENEVER EVALUATING THE ARGUMENTS TO THE LIT CALL, EITHER DIRECTLY
;       OR INDIRECTLY BY INVOKING THOSE ARGUMENTS VIA THE ?W MACRO.
;       ALSO BE VERY SURE THAT THE USER RADIX IS IN EFFECT WHEN ONE
;       OF THESE MACROS CALLS ANOTHER USER-CALLABLE MACRO, SINCE
;       THAT USER-CALLABLE MACRO WILL RE-SAVE THE RADIX IN ?L?UR.
;       (SEE, FOR EXAMPLE, HOW ?L?XI CALLS LIT.)
;
;
; COUNTERS/POINTERS
;
;       ?L?US           TOTAL NUMBER OF USES OF THE "LIT" MACRO,
;                       BUMPED ONLY ON PASS 1 SO THAT THE TOTAL NUMBER OF
;                       USES IS AVAILABLE THROUGHOUT PASS 2
;       ?L?CL           NUMBER OF LITERAL CURRENTLY WORKING ON IN "LIT"
;       ?L?PL           NUMBER OF LITERALS ALREADY DEPOSITED IN-LINE BY
;                       CALLS TO LPOOL
;       ?L?LO           NUMBER OF FIRST LITERAL THAT IS POSSIBLY WITHIN
;                       RANGE OF THE LITERAL WE ARE CURRENTLY WORKING
;                       ON IN "LPOOL" TO FIND A MATCH ALREADY STORED
;
; THE FOLLOWING INEQUALITY HOLDS FOR THE COUNTERS ABOVE:
;       ?L?LO <= ?L?PL <= ?L?CL <= ?L?US
; A THUMBNAIL SKETCH OF HOW THESE COUNTERS PROGRESS:
;       ?L?1X:  ON PASS 1, ALL INITIALIZED TO 0
;               ON PASS 2, ALL BUT ?L?US INITIALIZED TO 0
;       LIT:    ?L?US <- ?L?US + 1      (PASS 1 ONLY)
;               ?L?CL <- ?L?CL + 1      (BOTH PASSES)
;       LPOOL:  IN A LOOP, ?L?PL IS UPPED TO EQUAL ?L?CL AS EACH
;               LITERAL WORD IS GENERATED.  DURING THE LOOP, ?L?LO
;               IS ADVANCED WHEN APPROPRIATE.
;
;       ?L?GN           TOTAL NUMBER OF WORDS GENERATED BY CALLS TO LPOOL
;       ?L?NL           ON PASS 1, SET TO THE ADDRESS OF THE FIRST
;                       LITERAL WORD.  ON PASS 2, SET BY LIT EACH TIME TO
;                       THE ADDRESS OF THE LITERAL WORD FOR THE NEXT LITERAL
;                       USE TO OCCUR.
;
```

```
;
; ARRAYS
;
;       A GOOD DEAL OF INFORMATION MUST BE KEPT ABOUT EACH LITERAL USE.
; THIS INFORMATION IS HELD IN 5 "ARRAYS".  AN ARRAY IS MERELY A SET OF SYMBOLS
;  F THE FORM "XX---", WHERE "XX" IS ALPHABETIC AND "---" IS NUMERIC. (IN
; ORDER TO COMPLY WITH CERTAIN SYMBOL NAMING CONVENTIONS AND TO AVOID AS MANY
; SYMBOL CONFLICTS AS POSSIBLE, THE ARRAYS USED ALL BEGIN WITH "?".)  EACH
; ARRAY HOLDS THE SAME PIECE OF INFORMATION FOR ALL LITERAL USES.
; "SUBSCRIPTING" IS ACCOMPLISH WITH THE BACKSLASH ("\") FACILITY IN MAC.
; FOR EXAMPLE, TO ACCESS THE 16 BITS OF INFORMATION IN THE ?G ARRAY FOR
; THE LITERAL WHOSE NUMBER IS THE CURRENT VALUE OF ?I, SAY "?G\?I" WHILE THE
; INTERNAL RADIX IS IN EFFECT.  ("\" INSERTS DIGITS IN THE CURRENT INPUT
; RADIX.)  THE ARRAYS AND THEIR USE ARE DESCRIBED BELOW.
;
;       PREFIX          MEANING
;
;       ?G              THIS SYMBOL TELLS HOW THIS LITERAL HAS BEEN HANDLED.
;                       IT CONTAINS IN BITS 11-15 THE RADIX AT THE TIME
;                       THE LITERAL WAS USED (WHEN "LIT" WAS CALLED).
;                       IT ALSO CONTAINS THE FOLLOWING FLAGS:
;
;               ?L?G1   BIT SET BY LPOOL ON PASS 1 IF LITERAL USE
;                       REQUIRES LITERAL WORD TO BE GENERATED.  DURING
;                       PASS 2, SET MEANS LPOOL WILL(HAS) GENERATE(D) WORD
;                       FOR THIS LITERAL USE; CLEAR MEANS WILL(HAS) OPTIMIZE(D)
;                       THIS LITERAL WITH AN ALREADY GENERATED ONE.
;               ?L?CO   CAN'T OPTIMIZE THIS LITERAL.  SET BY LIT ON
;                       PASS 1 IF CAN'T CALCULATE VALUE AT THAT TIME.  IF SET,
;                       PREVENTS LPOOL FROM OPTIMIZING THIS LITERAL OR
;                       CONSIDERING THIS LITERAL FOR REUSE BY ANOTHER.
;               ?L?FG   FORCE GENERATION OF THIS LITERAL. (I.E.,DO NOT
;                       ATTEMPT TO OPTIMIZE THIS LITERAL USE.)  THIS FLAG
;                       IS NOT YET USED.
;
;       ?L              FOR LITERAL USE WHICH REQUIRES LITERAL WORD TO BE
;                       GENERATED (?L?G1 = 1), CONTAINS ADDRESS OF LOCATION
;                       WHERE LITERAL WAS DEPOSITED.  (DETERMINED BY LPOOL.)
;                       FOR LITERAL USE WHICH USED ANOTHER LITERAL WORD
;                       (?L?G1 = 0), CONTAINS NUMBER OF THAT OTHER LITERAL USE,
;                       UNTIL LPOOL TIME ON PASS 2, WHEN SET TO CONTAIN ADDRESS
;                       OF THAT OTHER LITERAL WORD, FOR CONSISTENCY IN THE
;                       CROSS REFERENCE.
;       ?U              CONTAINS THE ADDRESS AT WHICH THE LITERAL USE
;                       OCCURRED FOR THIS LITERAL.
;       ?V              HOLDS VALUE OF LITERAL.  THIS SYMBOL ONLY EXISTS IF
;                       A VALUE COULD BE DETERMINED FOR THE SYMBOL AT "LIT"
;                       TIME ON PASS 1.  THE ?L?CO FLAG TELLS WHETHER
;                       THIS SYMBOL EXISTS: ?L?CO = 0 MEANS YES, = 1 MEANS NO.
;       ?W              THIS SYMBOL IS A MACRO WHICH HOLDS THE TEXT
;                       OF THE ARGUMENT(S) TO "LIT".  IT IS INVOKED WHENEVER
;                       THOSE ARGUMENTS MUST BE DEALT WITH.
;
```

```
;
; INTERNAL MACROS
;
;       ?L?1X           INITIAL SETUP MACRO. CALLED BY "LIT", "LPOOL", AND
;                       "?L?XI" THE FIRST TIME THEY ARE CALLED.  THIS MACRO
;                       INITIALIZES ALL WORK SYMBOLS AND DEFINES THE
;                       PARAMETER SYMBOLS.
;       ?L?DU           EITHER DEFINES A SYMBOL NORMALLY ("?L?NS"=1) SO IT
;                       WILL APPEAR IN THE CROSS REFERENCE OR DEFINES IT
;                       USING ".DUSR" ("?L?NS"=0) SO IT WILL NOT APPEAR IN
;                       CROSS REFERENCE.
;       ?L?L1           DOES PASS 1 ONLY PROCESSING OF "LIT" MACRO CALL.
;                       CONTAINS THE CODE FOR FIGURING WHAT TO DO WITH LIT
;                       ON PASS 1.
;       ?L?L2           DOES PASS 2 PART OF LIT PROCESSING.
;       ?L?T1           TESTS FIRST ARGUMENT TO "LIT" CALL TO FIND OUT
;                       IF CAN OPTIMIZE ITS USAGE (I.E., CHECK THAT IS DOES NOT
;                       HAVE TO SET THE "?L?CO" BIT IN THE "?GXXX" SYMBOL FOR
;                       THE LITERAL).
;       ?L?T2           SECOND OPTIMIZABILITY TEST.
;       ?L?T3           FINAL OPTIMIZABILITY TEST.
;       ?L?DL           FORMATS THE APPEARANCE OF THE LITERAL WHEN IT IS
;                       DEPOSITED BY "LPOOL".
```

```
;       USE OF MAC'S T' FACILITY
;
; THESE MACROS USE THE SECRET (?) T' FACILITY OF MAC.  T' FOLLOWED
; BY AN ATOM RETURNS 16 BITS OF INFORMATION ABOUT THAT ATOM.
; BELOW ARE DESCRIBED THOSE PIECES OF INFORMATION WHICH THESE MACROS USE.

        B0              1 IFF "@" IMMEDIATELY PRECEDES ATOM
        B1              1 IFF ATOM IS SYMBOL
;               B2      1 IFF ATOM IS NUMBER
;               B3      1 IFF ATOM IS OPERATOR
;               B4      1 IFF ATOM IS BREAK
;       B5              1 IFF SYMBOL/ATOM IS DEFINED
;               B6      1 IFF SYMBOL/ATOM IS MULTIPLY DEFINED
;               B7      1 IFF SYMBOL/ATOM IS PERMANENT (E.G., .DUSR)
;
;       B8-12           SYMBOL TYPE:
;                       THE .D---'S OCCUPY MANY SLOTS HERE.  THE CASES
;                       OF INTEREST ARE:
;               1B1+0B12        .ENT
;               1B1+1B12        .EXTN
;               1B1+2B12        .COMM
;               1B1+3B12        .EXTD
;               1B1+4B12        .ENTO
;               1B1+7B12        MACRO (NOTE: T' ON MACRO BOMBS MAC.)
;               1B1+8.B12       USER SYMBOL
;
;       B13-15          RELOCATION TYPE:
;               1               ABSOLUTE
;               2               NREL
;               3               BYTE NREL
;               4               ZREL
;               5               BYTE ZREL
;               6               EXTD
```

```
;
; MUST BE CALLED BY ANY MACRO USING LITERAL FACILITY VARIABLES,
;  BEFORE DOING ANYTHING SUBSTANTIVE.

        .MACRO  ?L?1X

**.DO .MCALL==0          ;;ONCE ON EACH PASS
**
**  .DO .PASS==0         ;;ON PASS 1 ONLY
**
**;;IF OPTION SWITCH(ES) UNDEFINED, DEFAULT IT(THEM)
**
**    .DO T'?L?NS&1B5==0
**      .DUSR  ?L?NS = 0        ;;DEFAULT IS SHOW LITS AT LPOOL TIME
**    .ENDC
**
**    .DO T'?L?NI&1B5==0
**      .DUSR  ?L?NI = 1        ;;DEFAULT IS DON'T SHOW X-INSTR
**    .ENDC
**
**;;DEFINE PARAMETERS FOR THE LITERAL MACROS
**
**      .DUSR  ?L?IR = 10.      ;;INTERNAL RADIX
**
**      ;;STRUCTURE OF FLAGS WORD ("?GXXX") FOR EACH LITERAL
**
**      .DUSR  ?L?G1 = 1B0      ;;GENERATED ON PASS 1
**      .DUSR  ?L?CO = 1B1      ;;CAN'T OPTIMIZE (HENCE, HAS NO ?V WORD)
**      .DUSR  ?L?FG = 1B2      ;;FORCE GENERATION OF THIS LIT (NOT YET USED)
  ;;    .RDX AT TIME OF LIT IS SAVED IN LOW 5 BITS (DON'T CHANGE).
  ;;    MASK FOR THIS SAVED RADIX IS, THEREFORE, "31." .

**;;TOTAL NUMBER OF LITERALS DECLARED ON PASS 1
**
**      ?L?DU  ?L?US = 0        ;;NUMBER OF USES OF LITERALS
**
**  .ENDC
**
**;;COUNTERS USED ON BOTH PASSES
**
**      ?L?DU  ?L?CL = 0        ;;CURRENT LITERAL NUMBER
**      ?L?DU  ?L?GN = 0        ;;NUMBER OF WORDS GENERATED FOR LITERALS
**      .DUSR  ?L?PL = 0        ;;NUMBER OF LITERALS POOLED ALREADY
**      .DUSR  ?L?LO = 0        ;;FIRST LITERAL POSSIBLY WITHIN RANGE
**
**.ENDC
%


; ?L?DU  -  DEFINE A SYMBOL IN A .DUSR MANNER OR NOT,
;       ACCORDING TO SWITCH ?L?NS

        .MACRO  ?L?DU
**
 .DO ?L?NS<>0
                ^1 ^2 ^3        ;;LET IT SHOW IN XREF
  .ENDC X
**      .DUSR  ^1 ^2 ^3         ;;DON'T LET IT SHOW IN XREF
**[X]
%
```

```
; LIT  -  USE A LITERAL
;
; "LIT[<EXPRESSION>]" YIELDS THE ADDRESS OF A MEMORY LOCATION
; (OR LOCATIONS) CONTAINING THE VALUE OF <EXPRESSION>.

        .MACRO  LIT
..NL
**                                  ;;REPLACE LIT CALL BY ADDRESS ?L?NL
**
**.DO .MCALL==0          ;;THE FIRST TIME
**      ?L?1X                               ;;MAY NEED TO INIT
**.ENDC
**
**      .DUSR  ?L?UR = .RDX                 ;;SAVE USER RADIX
**      .RDX ?L?IR                          ;;ESTABLISH INTERNAL RADIX
**
**.DO ?L?CL/1000<>0                  ;;CHECK FOR OVERFLOW
**:     ;ERROR: TOO MANY LITERALS DEFINED
**.ENDC EXIT
**
**      .PUSH   .NOMAC
**      .NOMAC  1                   ;;DON'T SHOW ANYTHING
**
**      .PUSH   ?I                  ;;SAVE TEMP
**
**      .LOC    .-1                 ;;MAKE "." MEAN WHAT IT SHOULD
**      ?L?DU  ?U\?L?CL = .         ;;DEFINE USE LOCATION (?U###)
**
**
**.DO .PASS==0                       ;;ON PASS ONE:

        .DUSR  ?I = .ARGCT<>1       ;;INIT FLAG
*.
**      .MACRO ?W\?L?CL             ;;REMEMBER TEXT OF LITERAL AS MACRO
**.RDX .POP
_^1_^2_^3_^4_^5_^6^1 ^2 ^3 ^4 ^5 ^6 ^7 ^8 ^9
_%
**
**      ?L?L1   ^1                  ;;DO PASS 1 LIT HANDLING
**
**.ENDC PASS2                        ;;ON PASS TWO:
**
**      ?L?L2   ^1                  ;;DO LIT PASS 2 PROCESSING
**
**[PASS2]
**
**      .LOC    .+1                         ;;RESTORE LOCATION COUNTER
**      .DUSR  ?I = .POP                    ;;RESTORE TEMP
**      .NOMAC  .POP                        ;;RESTORE .NOMAC
**      .RDX ?L?UR                          ;;RESTORE USER RADIX
**
**[EXIT]
**
**%                      ;(SINCE LIT'S ARGS IN BRACKETS)

  ?L?L1 - PASS 1 ONLY "LIT" HANDLING

        .MACRO  ?L?L1
**
**      ?L?DU  ?L?US = ?L?US+1          ;;BUMP LITERAL COUNT
**
**        .RDX ?L?UR                    ;;SET USER RADIX
**
```

```
**.DO ?I==0              ;;TRY TO OPTIMIZE ONLY IF EXACTLY 1 ARG TO LIT
**
**  .DO 1
**      ?L?T1 ?I ^1             ;;FIRST TEST FOR OPTIMIZABILITY
    .ENDC                       ;;(CROCK IN MAC REQUIRES
                                ;; LOCAL .DO/.ENDC PAIRS)
    .DO 1
**      ?L?T2 ?I ^1             ;;SECOND TEST
**  .ENDC
**
**      .DUSR  ?I = ?I/2.       ;;KNOW AT THIS POINT THAT CAN'T OPTIM
**                              ;;ONLY IF BOTH TESTS ERRORED
**
**  .DO ?I==0                   ;;IF STILL LOOKS OPTIMIZABLE
**      ?L?T3 ?I ^1             ;;FINAL TEST FOR OPTIMIZABILITY
**  .ENDC
**
**.ENDC
**
**        .RDX ?L?IR                        ;;RESTORE INTERNAL RADIX
**
**      .DUSR  ?G\?L?CL = ?I*?L?CO+?L?UR    ;;INIT FLAGS WORD
**
**.DO ?I==0                      ;;IF OPTIMIZABLE
**        .RDX ?L?UR
**.DUSR  ?I = ^1         ;INVALID FORWARD REFERENCE EXPRESSION
**      ;;(ABOVE LINE GETS "U" ERROR ON INVALID F.R. EXPR)
**        .RDX ?L?IR
**      ?L?DU  ?V\?L?CL = ?I              ;;SAVE VALUE (?V###)
**.ENDC

        ?L?DU  ?L?CL = ?L?CL+1            ;;READY FOR NEXT LITERAL

; ?L?L2 - PASS 2 LIT PROCESSING

        .MACRO  ?L?L2
**
**.DO ?G\?L?CL&?L?CO==0          ;;IF OPTIMIZABLE
**        .RDX ?L?UR
**      .DUSR  ?I = ^1                          ;;GET VALUE (ALLOW FOR @)
**        .RDX ?L?IR
**  .DO ?V\?L?CL<>?I                ;;MAKE SURE THE VALUE
**:     ;ERROR: VALUES ON TWO PASSES OF LIT DIFFER
**  .ENDC                                       ;;AGREES WITH PASS ONE
**.ENDC
**
**      ?L?DU  ?L?CL = ?L?CL+1            ;;READY FOR NEXT LITERAL
**
**.DO   ?L?US>?L?CL      ;;ON PASS 2, SET UP ?L?NL FOR NEXT LIT
**                              ;;(IF THERE IS A NEXT LIT)
**  .DO ?G\?L?CL&?L?G1<>0                   ;;IF LIT WILL BE GENERATED
**      .DUSR  ?L?NL = ?L\?L?CL                     ;;?L\?L?CL IS NEXT LIT ADDR
**  .ENDC X                                 ;;ELSE
**      .DUSR  ?L?NL = ?L\?L\?L?CL                  ;;?L\?L?CL IS LIT # NEXT LIT
    [X]                                             ;;WAS OPTIMIZED WITH
  .ENDC

%
```

```
; MACROS FOR TESTING LIT'S ARGUMENT FOR OPTIMIZABILITY
;   DON'T TOUCH THESE GUYS UNLESS YOU REALLY UNDERSTAND
;   WHAT'S GOING ON (AND WHO DOES?).

  THE PRINCIPLE OF THESE MACROS IS THIS:
;   A FLAG (ARGUMENT 1) IS PRESET.  A .DO IS EXECUTED.  IF THE .DO
;   SUCCEEDS, THE NEXT LINE IS EXECUTED AND FLIPS THE VALUE OF THE
;   FLAG.  SINCE THERE IS NO TERMINATING .ENDC, THE MACRO ENDS WITH
;   AN UNFINISHED .DO, WHICH IS "DISCARDED".  (HOWEVER, THE MACRO CALL
;   MUST BE TIGHTLY EMBEDDED IN A .DO/.ENDC TO REVERT ASSEMBLY MODE
;   PROPERLY, DUE TO CROCK IN MAC.)  IF .DO FAILS THE TEST, THE
;   FOLLOWING LINE IS NOT EXECUTED, AND THE UNFINISHED .DO IS HANDLED
;   AS ABOVE.  IF THE .DO RESULTS IN AN ERROR (AS, FOR EXAMPLE, IF
;   THERE IS AN EXTERNAL SYMBOL IN THE .DO EXPRESSION), THE .DO IS
;   IGNORED, THE NEXT LINE GETS EXECUTED, AND THE MACRO ENDS NORMALLY
;   (I.E., NO UNFINISHED .DO).  NOTE THAT THESE MACROS ARE CALLED ONLY
;   ON PASS 1, AND ERRORS IN .DO STATEMENTS APPEAR ONLY ON PASS 2, SO
;   THE ASSEMBLY REMAINS CLEAN-LOOKING.  THIS MAC BEHAVIOR IS ADMITTEDLY
;   A CROCK, AS IS THESE MACROS' DEPENDENCE ON IT.


; THE FIRST 2 MACROS TEST FOR BASIC NON-OPTIMIZABILITY.  IF AN EXPRESSION
;   ERRORS ON BOTH .DO'S (E.G., NULL, @, EXTERNAL, MALFORMED EXPRESSION),
;   IT IS DEEMED NON-OPTIMIZABLE.

        .MACRO  ?L?T1
**.DO ^2==0                     ;;IF ERROR OR 0
**      .DUSR  ^1 = ^1+1                ;;INCREMENT FLAG
%

        .MACRO  ?L?T2
*     ^2<>0                     ;;IF ERROR OR NOT 0
*       .DUSR  ^1 = ^1+1                ;;INCREMENT FLAG
%


; THE FINAL OPTIMIZABILITY TEST MACRO IS REQUIRED BECAUSE
;   FORWARD REFERENCE EXPRESSIONS WILL COME THROUGH THE FIRST 2 MACROS
;   LOOKING OPTIMIZABLE.  THE USE OF FORWARD REFERENCE EXPRESSIONS,
;   HOWEVER, IS QUITE LIMITED: THE FORWARD REFERENCE SYMBOL MUST BE
;   THE VERY FIRST ATOM IN THE EXPRESSION, EXCEPT FOR AN @ SIGN.
;   (THE FIRST ATOM ESPECIALLY CANNOT BE +, -, OR LEFT PAREN.)
;   IF THIS RESTRICTION IS VIOLATED, THE USER WILL HEAR ABOUT IT
;   THROUGH A "U" ERROR ON THE ASSIGNMENT LINE IN LIT.

        .MACRO  ?L?T3           ;;FINAL OPTIMIZABILITY TEST
**
**      .DUSR  ^1 = 1           ;;ASSUME NOT OPTIMIZABLE
**.DO (-16449.&T'^2-(^2))<>0    ;;UNLESS UNDEFINED FIRST ATOM
**      .DUSR  ^1 = 0           ;;SIGNAL OPTIMIZABLE
%
                        ;NOTE: PARENTHESES ABOVE ARE REQUIRED (!!?!!)
                        ;       IN ORDER TO HANDLE LEADING +, -, (


  DETAILS CONCERNING THE THIRD MACRO'S OPERATION:
;     T'<EXPR>-(<EXPR>)    , AFTER CANCELLATION, COMES OUT TO
;   T'<1ST ATOM>-<1ST ATOM> .  IF <1ST ATOM> IS UNDEFINED (FORWARD
;   REFERENCE), T' OF IT IS EITHER OCTAL 40000 (FOR A .ENT'ED SYMBOL)
;   OR OCTAL 40100 (IF NOT .ENT'ED).  -16449. IS THE COMPLEMENT OF
;   OCTAL 40100, SO -16449.&T'<1ST ATOM> WILL BE 0 FOR AN UNDEFINED SYMBOL.
;   SINCE THE VALUE OF AN UNDEFINED SYMBOL ALSO APPEARS TO BE 0,
```

```
;    THE EXPRESSION WILL BE EQUAL TO 0, THE .DO TEST WILL FAIL,
;    THE NEXT LINE WILL BE SKIPPED, AND THE EXPRESSION WILL BE SIGNALLED
;    AS NON-OPTIMIZABLE.  IF <1ST ATOM> IS DEFINED WITH RELOCATION
;    OTHER THAN ABSOLUTE, THE .DO WILL ERROR SINCE THE
     RELOCATIONS WILL NOT WORK OUT.  IF <1ST ATOM> IS
     DEFINED AND ABSOLUTE, IN NEARLY ALL CASES THE .DO TEST WILL
     SUCCEED AND OPTIMIZABILITY WILL BE SIGNALLED.  THE ABSOLUTE CASES
;    WHICH WILL BE SIGNALLED (UNFORTUNATELY) AS NON-OPTIMIZABLE ARE:
;
;        LIT[20001]
;        SYM=2001    ...   LIT[SYM]
;        .DUSR SYM=2401   ...   LIT[SYM]
;
;                (ALL NUMBERS ABOVE ARE IN OCTAL.)
```

```
; LPOOL  -  DEPOSIT ALL LITERALS USED SINCE LAST LPOOL
;
; LPOOL GENERATES IN-LINE LITERAL WORDS FOR ALL LITERAL USES
; SINCE LAST LPOOL.  WHENEVER POSSIBLE, AN ALREADY GENERATED
  LITERAL WORD IS REUSED.

        .MACRO  LPOOL
**
**.DO .MCALL==0                         ;;IF FIRST TIME
**      ?L?1X                           ;;MAY NEED TO INIT
**.ENDC
**
**      .DUSR  ?L?UR = .RDX                     ;;SAVE USER RADIX
**      .RDX ?L?IR                              ;;ESTABLISH INTERNAL RADIX
**
**      .PUSH   .NOMAC
**      .NOMAC  ?L?NS           ;;SHOW THE POOL ACCORDING TO ?L?NS
**
**      .PUSH   .NOCON
**      .NOCON  1               ;; BUT NEVER ANY GARBAGE
**
**      .PUSH   ?A              ;;SAVE TEMPORARIES
**      .PUSH   ?B
**      .PUSH   ?C
**      .PUSH   ?D
**      .PUSH   ?I
**
**
**.DO .PASS==0           ;; IF PASS 1 THEN DO
**
        ?L?P1           ;; PASS 1 LPOOL ELSE

  .NDC PASS2            ;; DO PASS 2
**
**      ?L?P2
**
**[PASS2]
**
**      .DUSR   ?I = .POP       ;;RESTORE MACRO TEMPORARIES
**      .DUSR   ?D = .POP
**      .DUSR   ?C = .POP
**      .DUSR   ?B = .POP
**      .DUSR   ?A = .POP
**      .NOCON  .POP            ;;RESTORE LISTING CONTROLS
**      .NOMAC  .POP
**      .RDX    ?L?UR           ;;RESTORE USER RADIX
**
**.DO ?L?PL<>?L?CL               ;;IF LIT USED DURING LPOOL,
**      LPOOL                   ;;MAKE ANOTHER PASS
**.ENDC
**
%



 ?L?DL  -  DEPOSIT LITERAL

   TERNAL MACRO FOR GENERATING LPOOL LISTING LINES

        .MACRO  ?L?DL
**
**      .DUSR  ?B = ^2/10       ;;BREAK LIT NUMBER INTO DIGITS
**      .DUSR  ?C = ^2-(?B*10)+5.
```

```
**       .DUSR  ?A = ?B/10
**       .DUSR  ?B = ?B-(?A*10)+5.
**       .DUSR  ?A = ?A+5.
**
         .PUSH ?G\?L?PL&31.              ;;RADIX TO STACK FOR MACRO
       ?W\^4 ^1 ?L ^?A ^?B ^?C ^3                ;;DEPOSIT LITERAL
         .RDX ?L?IR                      ;;RESTORE INTERNAL RADIX
**
**%                                ;(SINCE ?L?DL'S ARGS IN BRACKETS)


; ?L?P1 - PASS 1 PART OF LPOOL
;
; FIGURES OUT THE OPTIMUM ALOCATION OF LITERALS IN CORE

       .MACRO  ?L?P1
**
**.DO ?L?CL==?L?PL                  ;;IF NOTHING TO DO THEN
**.ENDC DONE                       ;;DONE ELSE
**
**.DO ?L?PL-?L?LO                   ;; LOOP TO EXCLUDE ALL MATCH CANDIDATES
**                                  ;;THAT ARE OUT OF RANGE OR MUST SEPARATE
**  .DO (?G\?L?LO&(?L?G1+?L?CO)<>?L?G1)!((?L\?L?LO-?U\?L?PL)>128.)
**
**       .DUSR   ?L?LO = ?L?LO+1          ;;FORGET THIS LITERAL
**
**  .ENDC GOOD
**
**.ENDC
**
  [GOOD]

...DO ?L?CL-?L?PL                  ;;FOR EACH UNPOOLED LITERAL
**
** .DO T'?L\?L?PL&1B5<>0            ;;IF LIT ALREADY OPTIMIZED(BY X/XX INSTRUCTIONS)
** .ENDC NXTLIT                     ;;THEN ALL DONE ELSE
**
** .DO ?G\?L?PL&(?L?G1+?L?CO+?L?FG)==0  ;;IF SEPARATE LIT NOT REQUIRED
**
**       .DUSR  ?I = ?L?LO                ;;ONLY SCAN MATCH CANDIDATES
**                                        ;;WITHIN ADDRESSING RANGE
**  .DO ?L?PL-?L?LO                ;;FOR EACH LIT MAYBE STILL IN RANGE
**
**   .DO ?G\?I&(?L?G1+?L?CO)==?L?G1      ;;IF GENERATED & OPTIMIZABLE
**
**       .DUSR  ?D = ?L\?I-?U\?L?PL       ;;GET DISPLACEMENT TO CANDIDATE
**
**    .DO (?V\?L?PL==?V\?I)&(?D<128.)&(?D>=-128.)        ;;IF VALS MATCH & IN RAN
**      ?L\?L?PL = ?I    ;;SAVE LIT # OPTIMIZED WITH, LEAVE G1 BIT 0
**    .ENDC NXTLIT                     ;;DONE WITH THIS LIT
**
**   .ENDC
**
**       .DUSR   ?I = ?I+1          ;;TRY NEXT CANDIDATE
**
**  .ENDC

  .ENDC
**
**;;NEED TO GENERATE A NEW LITERAL
**
**       .DUSR   ?D = .             ;;REMEMBER WHERE WE STARTED
**
```

```
; ?L?XI  -  PROCESS AN "X" INSTRUCTION
;
; CALL:
;  ?L?XI <INSTR>,<AC>,<TARGET (MULTI ATOMS OK) >


        .MACRO  ?L?XI
**
**.DO .MCALL==0          ;;THE FIRST TIME,
**      ?L?1X                           ;;MAY NEED TO INIT
**.ENDC
**
**      .DUSR  ?L?UR = .RDX             ;;SAVE USER RADIX
**
**      .PUSH ?D                        ;;SAVE TEMPS
**      .PUSH ?F
**
**
**.DO '^3'&-1B7=='#'             ;;IF "#" USAGE
**      ?L?XN   ^1,^2,^3 ^4 ^5 ^6 ^7 ^8 ^9      ;;HANDLES #
**.ENDC DONE
**
**
**.DO '^3'&-1B7=='='             ;;IF "=" USAGE
**      ?L?XE   ^1,^2,^3 ^4 ^5 ^6 ^7 ^8 ^9      ;;HANDLES "="
**.ENDC DONE
**
**      ?L?XR   ^1,^2,^3 ^4 ^5 ^6 ^7 ^8 ^9      ;;HANDLES MEMEORY REFERENCE
**
 [DONE]
;
        .DUSR  ?F = .POP                ;;RESTORE TEMPS
**      .DUSR  ?D = .POP
**
%


; ?L?XN  -  PROCESS AN "#".
;
; HANDLES LITERALS THAT MUST BE SEPARATE.

        .MACRO  ?L?XN
**.DO '^3'<>'#'                 ;;IF MORE THAN JUST "#"
**:;    YOU FORGOT THE SPACE AFTER THE # SIGN
**      ^1 ^2 LIT[,0+0+0+0+0]           ;;ALLOC WORD HERE AND IN LPOOL
**.ENDC DONE
**
**.DO (.PASS==0)!(?L?NI<>0)
**      ^1  ^2 LIT[,^4 ^5 ^6 ^7 ^8 ^9]  ;;ALWAYS GEN A LITERAL
**.ENDC DONE
**      ?L?DI   ^1,^2,,0 1 2 3 4 5 6 7 8 9 A B C D E F G H I J
**      .DUSR   ?F = LIT[,^4 ^5 ^6 ^7 ^8 ^9]    ;;FAKE CALL TO LIT
**[DONE]
%

 ?L?XE  -  PROCESS "="
;
;  ..OCESS A LITERAL THAT CAN BE OPTIMIZED

        .MACRO  ?L?XE
**.DO '^3'<>'='                 ;;IF MORE THAN JUST "="
**:;    YOU FORGOT THE SPACE AFTER THE = SIGN
```

```
**       .PUSH   ?V\?L?PL                    ;;CORRECT VALUE TO STACK
**       ?L?DL [_____ ,?L?PL,:_____ .POP_____ _____:_____ ,?L?PL
0 1 2 3 4 5 6 7 8 9 A B C D E F G H I J]      ;;USE THE VALUE FORM
    .ENDC SKIP

   .ENDC                            ;;ELSE
**
**       ?L?DL [_____ ,?L?PL,:_____ ,?L?PL
0 1 2 3 4 5 6 7 8 9 A B C D E F G H I J]      ;;USE THE MACRO FORM
**
** [SKIP]
**
**       ?L?DU  ?L?GN = ?L?GN+.-?D           ;;INC # WORDS GENERATED
**       .DUSR  ?G\?L?PL = ?G\?L?PL!?L?G1    ;;FLAG AS GENERATED
**                                           ;;SO PASS 2 WILL DO SAME
**
**[NXTLIT]
**
**       .DUSR  ?L?PL = ?L?PL+1   ;;READY TO PROCESS NEXT
**
**.ENDC
%
```

```
;               X INSTRUCTIONS
;
;       SET OF ALTERNATIVE MEMORY REFERENCE INSTRUCTION MACROS
;       TO HELP GET AROUND NOVA ADDRESSING PROBLEMS

       THE X INSTRUCTIONS DO THEIR BEST TO ACCESS DATA IN THE MOST
EFFICIENT WAY POSSIBLE.  THEY HAVE THE IMPORTANT PROPERTY OF
BEING A SINGLE IN-LINE WORD IN ALL CASES, SO THEY MAY BE SKIPPED OVER.
IF THE ADDRESS REFERRED TO IS OUT OF RANGE, REFERENCE TO IT IS MADE
INDIRECT THROUGH A POINTER GENERATED USING LITERALS.

       THERE ARE SIX X INSTRUCTIONS: ONE TO MATCH EACH OF THE
BASIC NOVA MEMORY REFERENCE INSTRUCTIONS.  THE BASIC FORMAT IS
THE SAME AS FOR THE NOVA INSTRUCTIONS:

       XLDA    0,DATA          ;LOAD AC0 WITH CONTENTS OF WORD "DATA"
       XJSR    SUBRT           ;JUMP TO SUBROUTINE "SUBRT"
       XISZ    FLAG            ;INCREMENT "FLAG" AND SKIP IF 0
       XSTA    2,@PTR          ;STORE AC2'S CONTENTS INDIRECT THROUGH "PTR"
       XJMP    @DPTCH          ;JUMP INDIRECT THROUGH "DPTCH"
       XDSZ    COUNT           ;DECREMENT "COUNT" AND SKIP IF 0

NOTE THAT THE INDEXED MODES OF ADDRESSING MAY NOT BE USED WITH
THE X INSTRUCTIONS.


       IN ADDITION TO THE BASIC FORMAT DESCRIBED ABOVE, THERE
ARE TWO ALTERNATE FORMATS.  THESE ALTERNATE FORMATS GIVE THE USER
ACCESS TO THE LITERAL FACILITY THROUGH THE X INSTRUCTIONS.
THE USER SPECIFIES A LITERAL REFERENCE BY WRITING AN "=" SIGN
BEFORE THE DATA HE WISHES TO ACCESS LITERALLY.  FOR EXAMPLE:

       XLDA    0,= 377         ;LOAD CONSTANT 377 INTO AC0

THE ABOVE EXAMPLE IS EQUIVALENT TO

       LDA     0,LIT[377]

EXCEPT THAT THE FORMATTING COMES OUT NICER: NO CRAZY SYMBOL
APPEARS (AS WOULD FOLLOWING THE RIGHT BRACKET), AND THE
COMMENT, IF ANY, IS NOT LOST.

       PLEASE NOTE THAT THE SPACE FOLLOWING THE "=" SIGN IS
REQUIRED, SO THE XLDA MACRO CAN DEAL WITH "377" SEPARATELY.
(IF YOU EVER FORGET THE SPACE, YOU WILL GET AN ERROR MESSAGE
REMINDING YOU THAT YOU FORGOT IT.)

       IF IT SHOULD EVER BE NECESSARY TO FORCE THE LITERAL
FACILITY NOT TO TRY TO OPTIMIZE A LITERAL USE, WRITE A "#" SIGN
INSTEAD OF THE "=" SIGN.  AGAIN, THE SPACE FOLLOWING THE "#"
SIGN IS REQUIRED.
```

```
;                XX INSTRUCTIONS
;
;  XX INSTRUCTIONS ARE LIKE X INSTRUCTIONS, EXCEPT THAT THEY
;  INCLUDE A BOOKKEEPING SYMBOL FOR OPTIMIZING OTHERWISE NON-
;  OPTIMIZABLE LITERAL REFERENCES SUCH AS FORWARD REFERENCES
; AND EXTERNAL REFERENCES.  THE USER SUPPLIES A UNIQUE
; BOOKKEEPING SYMBOL FOR EACH SEPARATELY VALUED LITERAL REFERENCE.
; THE BOOKKEEPING SYMBOL IS WRITTEN AS THE LAST SYMBOL IN THE
; CALL TO THE XX INSTRUCTION MACRO.  FOR EXAMPLE:
;
;       XXLDA   0,DATA,.DATA
;
; ACTS JUST LIKE
;
;       XLDA    0,DATA
;
; EXCEPT THAT MULTIPLE REFERENCES OF THIS TYPE TO "DATA" MAY BE ABLE
; TO REUSE LITERAL WORDS ALREADY GENERATED FOR EARLIER REFERENCES TO "DATA".
;
;       THE BOOKKEEPING SYMBOL YOU PROVIDE MUST BE UNDEFINED AT
; THE TIME IT IS FIRST USED, AND CAN ONLY BE USED FOR LITERAL
; REFERENCES TO THE SAME EXPRESSION.  REMEMBER NOT TO USE THE SAME
; BOOKKEEPING SYMBOL FOR "DATA" AND "@DATA".
;
;       IF NO BOOKKEEPING SYMBOL IS PROVIDED THEN THE LITERAL EXPRESSION
; MUST CONSIST OF ONLY ONE SYMBOL.  THE MACRO WILL USE THIS SYMBOL
; PREFIXED BY "." AS THE BOOKKEEPING SYMBOL.  THE EXAMPLE STATED ABOVE
; CAN ALSO BE DONE AS

       XXLDA   0,DATA

;
; THIS HAS EXEACTLY THE SAME EFFECT AS THE PREVIOUS EXAMPLE.




; CURRENT RESTRICTIONS:
;       WON'T HANDLE ZREL, EXTD, EXTU, ABS
;       XX INSTRUCTIONS ALLOW ONLY 1 TARGET ATOM
;       DON'T USE SAME BOOKKEEPER FOR "SYM" AND "@SYM"
;       CAN'T REFERENCE EXTN SYMBOL FROM ABS CODE
;       IF @ USED IT MUST IMMEDIATELY PRECEDE FOLLOWING
;         SYMBOL WITH NO INTERVENING SPACE
```

```
** .DO ?G\?L?PL&?L?CO==0          ;;IF OPTIMIZABLE
**
**          .PUSH ?G\?L?PL&31.          ;;RADIX TO STACK FOR MACRO
**        ?W\?L?PL **.DUSR,__ ,?I=      ;;ASSIGN ?I TO MACRO VALUE
            .RDX ?L?IR                  ;;RESTORE INTERNAL RADIX

    .DO ?I<>?V\?L?PL                    ;;IF MACRO VALUE WRONG,
**        .PUSH   ?V\?L?PL              ;;CORRECT VALUE TO STACK
**        ?L?DL [_____ ,?L?PL,:_____ .POP_____ _____;_____ ,?L?PL
0 1 2 3 4 5 6 7 8 9 A B C D E F G H I J]     ;;USE THE VALUE FORM
**    .ENDC SKIP
**
** .ENDC                              ;;ELSE
**
**        ?L?DL [_____ ,?L?PL,:_____ ,?L?PL
0 1 2 3 4 5 6 7 8 9 A B C D E F G H I J]     ;;USE THE MACRO FORM
**
** [SKIP]
**
**        ?L?DU  ?L?GN = ?L?GN+.-?D   ;;INC # WORDS GENERATED
**        .DUSR  ?G\?L?PL = ?G\?L?PL!?L?G1  ;;FLAG AS GENERATED
**                                    ;;SO PASS 2 WILL DO SAME
**
**[NXTLIT]
**
**        .DUSR  ?L?PL = ?L?PL+1      ;;READY TO PROCESS NEXT
**
**.ENDC
**
**[DONE]

    D ?L?US>0    ;;WHEN ?L000 IS DEFINED
*         .DUSR  ?L?NL = ?L000        ;;INIT ?L?NL FOR PASS 2
**.ENDC
%


; ?L?P2 - PASS 2 LPOOL CODE
;
; DOES ALL PASS 2 POOLING OF LITERALS

          .MACRO  ?L?P2
**.DO ?L?CL-?L?PL                   ;;FOR EACH UNPOOLED LITERAL
**
** .DO ?G\?L?PL&?L?G1==0             ;;IF SEPARATE LIT NOT REQUIRED
**
**        ?L?DL [_____;,?L\?L?PL,:_____ ,?L?PL
0 1 2 3 4 5 6 7 8 9 A B C D E F G H I J]
**                                  ;;GENERATE THE LISTING LINE,
**                                  ;; SHOWING REUSE OF LIT # ?L\?L?PL
**
** .ENDC NXTLIT
**
**;;NEED TO GENERATE A NEW LITERAL
**
          .DUSR   ?D = .            ;;REMEMBER WHERE WE STARTED

    DO ?G\?L?PL&?L?CO==0             ;;IF OPTIMIZABLE
**
**          .PUSH ?G\?L?PL&31.          ;;RADIX TO STACK FOR MACRO
**        ?W\?L?PL **.DUSR,__ ,?I=      ;;ASSIGN ?I TO MACRO VALUE
**          .RDX ?L?IR                  ;;RESTORE INTERNAL RADIX
**
```

```
**       ^1 ^2 LIT[,0+0+0+0+0]              ;;ALLOC WORD HERE AND IN LPOOL
**.ENDC DONE
**
**.DO (.PASS==0)!(?L?NI<>0)
**       ^1 ^2 LIT[^4 ^5 ^6 ^7 ^8 ^9]
  .ENDC DONE
**       ?L?DI    ^1,^2,,0 1 2 3 4 5 6 7 8 9 A B C D E F G H I J
**       .DUSR    ?F = LIT[^4 ^5 ^6 ^7 ^8 ^9]      ;;FAKE LIT CALL
**[DONE]
%

; ?L?XR  -  PROCESS MEMEORY REFERENCE

         .MACRO   ?L?XR
**
**       .PUSH    .NOLOC
**       .NOLOC   1
**
**       .DUSR    ?D = 1             ;;SET TO GO INDIRECT
**
**.DO T'^3==(1B1+1B12)   ;;IF SYMBOL IS EXTERNAL
**.ENDC PUNT             ;;GO INDIRECT ELSE
**
**.DO T'^3==(1B1+3.B12) ;;IF IS PAGE ZERO SYMBOL THEN
**       .DUSR    ?D = 0             ;;GO DIRECT
**.ENDC X
**
**       .DUSR    ?F = .
**
**.DO (T'^3&15.)<>(T'?F&15.)     ;;IF RELOCATIONS DIFFER
  .ENDC PUNT                    ;;GO TRY TO REUSE A LITERAL

**.DO ^3<>0                             ;;DO WITHOUT PASS 1 ERRORS
**       .DUSR   ?D = ^3-.
**.ENDC
**
**[X]
**
**.DO (-128.<=?D)&(?D<=0)                ;;IF BACKWARD & IN RANGE
**   .DO (.PASS==0)!(?L?NI<>0)
**       ^1  ^2 ^3 ^4 ^5 ^6 ^7 ^8 ^9     ;;GO DIRECT
**   .ENDC XY
**   .DO ''^2''==''
            ^1       ^3 ^4 ^5 ^6 ^7 ^8 ^9
**   .ENDC XY
            ^1       ^2,^3 ^4 ^5 ^6 ^7 ^8 ^9
**   [XY]
**.ENDC DONE
**
**[PUNT]
**
**.DO (.PASS==0)!(?L?NI<>0)
**       ^1  ^2 @LIT[^3 ^4 ^5 ^6 ^7 ^8 ^9]
**.ENDC XY
**       ?L?DI    ^1,^2,@,0 1 2 3 4 5 6 7 8 9 A B C D E F G H I J
**       .DUSR    ?F = LIT[^3 ^4 ^5 ^6 ^7 ^8 ^9]  ;;FAKE LIT CALL
  [XY]

**[DONE]
**       .NOLOC   (.POP)
%

; ?L?DI  -  FORMAT X/XX INSTRUCTION LISTING
```

```
        .MACRO  ?L?DI
**
**      .PUSH   ?A
        .PUSH   ?B
        .PUSH   ?C
        .PUSH   ?D
**
**      .DUSR   ?D = ?L?CL
**
**        .RDX ?L?IR
**.DO ?G\?D&?L?G1==0             ;;IF WAS NOT GENENED THEN
**      .DUSR   ?D = ?L\?D        ;;GET ACTUAL NUMBER
**.ENDC
**
**      .DUSR   ?B = ?D/10        ;;BREAK LIT INTO DIGITS
**      .DUSR   ?C = ?D-(?B*10)+4.
**      .DUSR   ?A = ?B/10
**      .DUSR   ?B = ?B-(?A*10)+4.
**      .DUSR   ?A = ?A+4.
**
**        .RDX ?L?UR
**
**      ?L?D2   ^1,^2,^3,^?A,^?B,^?C
**
**      .DUSR   ?D = .POP
**      .DUSR   ?C = .POP
**      .DUSR   ?B = .POP
**      .DUSR   ?A = .POP
%

    D2  -  ACTUAL DEPOSIT THE FORMATED LINE

        .MACRO ?L?D2
**
**      .PUSH   .NOLOC
**      .NOLOC  1
**
**.DO ''^2''==''           ;;IF NO REGISTER THEN
          ^1      ^3?L^4^5^6
**.ENDC DONE              ;;ELSE
          ^1      ^2,^3?L^4^5^6
**[DONE]
**
**      .NOLOC  (.POP)
%


; ?L?XX  -  PROCESS AN "XX" INSTRUCTION
;
; CALL:
; ?L?XX <INSTR>,<AC>,<TARGET (INCLUDING BOOKKEEPING
;       SYMBOL AS LAST ATOM) >

        .MACRO  ?L?XX

    .MCALL==0           ;;THE FIRST TIME,
        ?L?1X                           ;;MAY NEED TO INIT
**.ENDC
**
**      .DUSR  ?L?UR = .RDX             ;;SAVE USER RADIX
**
**      .PUSH ?D                        ;;SAVE TEMPS
```

```
**
**
**.DO '^3'&-1B7=='#'                ;;IF "#" USAGE
 *      ?L?XN    ^1,^2,^3 ^4       ;;HANDLES #
 *.ENDC DONE

**
**.DO '^3'&-1B7=='='                ;;IF "=" USAGE
**      ?L?XE    ^1,^2,^3 ^4       ;;HANDLES "="
**      .DUSR    ?D = 5.           ;;SET TO HANDLE BOOKKEEPING
**.ENDC REACH
**
**      ?L?XR    ^1,^2,^3          ;;HANDLES MEMEORY REFERENCE
**
**.DO (-128.<=?D)&(?D<=0)                    ;;IF WENT DIRECT THEN
**.ENDC DONE                                 ;;DONE ELSE
**
**      .DUSR    ?D = 4.           ;;SET 4TH ARG AS BOOKKEEPER
**
**[REACH]
**
**.DO .PASS==0             ;;ON PASS 1 DO THE BOOKKEEPING
**
**  .DO '^?D'<>''          ;;IF BOOKKEEPING SYMBOL GIVEN THEN
**      ?L?XO    ^?D       ;;USE IT ELSE
**  .ENDC NOBOOK           ;;USE THE ARGUMENT PRECEEDED BY A
**      .DUSR    ?D = ?D-1
**      ?L?XO    .^?D      ;;A PERIOD
**  [NOBOOK]
 *
  .ENDC
**
**[DONE]
**
**      .DUSR   ?F = .POP                    ;;RESTORE TEMPS
**      .DUSR   ?D = .POP
**
%


; ?L?XO  -  XX INSTRUCTION BOOKKEEPING DEPARTMENT

        .MACRO  ?L?XO
**
**      .DUSR    ?F = ?L?CL-1                ;;GET NUMBER OF LITERAL GENED
**
**      .RDX     ?L?IR                       ;;ESTABLISH INTERNAL RADIX
**
**.DO (?G\?F&?L?CO==0)&(T'^1&1B5==0)     ;;IF LIT CAN OPTIMIZE USAGE
**.ENDC DONE                                ;;THEN WE DON'T NEED TOO ELSE
**
**.DO T'^1&1B5==0                  ;;IF 1ST TIME THEN
**      .DUSR    ^1 = ?F          ;;SAVE LITERAL NUMBER
**.ENDC DONE
**
 *.DO ^1>=?L?PL                    ;;IF PREVIOUS USAGE NOT YET
  .ENDC REUSE                      ;;POOLED THEN USE IT ELSE
**
**.DO .-?L\^1>=128.                ;;IF PREVIOUS POOL OF IT OUT OF RANGE
**      .DUSR    ^1 = ?F          ;;THEN SET TO USE NEW POOL
**.ENDC DONE
**
```