# APPENDIX F

I. The hexadecimal (16) entry mode: this mode may be used in the same fields as any other radix entry mode, and is treated exactly the same. The legal hex characters are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. A group of alpha-numeric characters in a hex field is symbolic if, and only if, the group begins with an alphabetic character and contains an alphabetic character which is not a legal hex character. Any group of alpha-numeric characters which are non-symbolic in a hex field is a number in base 16.

Examples:

A symbol in a hex field (16) A013G

A number in a hex field (16) 95AB = $(112653)_8$

A number in a hex field (16) FF6 = $(7766)_8$

If a number in a hex field contains an illegal hex character, the character is discarded into the STRAP-1 error mark field and the remaining characters are converted.

Examples:

1. (16) 95ZA = (16) 95A and (16) 95ZWYA = (16) 95A

2. (16) DD (Bu), (.31) FFFF (.45)3C
   or DD (Bu), (.31,16) FFFF (16, .45)3C places ones in bits (16 - 31) and (42 - 45) of a full word and zeroes everywhere else in that word.

II. Pseudo Operations:

1. XM "Index Multiples of" NAME|XM,A,C

The symbol, A, represents a multidimensional array defined by a DR card e.g. A|DR(dds), ($L_1$, $L_2$,..., $L_M$). The quantity C, if given will be entered in the count fields of the index words described below. If C is not given then $L_1$ is entered in all of the count fields in the place of C. The pseudo operation XM will create ($L_2 + L_3 + \cdots L_M$) index words containing "Index Multiples of A" in the value fields in the following order:

| Location | Word Format | |
|---|---|---|
| NAME | XW, 0, C, NAME + 1 | |
| NAME + 1 | XW, F, C, NAME + 2 | |
| NAME + 2 | XW, 2F, C, NAME + 3 | |
| ⋮ | ⋮ | |
| NAME + $L_2$ -2 | XW, F ($L_2$-2), C, NAME + $L_2$ -1 | |
| NAME + $L_2$ -1 | XW, F ($L_2$-1), C, NAME + 0 | ,4 |
| NAME + $L_2$ | XW, 0, C, NAME + $L_2$ + 1 | |
| NAME + $L_2$ + 1 | XW, F$L_2$, C, NAME + $L_2$ + 2 | |
| NAME + $L_2$ + 2 | XW, 2F$L_2$, C, NAME + $L_2$ + 3 | |
| ⋮ | ⋮ | |
| NAME + $L_2$ + $L_3$ -1 | C, NAME + $L_2$ | ,4 |
| ⋮ | ⋮ | |
| NAME + $L_2$ + $L_3$ + ... + $L_M$ -1 | XW, F$L_2L_3$...($L_M$-1), C, NAME + $L_2$ + ... + $L_{M-1}$ | ,4 |

Notes: 1. F = $L_1$ times the field length of A, if A represents full words F = $L_1$.0

2. A flag bit of 1 is inserted in each index word at NAME + $L_{i+1}$ -1

3. The location counter will be rounded up to the nearest full word for the symbol NAME.

4. The operation XM is completely ignored if the symbol referred to is not defined or has only one dimension.

Example:

```
A|DR(N), (I,J,K)
I|SYN, 9
J|SYN, 4
K|SYN, 3
Z|XM, A
```

Note: The order given is not necessary and code may intervene. Of course, the dimensions I, J, and K could have been given explicitly.

The address of the symbol Z will be a full word address and the following 7, (4 + 3) index words will be associated with Z.

```
Z          XW,  0.0, 9, Z + 1.0
Z + 1.0    XW,  9.0, 9, Z + 2.0
Z + 2.0    XW, 18.0, 9, Z + 3.0
Z + 3.0    XW, 27.0, 9, Z + 0.0, 4
Z + 4.0    XW,  0.0, 9, Z + 5.0
Z + 5.0    XW, 36.0, 9, Z + 6.0
Z + 6.0    XW, 72.0, 9, Z + 4.0, 4
```

This block of index words can then be used as follows:

Suppose index registers XI, XJ, and XK contain the quantities i, j, and k, respectively, and we wish to load the element $A_{ijk}$ using index register, ⌀5. (i = 0,1,...9; j = 0,1,2,3; k = 0,1,2).

| STATEMENT | | |
|---|---|---|
| LX, ⌀5, Z + J.0 (XK) | " | kIJ goes to value of X5. |
| V + , ⌀5, Z.0 (XL) | " | jI + kIJ to value of X5. |
| V + , ⌀5, XI | " | i + jI + kIJ to value of X5. |
| L, A(⌀5) | " | $A_{ijk}$ to accumulator. |

2. PRNID "Print ID" | PRNID, NAME PHONE ETC.

When this card is detected in pass 1, its entire contents are immediately printed both on-line and off-line, in order to identify both listings.

3. TAIL "tail a character | TAIL,X

Each symbol occurring in the name and address fields following this instruction is treated as if it were terminated by the character X. Such symbols must consist of <u>five</u> or fewer alpha-numeric characters. X must be a single alpha-numeric. A second tail card immediately overrules the preceeding one.

4. UNTAIL "untail a character" | UNTAIL or UNTAIL,X

Restores conditions to normal after a TAIL. X is ignored.

5. NØPUN "no punch" | NØPUN

Suppresses all punching from this point on. Punching conditions may be returned to normal by use of a PUNNØR.

6. SEM "suppress error marks" | SEM, $E_1$, $E_2$, ... $E_n$ or | SEM

If the address field is blank, the printing of all error marks except certain punch error marks is suppressed from this point on. If one or more error mark characters appear between commas in the address field, these and only these particular error mark characters are suppressed.

7. REM "restore error marks" | REM, $E_1$, $E_2$, ..., $E_n$ or | REM

This card is used to restore conditions to normal after a SEM. If the address field is blank, the printing of all error marks is resumed. If one or more error mark characters appear between commas in the address field, these and only these particular error mark characters are restored.

On both SEM and REM, null fields (i.e., two commas with nothing between them) are not allowed; if used, no character occurring after the null field will be noticed by STRAP-1. Thus

| SEM, , A, B, C is treated as | SEM

| SEM, A, B,, C is treated as | SEM, A, B

etc. A null field is thus considered a termination.

Neither SEM nor REM cards are themselves error marked.

8. INDMK "Indicator mask" | INDMK, $A_1$, $A_2$, $A_3$,..., $A_n$

$A_i$ are symbols or system symbols for any of the indicators; this instruction causes the location counter to be rounded to a full word, and a 64-bit word is then constructed with 1's in the positions corresponding to the indicators named.

9. PUNSYM "punch cards for symbols" PUNSYM, $A_1$, $A_2$, $A_3$,..., $A_n$

The $A_i$ are any legal symbols occurring elsewhere in the program, whether defined or undefined. After the entire binary deck has been punched, two cards will be punched for each $A_i$, as follows:

(a) If $A_i$ is not an array, the card format will be:

| name | 10 statement |
|---|---|
| AI | SYN(M, FL, BS),(8)±XXXXXX.XX±(8)XXXXXXXX |
| AI | SLC, AI |

where the X's represent octal digits.

(b) If $A_i$ is an array, the card format will be:

| name | 10 statement |
|---|---|
| AI | $SLC, (8)XXXXXXXX |
| AI | DR(M, FL, BS), ($D_1$, $D_2$, $D_3$,..., $D_n$) |

where the X's represent octal digits, and the $D_i$'s are dimensions.

These cards will contain in columns 73-80 the last ID specified by a PUNID card in the program, if any. These cards can be used in reassembling portions of one's code where the symbols $A_i$ have been defined by STRAP.