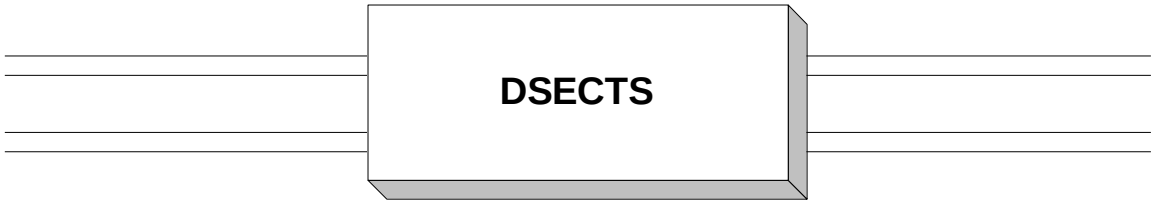

A DSECT or Dummy Section provides the programmer with the ability to describe the layout of an area of storage without reserving virtual storage for the area that is described. The DSECT layout can then be used to reference any area of storage which is addressable by the program. Think of the DSECT as a template, or pattern, which can be "dropped" on any area of storage. Once the DSECT is "positioned", the symbolic names in the DSECT can be used to extract data from the underlying storage area.

How is the DSECT positioned? This is a two-step process. First, a register is associated with the DSECT name by coding a USING directive. Loading the register with the storage address completes the process. Consider the sample code below.

```
CUSTOMER DSECT
FNAME    DS    CL4
LNAME    DS    CL5
BALANCE  DS    PL5
MAIN     CSECT
         ...
         USING  CUSTOMER,R7
         LA     R7,TABLE
         MVC    NAME1,FNAME
         MVC    NAME2,LNAME
         ...
TABLE    EQU    *
         DC     CL4'FRED'
         DC     CL5'SMITH'
         DC     PL5'432.98'
         ...
NAME1    DS     CL10
NAME2    DS     CL20
```

The "CUSTOMER" DSECT is created by coding the DSECT directive which indicates the beginning of the dummy section. The general format for this directive is listed below.

```
NAME           OPERATION     OPERAND

Any Symbol     DSECT       Not required
```

The pattern for the DSECT is created using a series of DS directives to describe a collection of fields in storage. The end of the DSECT is indicated by the beginning of the CSECT. While the DSECT can be coded almost anywhere in the program, it is a common practice to place any DSECTs you will need above the main control section, as in the example above. In order to code a DSECT in the "middle" of a control section, the CSECT directive would appear twice as indicated below.

```
MAIN     CSECT
         ...
CUSTOMER DSECT
FNAME    DS    CL4
LNAME    DS    CL5
```

```
BALANCE  DS    PL5
MAIN     CSECT
```

The second CSECT directive indicates that the “MAIN” control section is being continued.

After defining the dummy section, it must be associated with an available register in a USING directive. In the example code above, the statement “USING CUSTOMER,R7” associates register seven with the “CUSTOMER” dummy section. Addressability is established when the register is loaded with the address of the table. At that point, FNAME contains the value “FRED”, and LNAME contains the value “SMITH”. Loading the register “drops” the DSECT on the storage area whose address is contained in the register.

You should consider the power of the above technique. There were no symbolic names associated with the table when it was defined. By creating a DSECT, we are able to dynamically assign symbols to an area of storage and retrieve the data the area contains. DSECTs have many uses in assembly language including array processing, locate mode I/O (See **SEQUENTIAL FILE PROCESSING**), and data structure creation and manipulation. Most sophisticated assembler programs involve DSECTS to some degree. Because of their importance, we will consider the use of DSECTs in array processing.

Loading an Array With Data

In the following example we will load an “empty” storage area with data that is read from a file. We will use the same DSECT as in the previous example.

```
CUSTOMER DSECT
FNAME    DS     CL4
LNAME    DS     CL5
BALANCE  DS     PL5
CRECLEN  EQU    *-CUSTOMER
CRECLENF DC     A(CRECLEN)       REC LEN AS A FULLWORD
MAIN     CSECT
         ...
         USING  CUSTOMER,R7
         LA     R7,TABLE         POSITION THE DSECT
         LA     R4,CRECLEN       LOAD REC LENGTH INTO R4
*        L      R4,CRECLENF      LOAD REC LENGTH INTO R4
         LA     R5,0             START REC CNT IN R5
LOOP     EQU    *
         GET    FILEIN,RECIN     READ A RECORD
         A      R5,=F'1'         ADD 1 TO THE REC CNT
         MVC    FNAME,FNAMEIN    MOVE DATA ...
         MVC    LNAME,LNAMEIN    ... FROM RECIN ...
         ZAP    BALANCE,BALIN    ... TO THE TABLE
         LA     R7,CRECLEN(R0,R7)  BUMP THE DSECT
         B      LOOP             BRANCH BACK FOR MORE RECS
STCNT    EQU    *
         ST     R5,RECCNT
         ...
TABLE    DS  100CL(CRECLEN) EMPTY AREA FOR 100 RECORDS
RECCNT   DS     F                RECORD COUNT
```

The address of the empty table is placed in R7 by the first LA instruction. This has the effect of positioning the DSECT at the beginning of the table area, since the USING directive associated register seven with the DSECT name. The length of a typical table entry is computed as “CRECLEN” using a EQUATE directive and placed in R4 by the LA instruction. “CRECLENF” is

also created as a fullword that contains the record length. Alternatively, the record length could have been loaded into R4 with the next line (which has been removed by commenting). A record is retrieved and the data is moved from the input buffer, RECIN, and placed in the table using the symbolic names that were defined in the DSECT.

In order to move the DSECT to the next empty table entry we code the following line,

```
LA      R7,CRECLEN(R0,R7)  BUMP THE DSECT
```

This type of statement is used quite often in DSECT processing and so we consider it in some detail. The statement uses explicit addressing and the technique is considered acceptable practice. The effect is to compute the second operand address explicitly from the base register (R7), index register (R0), and displacement (CRECLEN). When R0 is specified as an index register, that component of the address computation is ignored. The “effective” address that would be computed consists of the contents of R7 plus the displacement of 14 represented by CRECLEN. Since R7 contains the address of a table entry, adding 14 to R7 will create the address of the next table entry. The DSECT has been “bumped” down to the next table entry.

In writing the program, we assume that FILEIN contains 100 or fewer records so that the table will contain all the records in the file. We also assume that the FILEIN DCB contains EODAD=STCNT so that we store the record count whenever we run out of records in FILEIN.