C-TYPE

CHARACTER DATA

Character data is created by specifying a “C” for the type in a DC or DS declarative. In this format, each character occupies 1 byte of storage. Characters are represented using the EBCDIC encoding sequence where each character is presented using 8 bits. As a result, there are $2^8 = 256$ possible bit patterns or characters which can be formed.

A character field can be created using either of the following formats,

```
name            DC      dCLn'constant'
```

or

```
name            DS      dCLn
```

where ‘name’ is an optional field name
‘d’ is a duplication factor used to create consecutive copies of the field (default = 1 copy)
‘C’ represents the character data type
‘L’ represents an optional length (default = 1 byte)
‘n’ is the number of bytes in the field if L is coded
‘constant’ is an initial value of the field in character format.

It is important to note that if a field name is specified, it will represent the address of the first byte of the field. Additionally, the name of the field will be associated with a length attribute which equals the number of bytes in the field. If the “Ln” construction is omitted in a DS, the length will default to 1 byte. If the “Ln” construction is omitted in a DC, the length of the required constant determines the field length. The length of a field defined using DC is limited to a maximum of 256 bytes and the length of a field defined using DS is 65,535 bytes.

When specifying a constant, the length of the constant and the length of the field may differ in size. If the length of the constant is shorter than the field length, the assembler will pad the constant on the right with blanks to fill up the field. If the constant is longer than the field length, the constant will be truncated on the right so that it will fit inside the field. The assembler does not generate a warning message when this occurs.

During assembly, fields that are defined consecutively in the source code with DC’s or DS’s, are assigned consecutive storage locations in the program according to the value of the location counter maintained by the assembler. For example, in the fields below, if FIELDA is associated with address x’1000’, then FIELDB is located at x’1003’ and FIELDC is located at x’100C’.

LOCATION

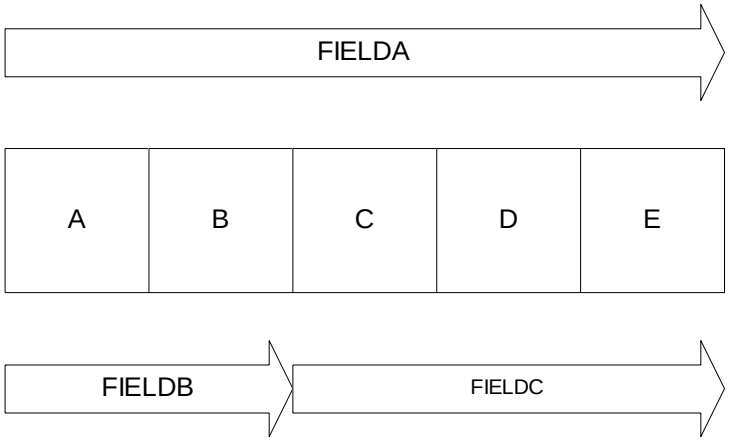
```
1000      FIELDA    DS    CL3
1003      FIELDB    DS    CL9
100C      FIELDC    DS    CL3
```

An exception to sequential allocation of fields occurs if the duplication factor is specified as 0. In this case the location counter is not advanced and the next field redefines the previous field. Consider the following example and note the location counter values.

LOCATION

```
1000     FIELDA   DS    0CL5
1000     FIELDB   DC     CL2'AB'
1006     FIELDC   DC     CL3'CDE'
```

In this case, FIELDB and FIELDC are allocated addresses "inside" FIELDA.



Examples

Some Typical DS's and DC's:

```
A         DS    CL8       An 8-byte character field
B         DS    C         A 1 byte character field
C         DS    CL2000    A 2000 byte field
NAME      DS   0CL20      A field that is subdivided
LNAME     DS    CL10      The first subfield
FNAME     DS    CL10      The second subfield
BLANK1    DC    CL80' '   A blank 80-byte field, L'BLANK1 = 80
BLANK2    DC  80C' '      80 consecutive 1-byte fields, L'BLANK2 = 1
D         DC    CL2'ABC'  Constant too long - right truncation, D= 'AB'
E         DC    CL5'AB'   Constant too short – blank padded,E = 'AB   '
F         DC   3C'XY'     L'F = 2, 3 fields created (XYXYXY), F = 'XY'
          DC    C'COST'   An unnamed, initialized field
```

Note - L' indicates the length attribute of the field it qualifies.

Tips

1. A common coding error that occurs when defining a field with a constant, is to choose DS instead of DC. Consider the following example,

```
LNAME   DS    CL10'SMITH'
```

On first glance, the code appears correct, and in fact, it is syntactically correct. The assembler will not complain about the construction. But the constant is treated as a comment and the field remains uninitialized.

2. Pay careful attention to the constants you provide. If the constant is too long, it will be truncated. If it is too short, it will be padded with blanks.

3. Learn to recognize a working subset of characters and their hexadecimal equivalents. The list below, consisting of the alphabet, a blank, the digits, and some punctuation symbols, is a good start.

Character	Hex Equivalent	Character	Hex Equivalent
A	C1	0	F0
B	C2	1	F1
C	C3	2	F2
D	C4	3	F3
E	C5	4	F4
F	C6	5	F5
G	C7	6	F6
H	C8	7	F7
I	C9	8	F8
J	D1	9	F9
K	D2	BLANK	40
L	D3	,(COMMA)	6B
M	D4	. (PERIOD)	4B
N	D5	* (ASTERISK)	5C
O	D6		
P	D7		
Q	D8		
R	D9		
S	E2		
T	E3		
U	E4		
V	E5		
W	E6		
X	E7		
Y	E8		
Z	E9		