

**Chapter
2**

**DEFINING
DATA CONSTANTS
AND SYMBOLS**

*Get on the
Fast Track!*



TM

**SYS-ED/
Computer
Education
Techniques, Inc.**

Objectives

You will learn:

- Data types.
- Defining constants.
- Truncation and padding.
- Alignment - constants and boundary.
- CNOP instruction.
- Defining literals.
- Defining storage: DS instruction.
- Defining symbols: EQU instruction.

1 Data Types

The Assembler Language provides for the use of the common data types.

These include:

	Example	PL/1	COBOL Picture
B Binary	B`10110010'	bit string	
C Character	C`HELLO'	CHAR	X(5)
X Hexadecimal	X`F3'		
F Fullword	F`158'	FIXED BIN(31,0)	9(8) COMP
H Halfword	H`15'	FIXED BIN(15,0)	9(4) COMP
P Packed decimal	P`523'	FIXED DEC0)	9(3) COMP-3
Z Zoned decimal	Z`523'		9(3) DISPLAY
A Address	A(Location)		

Floating point data type also can be defined.

2 Defining Constants

2.1 DC Instruction

Constants are defined using the DC instruction. This is an assembler instruction; not a machine instruction.

It is not a standard practice to assign a label to each constant in order that it may be referenced by other instructions.

The usual subfields of the DC operands are:

datatype value

Examples:

			Implied length (bytes)
BIN	DC	B`00010110`	1
TENSTR	DC	CL10`STRING`	10
F3	DC	X`FFF3`	2
FULL	DC	F`43`	4
KEY	DC	H`21`	2
COMP3	DC	P`183`	2 i.e. 183C
DEC	DC	Z`253`	3
ADDR	DC	A(LOCATION)	4

The second example defines a constant ten bytes long with an initial value of `STRING' and also highlights two other important considerations:

1. Specifying the length of a constant.
2. Padding Characters.

2.2 Subfields

The length of a constant is one of four permitted subfields of the DC instruction.

The subfields are:

duplication	type	length	value
-------------	------	--------	-------

Length can be explicitly coded as L5, L3, etc:

DC CL5`1' Defines a five-byte character constant.

DC ZL8`0' Defines an eight-byte zoned decimal constant.

If L is omitted, the constant takes the length implied by the value.

Duplication permits a constant to be repeated.

Therefore, OP3 DC 3C`ABCD' each of which has implied an length of 4 is the same as:

```
OP3 DC C`ABCD'
      DC C`ABCD'
      DC C`ABCD'
```

It is also permissible to have multiple operands for a constant.

These may be of different types:

```
OPMIX DC C`A1', H`2', X`FF'
OP4    DC F`1, 21, 141, 200'
ADD3   DC A(ONE, TWO, THREE)
```

The last two examples are the same as:

```
OP4    DC F`1'
DC     F`21'
DC     F`141'
DC     F`200'
ADD3   DC A(ONE)
DC     A(TWO)
DC     A(THREE)
```

3 Truncation and Padding

Padding is performed in a predictable manner.

Character data fields are padded with blanks on the right.

Numeric data fields are padded with zeros on the left:

- binary zeros for B,X,H,F,P and address constants - A.
- EBCDIC zeros for Z.

Truncation also corresponds to the implementation with COBOL and PL/1.

- Character data is truncated on the right.
- Numeric data and address are truncated on the left.

However, fixed point constants - H and F, are not truncated. Instead they are flagged if significant bits are lost through truncation.

3.1 DC Examples

NAME	DC	C`JOHN'
NAME	DC	CL6`JOHN'
A	DC	CL5`1'
A	DC	ZL5`1'
B	DC	3CL5`ABCD'
C	DC	PL3`452'
D	DC	PL3`-452'
E	DC	P`452'
F	DC	P`8452'
G	DC	PL2`8452'
H	DC	F`19'
I	DC	F`-19'
J	DC	H`19'
K	DC	H`-19'
ADDR1	DC	A(LOCATION)
ADDR1	DC	V(SUBRTN2)

4 Alignment - Constants

At minimum, constants are aligned on byte boundaries unless lengths are defined as bit-length specifications - L.n e.g. L.3, L.12.

The bit fields are aligned on the leftmost, high-order end. Any padding is filled with zeros.

Fullword constants are aligned on fullword boundaries.

Halfword constants are aligned on halfword boundaries.

It will be necessary to take into account any slack storage when calculating the length of storage fields.

5 CNOP Instruction

Since alignment may cause slack storage bytes to be inserted by the assembler, a program might contain some bytes with unpredictable content.

In order to prevent this, some programmers use the CNOP instruction. The CNOP instruction will insert bytes unless the program is currently on the required boundary.

The extra bytes are of the form 0700, where 07 is the operation code for CNOP.

Example:

```
CNOP 0,4
```

This instruction will cause alignment on a fullword boundary.

The first operand may be 0, 2, 4, or 6.

The second operand may be 4 or 8.

Some of the valid values for the CNOP command are:

Values	Specify
0,4	Beginning of a word.
2,4	Middle of a word.
0,8	Beginning of a doubleword.
2,8	Second halfword of a doubleword.
4,8	Middle third halfword of a doubleword.
6,8	Fourth halfword of a doubleword.

There are other values for quad addresses.

6 Defining Literals

Literals differ from constants in several ways.

A literal represents data which is a constant represented as a relocatable address by its label.

```
L 5,=F'25'
                                have the identical affect.
L 5,F25
.
.
.
F25 DC F'25'
```

Literals cannot be used as terms in expressions and do not have relocatable values.

Literals appear in the Literal Pool, which is part of the listings produced by the assembler.

The addresses of the literals, which are a location in the literal pool, not their values are assembled in the machine instructions.

Examples:

```
L 4,=F'24'
L 5,=A(SUBA)
MVC MESS(20),=C`SYSED ASSEMBLER COURSE`
```

Consider what the result would be if the following code was implemented:

```
MVC MESS(24),=C`SYSED ASSEMBLER COURSE`
```

7 Defining Storage: DS Instruction

Storage can be reserved by the DS instruction. This reserves areas of storage and assigns labels to them. The storage is used for work areas, buffers, etc., and does not have to have a nominal value as with the DC instruction.

The length of storage can be implicit or explicit.

The implicit values are:

F	4 bytes
H	2 bytes
A	4 bytes
5F	20 bytes

Examples:

Explicit lengths are:

CL10	10 bytes
XL128	128 bytes

Large areas are reserved with the duplication factor.

Examples:

18F	18 fullwords (72 bytes)
9D	9 doublewords which are used for save areas.
3CL1000	3000 bytes

Nominal values are optional:

```
CHARI DS C'HELLO, GOOD EVENING, AND WELCOME'
```

The assembler will calculate the length of CHARI as 32 bytes.

The DS instruction does not generate code; it only reserves bytes of storage. If a nominal value is specified, that value will not be assigned to those bytes. This compares with the VALUE clause in the FILE SECTION of a COBOL program.

Examples:

NUM	DS	F
BALANCE	DS	PL4
MESSAGE	DS	CL20
MESSAGE	DS	C'SYSED ASSEMBLER COURSE'
REC	DS	CL80
SAVE	DS	18F
TOTALS	DS	10PL5
ADDR	DS	4CL20

7.1 Alignment - Boundary

A useful feature of DS is the capability to force boundary alignment.

This is done with a duplication factor of zero:

- 0H aligns on halfword boundary.
- 0F aligns on fullword boundary.
- 0D aligns on doubleword boundary.

The zero duplication also is used when setting out in storage a record which contains group and elementary fields:

	EMP NUM		DATE STARTED	DEPT		SALARY	
bytes 5	10	11	32	33	39	41 51	58

	Assembler		PL/1		COBOL
REC	DS 0CL80		DCL 1 REC,		01 REC-C
	DS CL4		2 ..		03 FILLER ..
EMPNUM	DS CL6		2 EMPNUM ..		03 EMPNUM-C ..
NAME	DS CL22		2 NM ..		03 NAME-C ..
DATE	DS 0CL6		2 DDATE,		03 DATE-C
DAY	DS CL2		4 DDAY ..		05 DAY-C ..
MONTH	DS CL2		4 MMTH ..		05 MONTH-C..
YEAR	DS CL2		4 YYR ..		05 YEAR-C ..
DEPT	DS CL3		2 DEPT ..		03 DEPT-C ..
	DS CL9		2 ..		03 FILLER ..
SALARY	DS PL8		2 SAL ..		03 SALARY-C ..

The 80 byte area REC is not aligned on a word boundary.

This could be achieved by an initial:

```
DS      OF
```

Any bytes which are added for alignment purposes are not initialized. If they are to appear in printer fields, it will be necessary to establish their contents. Similarly, the initial contents of the reserved storage area are dependant on the version of the operating system.

8 Defining Symbols: EQU Instruction

The EQU instruction permits the assignment of absolute or relocatable values to symbols.

Absolute symbols are symbols whose values are a fixed number which will be the same regardless of where the program is loaded in storage. The RO can be used to represent the number 0 in RR and RX instructions.

The value of relocatable symbols depends on where the program is loaded; they represent storage addresses. When a relocatable symbol is used in an Assembler Language program, it is replaced by a base and displacement.

Symbols may be assigned to:

registers	other symbols	expressions
-----------	---------------	-------------

Since each symbol is a mnemonic, this provides freedom to use more meaningful terms in the coding.

```

AREA      DS XL4000
REG3      EQU 3
R3        EQU 3
WHERE     EQU AREA+100

```

```

TABSTART DS 40CL80
TABEND   DS H'0'

```

```

TABLEN EQU TABEND-TABSTART i.e. difference between addresses.

```

All symbols are listed in the cross reference listings.