Crucial!!You'll use this all the time...

- **BYTE (8 bits)**: Short form - **B**
- **SBYTE (8 bits, signed)**: Short form - **SB**
- **WORD (16 bits)**: Short form - **W**
- **SWORD (16 bits, signed)**: Short form - **SW**
- **DWORD (32 bits)**: Short form - **D**
- **SDWORD (32 bits, signed)**: Short form - **SD**
- **FWORD (48 bits)**: Short form - **FW**
- **QWORD (64 bits)**: Short form - **Q**
- **TBYTE (80 bits)**: Short form - **T**
- **REAL4 (32-bit floating-point)**: Short form - **F**
- **REAL8 (64-bit floating-point)**: Short form - **FF**
- **REAL10 (80-bit floating-point)**: Short form - **FT**

A data definition statement in assembly language is used to reserve memory for a variable and to specify its data type. The general syntax for a data definition statement is as follows:

[label] directive value

Where:
- **label** is an optional name for the variable.
- **directive** is the data type of the variable.
- **value** is the initial value of the variable.

For example, the following data definition statement reserves 4 bytes of memory for a variable named count and initializes it to the value 12345:

```
count DWORD 12345
int count = 12345;
```

So, **label is variable, directive is the datatype, and value is the value.**

The following are some other examples of data definition statements in assembly language:

```
message DB "Hello, world!"
age BYTE 25
salary SDWORD 100000
```

The **DUP operator takes two arguments: a count and a value.** The count is an integer expression that specifies the number of times to duplicate the value.