ASSEMBLY - VARIABLES

NASM provides various **define directives** for reserving storage space for variables. The define assembler directive is used for allocation of storage space. It can be used to reserve as well as initialize one or more bytes.

Allocating Storage Space for Initialized Data

The syntax for storage allocation statement for initialized data is −

```
[variable-name]    define-directive    initial-value   [,initial-value]...
```

Where, *variable-name* is the identifier for each storage space. The assembler associates an offset value for each variable name defined in the data segment.

There are five basic forms of the define directive −

Directive	Purpose	Storage Space
DB	Define Byte	allocates 1 byte
DW	Define Word	allocates 2 bytes
DD	Define Doubleword	allocates 4 bytes
DQ	Define Quadword	allocates 8 bytes
DT	Define Ten Bytes	allocates 10 bytes

Following are some examples of using define directives −

```
choice  DB 'y'
number  DW 12345
neg_number DW -12345
big_number DQ 123456789
real_number1 DD 1.234
real_number2 DQ 123.456
```

Please note that −

- Each byte of character is stored as its ASCII value in hexadecimal.
- Each decimal value is automatically converted to its 16-bit binary equivalent and stored as a hexadecimal number.
- Processor uses the little-endian byte ordering.
- Negative numbers are converted to its 2's complement representation.
- Short and long floating-point numbers are represented using 32 or 64 bits, respectively.

The following program shows the use of define directive −

```
section .text
   global _start          ;must be declared for linker (gcc)

_start:                   ;tell linker entry point
   mov edx,1    ;message length
   mov ecx,choice         ;message to write
   mov ebx,1    ;file descriptor (stdout)
```

```
   mov eax,4    ;system call number (sys_write)
   int 0x80     ;call kernel

   mov eax,1    ;system call number (sys_exit)
   int 0x80     ;call kernel

section .data
choice DB 'y'
```

When the above code is compiled and executed, it produces the following result −

```
y
```

Allocating Storage Space for Uninitialized Data

The reserve directives are used for reserving space for uninitialized data. The reserve directives take a single operand that specifies the number of units of space to be reserved. Each define directive has a related reserve directive.

There are five basic forms of the reserve directive −

Directive	Purpose
RESB	Reserve a Byte
RESW	Reserve a Word
RESD	Reserve a Doubleword
RESQ	Reserve a Quadword
REST	Reserve a Ten Bytes

Multiple Definitions

You can have multiple data definition statements in a program. For example −

```
choice   DB  'Y'    ;ASCII of y = 79H
number1   DW  12345   ;12345D = 3039H
number2    DD  12345679  ;123456789D = 75BCD15H
```

The assembler allocates contiguous memory for multiple variable definitions.

Multiple Initializations

The TIMES directive allows multiple initializations to the same value. For example, an array named marks of size 9 can be defined and initialized to zero using the following statement −

```
marks  TIMES  9  DW  0
```

The TIMES directive is useful in defining arrays and tables. The following program displays 9 asterisks on the screen −

```
section .text
   global _start        ;must be declared for linker (ld)

_start:                 ;tell linker entry point
   mov edx,9  ;message length
   mov ecx, stars ;message to write
   mov ebx,1  ;file descriptor (stdout)
   mov eax,4  ;system call number (sys_write)
   int 0x80  ;call kernel
```

```
   mov eax,1  ;system call number (sys_exit)
   int 0x80  ;call kernel

section .data
stars   times 9 db '*'
```

When the above code is compiled and executed, it produces the following result −

```
*********
```