Sub :

Day

Time : Date : / /

Md. NUR-E-Azam

ID: 1512268042

Section: 07

CSE-331 LAB

LAB-2

□ Topics to be covered

1. Creating variables
2. Creating Arrays
3. Create constants
4. Introduction to INC, DEC, LEA. instruction
5. Learn how to access memory.

□ Creating variables

Syntax for a variable declaration:

```
name DB value
name DW value
```

Sub :

Day

Time : Date : / /

□ Creating constants:

Syntax for a constants declaration:

mame EQU <any expression>

For example:

```
K EQU 5
Mov AX, K
```

□ Creating Arrays:

Some array deffnition examples:

```
a DB 48h, 65h, 6ch, 6ch, 6Fh, 00h
b ~~B~~DB 'Hello', 0
```

⇒ You can access the value of any element in array using square brackets, for example:

```
MOV AL, a[3]
```

Sub : Day Time : Date : / /

⇒ You can also use any of the memory index registers BX, SI, DI, BP. for example:

```
MOV SI, 3
MOV AL, a[SI]
```

⇒ If you need to declare a large array you can use DUP operator.

□ The syntax for DUP:

number DUP (value(s))

for example:

```
c DB 5 DUP (9)
```

is an alternative way of declaring:

```
c DB 9, 9, 9, 9, 9
```

One more example:

```
d DB 5 DUP (1, 2)
```

Sub :

Day

Time : Date : / /

☐ Memory Access:

To Access memory, we can use these four registers: Bx
SI
DI
Bp

[Bx + SI] [Bx + DI] [Bp + SI] [Bp + DI]	[SI] [DI] [d(16)] [Bx]	[Bx + SI + d8] [Bx + DI + d8] [Bp + SI + d8] [Bp + DI + d8]
[SI + d8] [DI + d8] [Bp + d8] [Bx + d8]	[Bx + SI + d16] [Bx + DI + d16] [Bp + SI + d16] [Bp + DI + d16]	[SI + d16] [DI + d16] [Bp + d16] [Bx + d16]

Sub : Day Time : Date : / /

☐ Declaring Array :

Array Name db size DUP (?)

☐ Value initialize :

arr1 db 50 dup (5,10,12)

☐ Index Values :

```
mov bx, offset arr0
mov [bx] , 6 ; inc bx
mov [bx+1] , 10
mov [bx+9] , 9
```

☐ OFFSET :

Offset is an assembler directive in x86 assembly language. It actually means "address" and is a way of handling the overloading of the "mov" instruction. Allow me to illustrate the usage -

Scanned with CamScanner

Sub :

Day

Time : Date : / /

1. mov si, offset variable

2. mov si , variable

As a matter of style, when I wrote X86 assembler I would write it this way

1. mov si, offset variable

2. mov si, [variable]

The square brackets aren't necessary, bu they made it much clearer while loadin the contents rather than the address.