Computer Organization & Assembly Language

Lab-4

Variables, data types, offset and LEA (Load Effective Address) in Assembly Language-1

dosseg ;dos segment

.model small

.stack 100h

.data ***Variables are define in .data directives in the program structure.***

.code

How to initialize variables?

Initializer Directive

Initializer

VariableName DataSize Value

Reserve Words such as (ADD, SUB, DIV,MUL, AL,BL,CL,DL,MOV, PUSH, POP etc.) are not allowed as Variable Names

Variables, data types, offset and LEA (Load Effective Address) in Assembly Language-2

```
dosseg ;dos segment
.model small
.stack 100h
.data
.code
```

Variables are define in .data directives in the program structure.

How to initialize variables?

Initializer Directive **Initializer**

VariableName DataSize Value

Var1	DB	Define Byte	1 byte, 8 bits.
	DW	Define Word	2 bytes, 16 bits.
	DD	Define Double Word	4 bytes, 32 bits.
	DQ	Define Quad Word	8 bytes, 64 bits.
	DT	Define Ten Bytes	10 bytes, 80 bits.

Variables, data types, offset and LEA (Load Effective Address) in Assembly Language-3

.data ***Variables are define in .data directives in the program structure.***

.code

How to initialize variables?

Initializer Directive

Initializer

VariableName	DataSize	Value	
Var1	db	49	49 is ASCII code of 1
Var1	db	?	When you don't to want to assign value then use ? Mark. Later we can initialize it in the .code section
Var1	db	'1'	Or you can directly initialize with value if you don't remember the ASCII code.
Var1	db	'1235$'	
Var1	db	'hello world!$'	

$ is Terminator or end point of String.

$ must be used to end of string

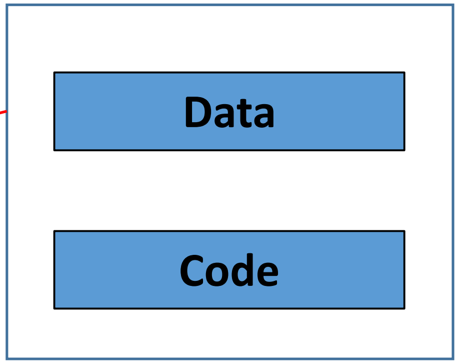
Variables, data types, offset and LEA (Load Effective Address) in Assembly Language-4

```
dosseg ;dos segment
.model small
.stack 100h
.data
Var1 db '1'
Var2 db ?
Var3 db '123$'
.code
Main proc
Mov ax,@data
Mov ds,ax

Main endp
End Main
```

It moves the memory location of @DATA into AX register (16 bits)

Moves data address to DS (heap memory – Fast Memory) so that data segment get initialized as heap memory to access variables fast.



Variables, data types, offset and LEA (Load Effective Address) in Assembly Language-5 (printing values

```
dosseg ;dos segment
.model small
.stack 100h
.data
Var1 db ‘1’
Var2 db ?
Var3 db ‘123$’
.code
Main proc
Mov ax,@data
Mov ds,ax
Mov dl,var1
Mov ah,2
INT 21h
mov ah,4ch
INT 21h
Main endp
End Main
```

**For printing VAR1 value,
move 8 bits value into dl – type matched – if we try to use dx then it will be an error of type mismatch**

Variables, data types, offset and LEA (Load Effective Address) in Assembly Language-6 (printing values

```
dosseg ;dos segment
.model small
.stack 100h
.data
Var1 db '1'
Var2 db ?
Var3 db '123$'
.code
Main proc
Mov ax,@data
Mov ds,ax
Mov dl,var1
Mov ah,2
INT 21h
Mov var2,bl
```

It will move VAR2 value onto bl register

```
mov ah,4ch
INT 21h
Main endp
End Main
```

Variables, data types, offset and LEA (Load Effective Address) in Assembly Language-7 (printing values

```
dosseg ;dos segment
.model small
.stack 100h
.data
Var1 db '1'
Var2 db ?
Var3 db '123$'
.code
Main proc
Mov ax,@data
Mov ds,ax
Mov dl,var1
Mov ah,2
INT 21h
Mov var2,bl
```

Mov dl,var2

To access **VAR3**, if we do like this, then it will get the only first character of the string but not the complete string. So this is not a proper way.

Mov dx, offset var3

Offset will give us the address of the string so we can get from 1st to last.

Offset will hold the beginning address of the variable as 16 bits.

Lea dx,var3

Load Effective Address (lea) is another method of accessing string variable if you don't want to use **OFFSET**

Load Effective Address
It is an indirect instructions used as pointer in which first variables points the address of second variable.

```
Mov ah,9
INT 21h
Main endp
End Main
```

Assembly program to print two strings on two different lines, Linefeed, Carriage return. (write a code using DosBox Edit) and save as **abc.asm**

```
dosseg ;dos segment
.model small
.stack 100h
.data
Mesg1 db 'hello$'
Mesg2 db 'world$'
.code
Main proc
Mov ax,@data
Mov ds,ax
Mov dx, offset mesg1
Mov ah,9
INT 21h
Mov dx,10
Mov ah,2
INT21h
Mov dx,13
Mov ah,2
INT 21h
Mov dx, offset mesg2
Mov ah,9
INT 21h
mov ah,4ch
INT   21h
Main endp
End Main
```

Accessing first string and printing with service routine 9

ASCII code for printing next line

ASCII code for printing carriage return

Accessing second string and printing with service routine 9

DosBox Commands

- Edit Filename.asm (to create new file if not exists/open existing file)
- MASM Filename.asm; (to convert into object file using MASM assembler)
- LINK Filename.obj; (to convert object file into execution file using linker)
- To execute the exe file you just created,
  - Filename.exe (it will execute)

- NOTE: (Semicolon is mandatory while converting via assembler and linker only)