Computer Organization & Assembly Language

Lab-2

Addressing Modes, Data Transfer Instructions, Service Routine, ASCII Code and Interrupts.

- **Addressing Modes:**
  - Ways/models to access data

$$\underbrace{2 \overset{\substack{\text{Opcode}\\ \downarrow}}{+} 2}_{\text{Operands}}$$

- **Operational Code (Opcode Register1, Register2)**

Add Dl,Al

When both operators are registers, the statement will be called **<u>Registers Addressing.</u>**

- **Operational Code (Opcode Register, Value / Opcode Value, Register)**

Add Dl,2 / Add 2,Dl

When one operator is register and second is value, the statement will be called **<u>Immediate Addressing.</u>**

Operational Code (Opcode Register, [Address])

Add Dl,[Address]

When access static data directly, the statement will be called **<u>Memory Addresssing.</u>**

Addressing Modes, Data Transfer Instructions, Service Routine, ASCII Code and Interrupts.

- **Data Transfer Instructions:**

To move instruction from one register/or memory address we use **MOV**

Mov Dl, 2

- **Service Routine:**

To print or input from screen we use some service routine such as in c# we use WriteLine or ReadLine

Mov Ah,2

Important Service Routines

1=Input a character with echo
2=Print/Output a single character 'a'
8=Input a character without echo
9=Print collection of characters 'abcd' ------- String
4ch=Exit

Addressing Modes, Data Transfer Instructions, Service Routine, ASCII Code and Interrupts.

- **Interrupts:**

Stop the current program and allow microprocessor to access hardware to take input or give output

INT 21H – Interrupt for text handling

INT 20H – Interrupt for graphics/video handling

Example 1: Output

```
Mov ah,2
INT 21H
```

Example 2: Input

```
Mov ah,1
INT 21H
```

Addressing Modes, Data Transfer Instructions, Service Routine, ASCII Code and Interrupts.

- **ASCII:**

American Standard Code for Information Interchange, is a character encoding scheme.

A – Z (65 – 90)

a-z (91-122)

0-9 (48-57)

Next Line = 10 (and print) for next line

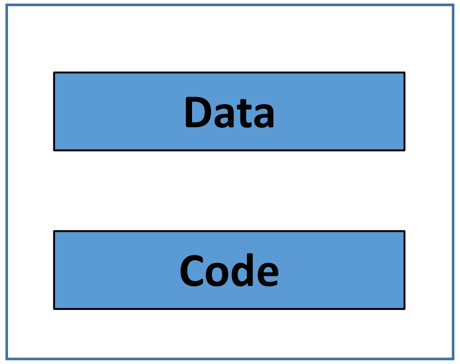
Carriage Return = 13 such as Enter Key

Model Directives

- Defines the total amount of memory program needed

.model

Tiny	Data + Code <=64KB
Small	Data <=64KB, Code <=64KB
Medium	Data <=64KB, Code = Any size
Compact	Data = Any size, Code<=64KB
Large	Data = Any size, Code=Any size
Huge	Data = Any size, Code=Any size



- Being beginner we always use following:

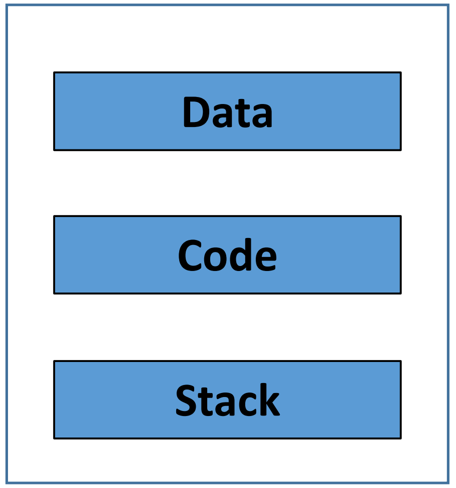
.model small (at the start of every program)

Stack Segment Directives

- Defines the storage of stack in RAM

.model small

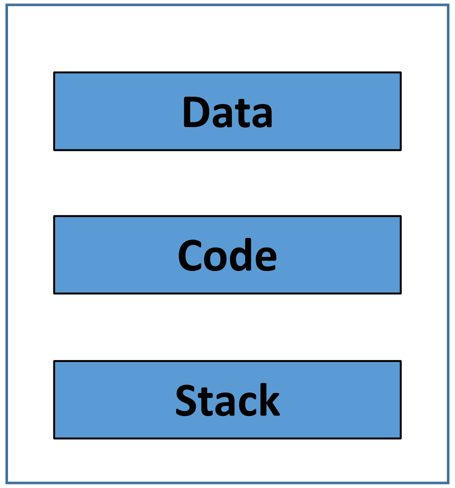
.stack 100h (its 100 hexadecimal number) (mandatory when you specifies the stack storage in RAM)



Data Segment Directives

- Variables are defined in RAM

```
.model small
.stack 100h
.data    ;variables are defined here
```

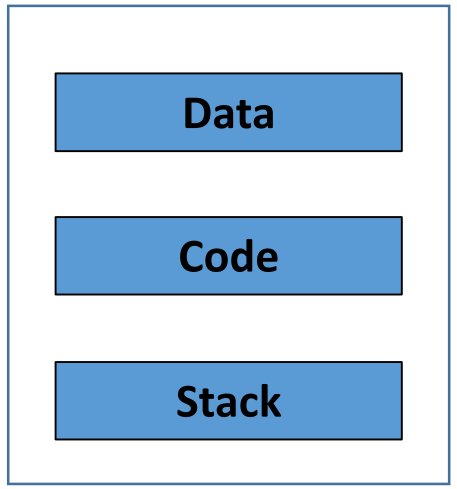


Code Segment Directives

```
.model small
.stack 100h
.data
.code


   ;code or executable instructions goes here


End Main
```



Assembly Program

.model small

.stack 100h

.data

.code

Main proc

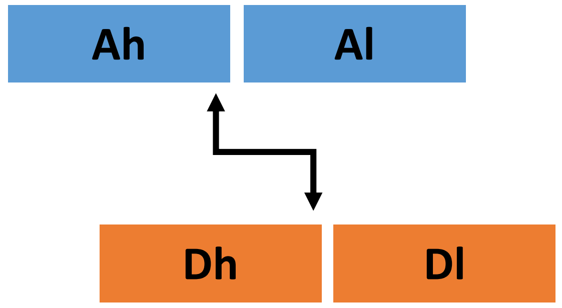
(it’s a main procedure of the code, you can define as many as procedure in the code ended with “*Main endp*”

Main endp

End Main

Program to print single character on screen

- When we want to print a character on a screen we need to Accumulator such as:



So we move accumulator into Data register as shown in the above mentioned figure. We may write

Mov dl,’A’

Syntax Rules: (non acceptable)

So we move accumulator into Data register as shown in the above mentioned figure. We may write

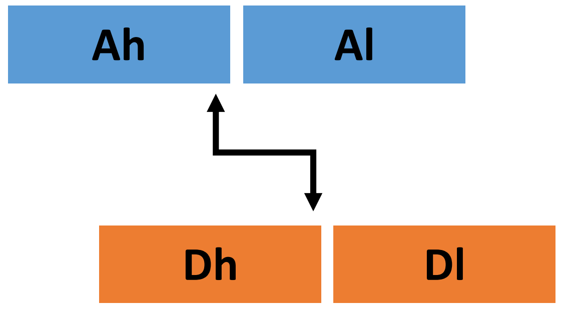
Mov ‘B’ , ’A’ (not allowed) – because you have to use one register at-least

Similarly

Mov 2,3 (not allowed)

Moreover,

Mov dl,AX (not allowed) – because both registers are type mismatched – dl is 8 bits and AX is 16 bits.



Syntax Rules: (acceptable)

Mov dl, ‘A’

Mov dl,2

Mov dx,Ax

Mov dh,al

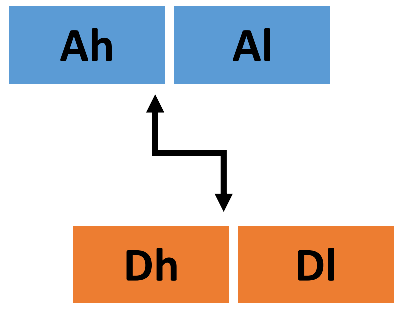
Syntax Rules:

-> Space after OpCode (Mov dx,Ax).

-> One operand must be general purpose register.

-> Operands must be same sized.

-> Comma, between operands.



Assembly Program (write a code using DosBox Edit) and save as **abc.asm**

```
dosseg ;dos segment
.model small
.stack 100h
.data
.code
Main proc
 Mov dl,'A'
Mov ah,2
INT 21h
Mov ah,4ch
INT 21h
Main endp
End Main
```

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)