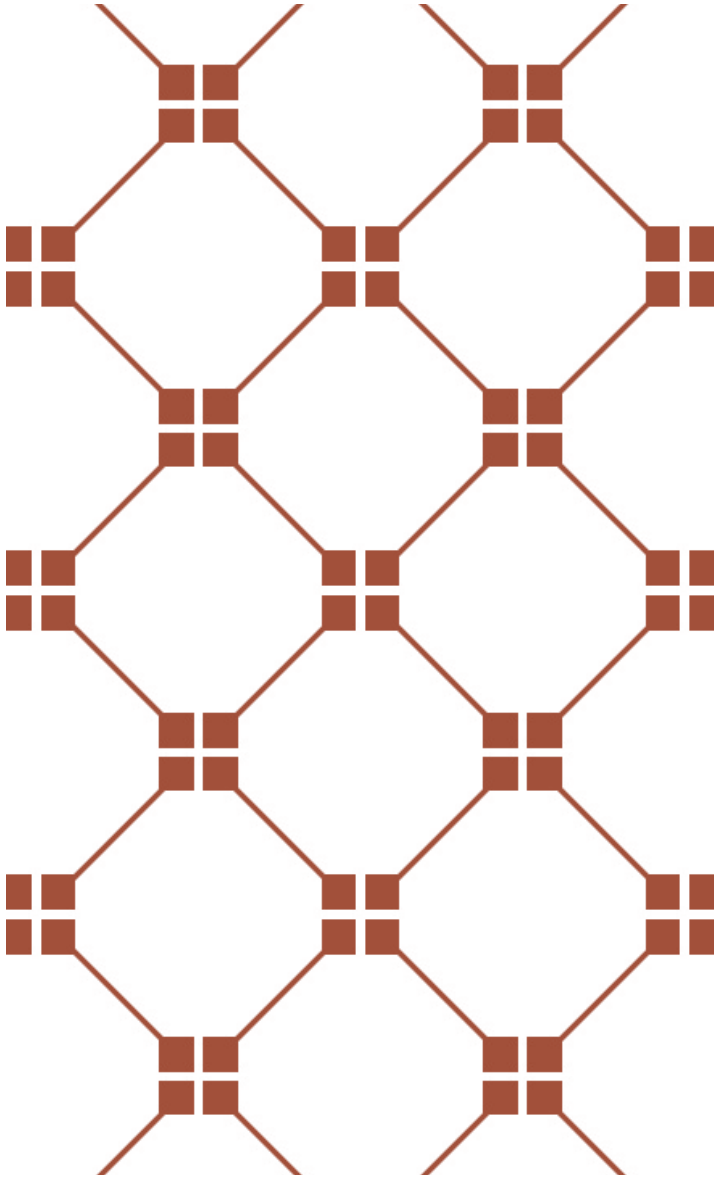


BASIC STRUCTURE AND INSTALLATION

Assembly directives are instructions that direct the assembler to do something. They are not converted to machine code. There are different directives in assembly language, some of them are;

- .model
- .stack
- .data
- .code
- Db, dw, dt

```
;Title of the program
Dosseg
.model small
.stack 100h
.data
        ; variables are defined here
.code
Main proc     ; procedure start
        ; executable instructions

Main endp     ; procedure end
End main
```

Program Structure

To understand program structure, Instructions.

First we write

① .model ⟶ .model directive

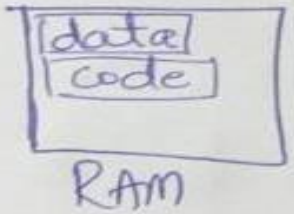
Work is to define our assembly code Space in RAM.

(Any file Doc, text it has HD space reseved in MB, GB). when we reen it divide it in part and go to RAM & RAM Acces it from CPU and run it).

First we reserve part of our program in RAM, Our code access by CPU & understand what is our program and behave.

What space we need in RAM.

Every Programming language has used 2 segments Data segments & Code segment.
One is data define and other is execute it.



We need 2 spaces in Assembly.

In data, Variables are define
Code, instructions are written.

What space needed ?

Tiny code + Data <= 64kB. (for both <= 64kB)

Small code <= 64 kB , Data <= 64kB

~~Code Data~~

(For seperate code & data)

Medium	code = Anysize, Data <= 64kB.
Compact	code <= 64kB, Data = Anysize.
Large	code = Anysize, Data = Anysize.
Huge	Code = Anysize, Data = Anysize

We used small, don't reseve large space of RAM

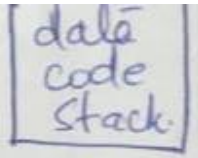
- model small
- model directive → "Specifies the total amount of memory the program would take."

(2). We used stack memory, stack management, data first put retreive last.
we have to define it.
If we take stack in RAM have to define it.

• stack 100h → size.

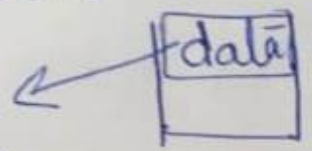
Hexadecimal defines the size,
Only define if we used stack.

." Stack Segment Directive Specifies the storage for Stack."



③ .data Data Segment Directive

Any values we define here, go to data (RAM).
Variables are defined here, it has segment where all data is saved.



Variables defined here.
thus part of RAM is reserve

(4) .code Code Segment Directive

→ Between .data & .code we defined Variables.

→ Instruction which we want to run, code is written, it is reserved in RAM.

(5) End Main :- End of program.

~~Where code we write executable instructions.~~

Sample

```
.model small
.stack 100h
.data
; variables defined here.
.code
; here we write our code.
End Main
```

In .code we write executable instructions.
After it we write our program.

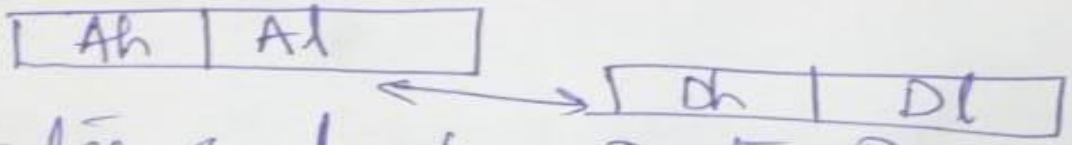
Main proc ⟶ If our program for only one work we write procedure & write our code.
We use any name rather then Proc.
We made any proc under Main proc.
We work on one procedure.

Main endp
End Main.

Now we write code to print a character.

We used Accu. Reg for I/O.

→ Accu. go to data Reg, take value from there and print.



Register interaction.

→ First character send to Data Register.

Mov dl, 'A' ✓

This is immediate addressing mode.

- Mov dl, 2 ✓
- Mov dl, 'A' ✓
- Mov dx, Ax ✓
- Mov dh, al ✓

Practise

One Reg is Must

Mov 'B', 'A'	x
Mov 2, 3	x
Mov dl, Ax	x

(dl: 16bit, Ax: 8bit)

Type Match is must.

Syntax Rule

1) Space is necesary after Opcode (space after Mov).
2) One operand must be General purpose Register.
3) Operands must be of same size.
4) Comma b/w Operands.

Now to print A.

```
Mov ah, 2
INT 21h.
```

→ 2 is service routine for single character print.

Now we are in register, we have to return from register and use the service routine.

```
Mov ah, 4CH
INT 21H.
```

These are part of hardware Interaction:, these two are part of program.

Mov ah, 4CH
INT 21h.

→ If you forget the arrangement of lines then write dosseg before starting of program

→ dosseg ⟶ Dos Segment ⟶ Manages the arrangement (Automatically arranges) of segments in a program.

⟶ ; comment (Used for comment).

DosBox, MASM, LINK.

Emulator → Environment provide facility to run Program / software of Assembly program.

Manufactured by July. 22, 2002 peter Veenstra.

Advantages (Why DosBox).

Why not MASM Emulator by Ms itself.
" " EMU8066?
" " Visual Studio with kip Irvine Lib.

1) free software, no need to crack / purchase.
2) Light Simple easy to use.
3) Run on every environment Linux, Android, Windows, Mac.
4) Can learn Debugging (Identify & correct mistakes).

5) Can quit to Syntax.

- In DosBox we made Assembly Lang. Prog. to Convert into exe file we need MASM & LINK. (Convert Assembly code to executable code.)
- To change Assembly code to Object file we need Assemblee MASM.
- DosBox ⟶ MASM ⟶ Objectfile ⟶ Linkee ⟶ Machine code.
- Filename.asm —MASM→ Filename.obj —Linkee→ Filename.exe

Commands to Create file and Run of DosBox.

To create new file, or view a file.

Edit Filename.asm. ⟶ If no file exict it create new, or if file exist it open it for edit.

To Run a file

We need assembler.

MASM Filename.asm; Convert file into obj

To change into obj

LINK Filename.obj; to change into exe.

Filename.exe

; is in MASM and LINK, backend processing is not required.

1) Download it.
2) Extract.
3) Install DosBox
4) Installed.
5) Coby MP folder (MASM and LINK), go to main C drive and paste it.
6). Open DosBox.

Z

It is another enviroment, built in drive
We have to go to MP folder.
(create virtual connection.

mount c c:\MP

Enter.

Give path, MP folder mounts in DosBox with name of C, like C drive MP folder connected Path is linked now.

Program to print a single character on screen

```
.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
```

2. PROGRAM TO PRINT NAME WITH CHARACTER

```
.model small
.stack 100h
.data
.code
main proc
mov dl, 'A'
mov ah,2
INT 21h
mov dl, 'L'
mov ah,2
INT 21h

mov dl, 'I'
mov ah,2
INT 21h

mov ah, 4ch
INT 21h
main endp

end main
```

3. PROGRAM TO TAKE INPUT CHARACTER AND PRINT ON SCREEN

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

main proc

mov ah,1
INT 21h

mov dl, al
mov ah,2
INT 21h

mov ah,4ch
INT 21h

main endp

end main
```

1). Program to print a name with characters.
Every letter is single character.

→ Acc. Reg is used b/c Print | Input.
→ 'A' moved to dl (data Reg), Acc Print it.
→ Same for every character.
→ mov ah, 4ch] to exit from register.
int 21h

2) Program to ~~print~~ take input characters and print on screen.

For taking Input Acc. Reg is used (mov ah, 1).
If input is in accumulator it is place in(al).
Send it to data register for print, b/c for print the output the value present in data register, Acc Reg. Print it.