Multiplication In Assembly.

5 x 9

multiplicand → Multiplier.

1) In assembly if Multiplier is of 8 bits, multiplicand is also of 8 bits,

2) Both should be same.

Multiplicand – Multiplier
AL – BL, CL, DL
AX – BX, CX, DX.

3). Multiplication in Assembly.

Keyword ← mul multiplier → Value of multiplier. (BL, CL, DL, BX, CX, DX).

4). Example of Multiplication for 8 bits.

multiply ↓
Mov al, 5 → 5 moves to al
Mov bl, 9 → 9 " bl.
mul bl

5). In case of 8 bits, put 5 to al & 9 to bl..

al (5) x bl (9) result go to AX

put Ax to Dx & print the result.

6) In case of 16 bits, result not handled by single register, we used two registers DX & AX,

AX × BX = DX AX

5 × 9 = 45

for result, two registers are used,
Converted into Binary as, 00000000 00010101

7). Result of 8 bits, result goes to ax then move to dx.

```
mov dx, ax   ──→ print a number on screen
add dx, 48   ──→ Ascii code convert.
mov ah, 2    ──→ Print single digit.
int 21h
```

8) If result not single digit, 5x9=45.
function 2 print one digit, (not two digits), technique used

AX = AH : AL ──→ put 4 in AH & 5 in AL.
4 5

9). How this performed in Assembly,
used directive called AAM (ASCII Adjust after Multiplication)
Like,

```
mov al, 5
mov bl, 9
mul bl
AAM
```

AAM → It perform the functionality break the AX in two Parts (AH & AL) and adjust.

10). After writing AAM, 4 goes to AH & 5 to AL. (3)

11). We have to store it in another register to save the value.

```
        mov ch, ah
        mov cl, al
mov dl, ch
add dl, 48
mov ah, 2
int 21h
mov dl, cl
add dl, 48
mov ah, 2
int 21h
```

(4).

Code to Multiply two number & print the product

```
.code
main proc
mov al, 5
mov bl, 9
mul bl
AAM
mov ch, ah
mov cl, al
mov dl, ch
add dl, 48
mov ah, 2
int 21h
mov dl, cl
add dl, 48
mov ah, 2
int 21h
mov ah, 4ch
int 21h
main endp
end main
```

Graphics :-

Two concepts → Graph
→ Graphics.

·) Screen is made up of pixels.

· Two points connect/ multiple points connect together, fill them, image is created, process is called graphics.

Graph :- Connect points / Relationship b/w points or objects.

Graphics :- to draw graph using computer is called Computer graphics i.e. graphics.

1). In assembly graphics, deep understanding
2). Draw shapes, write text.
3) Games codes.

Graphics In Assembly.

1). Interrupt used for graphics,

int 10h

2). We know that, mov ah, 2

AH set function / Service Routine.

Same as different functions in Graphics mode, give it in AH & called int 10H after it.

3). Graphics API functions / Service Routines.

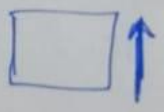
00h : Set Video mode
01h : Set cursor lines
02h : Set cursor position
03h : Get cursor position & size
06h : Scroll window up
07h : Scroll window down.
08h : Read character & attribute
09h : Write character & attribute
0Ah : Write character
10h (AL=03h) : Toggle blinking / intensity bit
0Fh : Get video mode
13h : Write string in teletype mode.

4) Program to draw box,

function used for BOX / Square / Rectangle.

```
mov ah, 06h
int 10h
```
function to draw Box.
↳ Scroll up on DOSBox screen.

5). For Set height



. How many lines up to ↳ fill.
. With AL.

AL : Number of lines to be scrolled, lines to be filled.

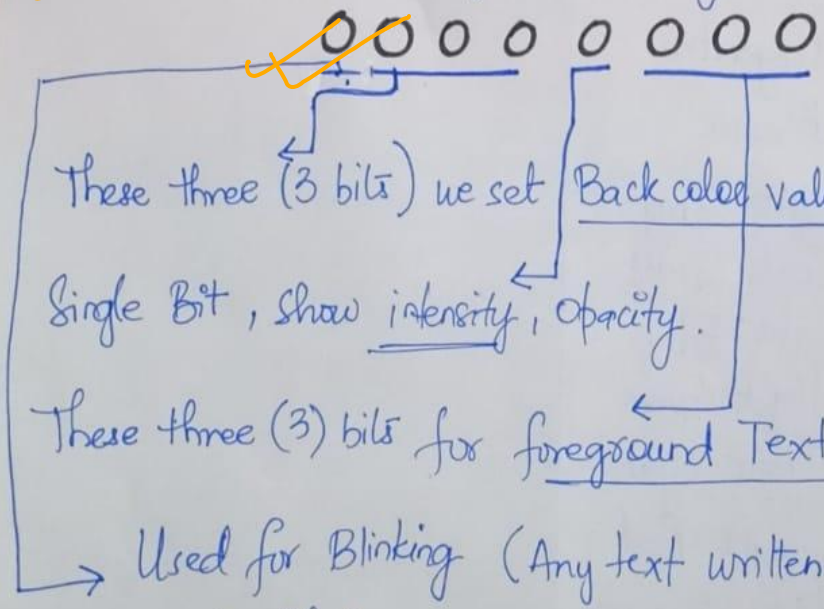
e.g. mov al, 5 → 5# of lines scroll up & filled.

6). If write mov al, 00h, full screen filled

7). For Set Color : Used bh.

Mov bh, Color Value.

• Color value should given in binary, because of 8 bits.



Remember ① Because box not blink, If not writing text (only text will blink), If not writing text the value set 0 or 1, no matter.

② Set 000 for foreground text color if only draw Box

Back Color Value in Binary.

Binary	Color
0000	Black
0001	Blue
0010	Green
0011	Cyan
0100	Red
0101	Magenta
0110	Brown
0111	Light Gray
1000	Dark Gray
1001	light Blue
1010	Light Green
1011	Light Cyan
1100	Light Red
1101	Light Magenta
1110	Yellow
1111	White

Give Colors as, Mov bh, 00010000b

for Blue color intensity & text set 0.

8). Starting Position of Box :- Need to give points.

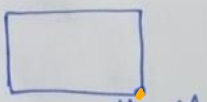
- Set starting quadrants of Box.
- Set points in ch, cl.

CH : Top Row of Window.
CL : Left Most Column of Window

If
```
Mov ch, 0
Mov cl, 0
```
} set first Corner of Screen.

9). Bottom :-

DH : Bottom row of Window
DL : Right most Column of Window



10). Height set by al,
Width set by dh, dl

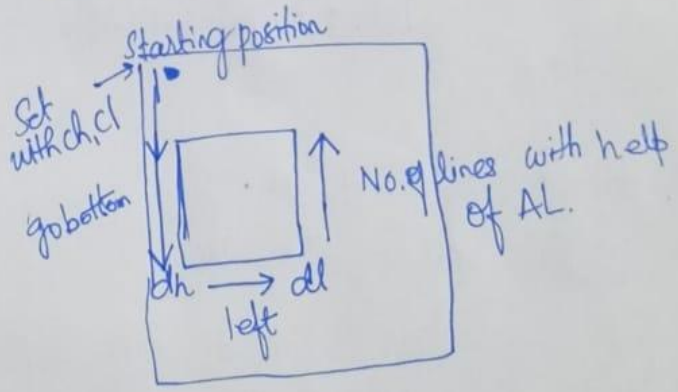
If given
```
Mov dh, 24
Mov dl, 24.
```

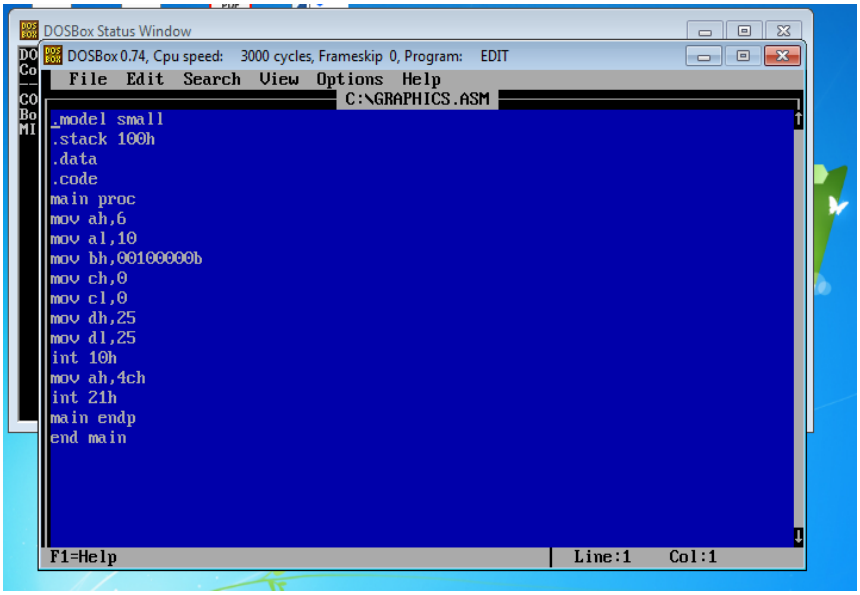
11). If write Mov dx, 184fh

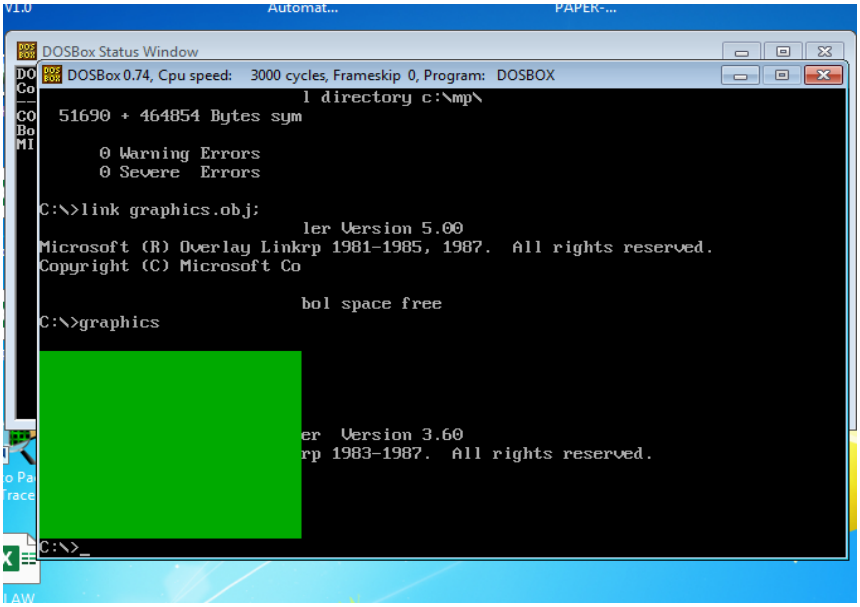
full screen quadrants fill.

```
.model small

.code
main proc
mov ah, 6      → Set function first
mov al, 10     → No. of lines set.
mov bh, 00010000b → Set Color.
mov ch, 0
mov cl, 0      } starting position.
mov dh, 25
mov dl, 25     } set Bottom & left.
int 10h        → Call Interrupt.
mov ah, 4ch
int 21h
main endp
end main
```




DOSBox Status Window
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: EDIT
File Edit Search View Options Help
C:\GRAPHICS.ASM
.model small
.stack 100h
.data
.code
main proc
mov ah,6
mov al,10
mov bh,00100000b
mov ch,0
mov cl,0
mov dh,25
mov dl,25
int 10h
mov ah,4ch
int 21h
main endp
end main
F1=Help
Line:1 Col:1


DOSBox Status Window
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
l directory c:\mp\
51690 + 464854 Bytes sym
0 Warning Errors
0 Severe Errors
C:\>link graphics.obj;
ler Version 5.00
Microsoft (R) Overlay Linkrp 1981-1985, 1987. All rights reserved.
Copyright (C) Microsoft Co
bol space free
C:\>graphics
er Version 3.60
rp 1983-1987. All rights reserved.
C:\>_