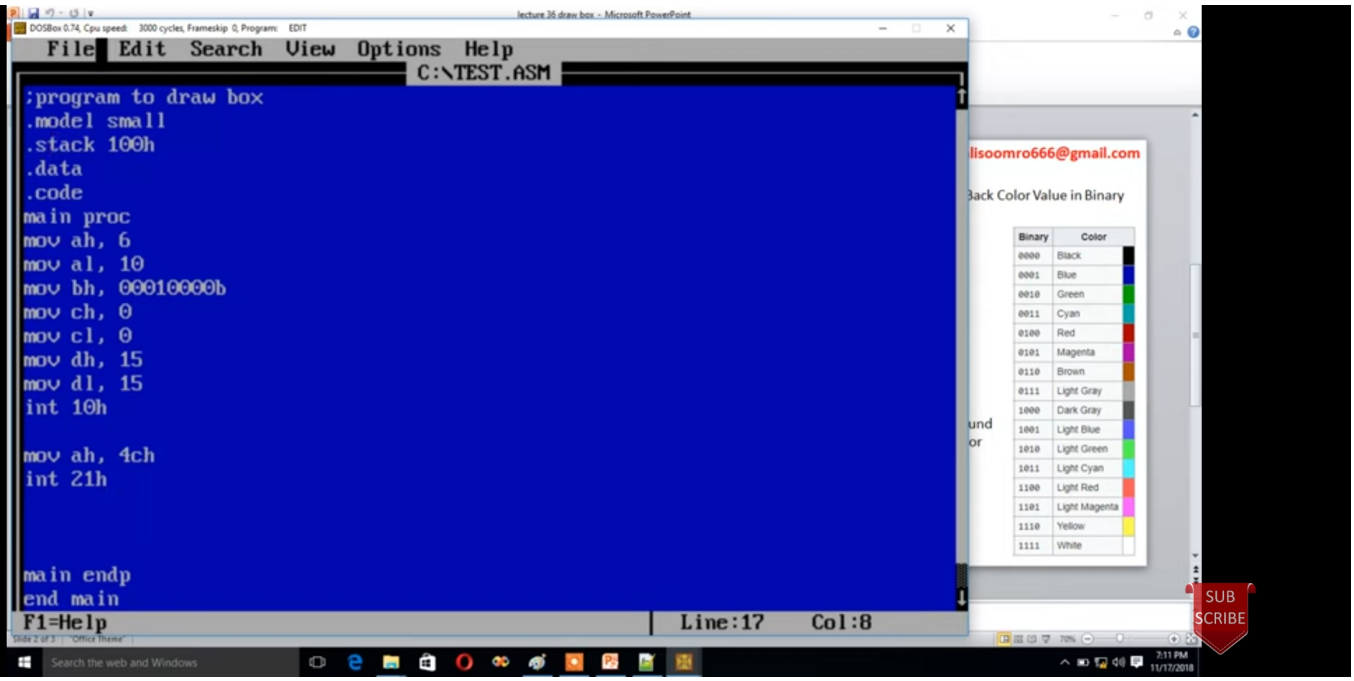

lecture 36 draw box - Microsoft PowerPoint
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: EDIT
File Edit Search View Options Help
C:\TEST.ASM
;program to draw box
.model small
.stack 100h
.data
.code
main proc
mov ah, 6
mov al, 10
mov bh, 00010000b
mov ch, 0
mov cl, 0
mov dh, 15
mov dl, 15
int 10h
mov ah, 4ch
int 21h
main endp
end main
F1=Help
Line:17 Col:8
lisoomro666@gmail.com
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
SUB SCRIBE
Search the web and Windows
7:11 PM
11/17/2018

Lecture 36: Program to Draw Box/Rectangle/Square in Graphics int 10h

alisoomro666@gmail.com

mov ah, 06h
Int 10h

mov ah, 2 Function to Print Char
Int 21h Text Mode

Back Color Value in Binary

Ah, 06 Function/Routine to request to Scroll lines up
Al number of lines to be scrolled, lines to be filled
BH: color attribute
CH: Top row of window
CL: Left most column of window
DH: Bottom row of window
DL: right most column of window

```
Mov ah, 06h
Mov al, 10
Mov bh, 00010000h
Mov ch, 0
Mov cl,  0
Mov dh, 15
Mov dl,  15
Int 10h
```

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

Mov ch, 0
Mov cl, 0
ch, cl

Mov cx, 0000

Mov dx, 184fh Full End Coordinates

Mov al, 00h Full Screen

dh, dl

Width ?

height? Mov al, 10

Color? Mov bh, 00010000b

Foreground Text Color

Intensity

0 0 0 0 0 0 0 0

Set Blinking

Color

Mov dh, 24
Mov dl, 24

1: Blinking
0: Not Blinking

SUB SCRIBE

Lecture 35: Graphics, Int 10, AH Functions / Routines

alisoomro666@gmail.com

Graph?

Connect points / Relationship between points/objects

What is Graphics?

to draw graph using computer is called Computer Graphics (I.e. Graphics)

Why Graphics in Assembly?

- ✓ Deep understanding of Computer Graphics
- ✓ Program to make shapes and text
- ✓ Games

How to perform Graphics in Assembly?

Int 10H

Interrupt for Graphics

Graphics AH Functions/Service Routines

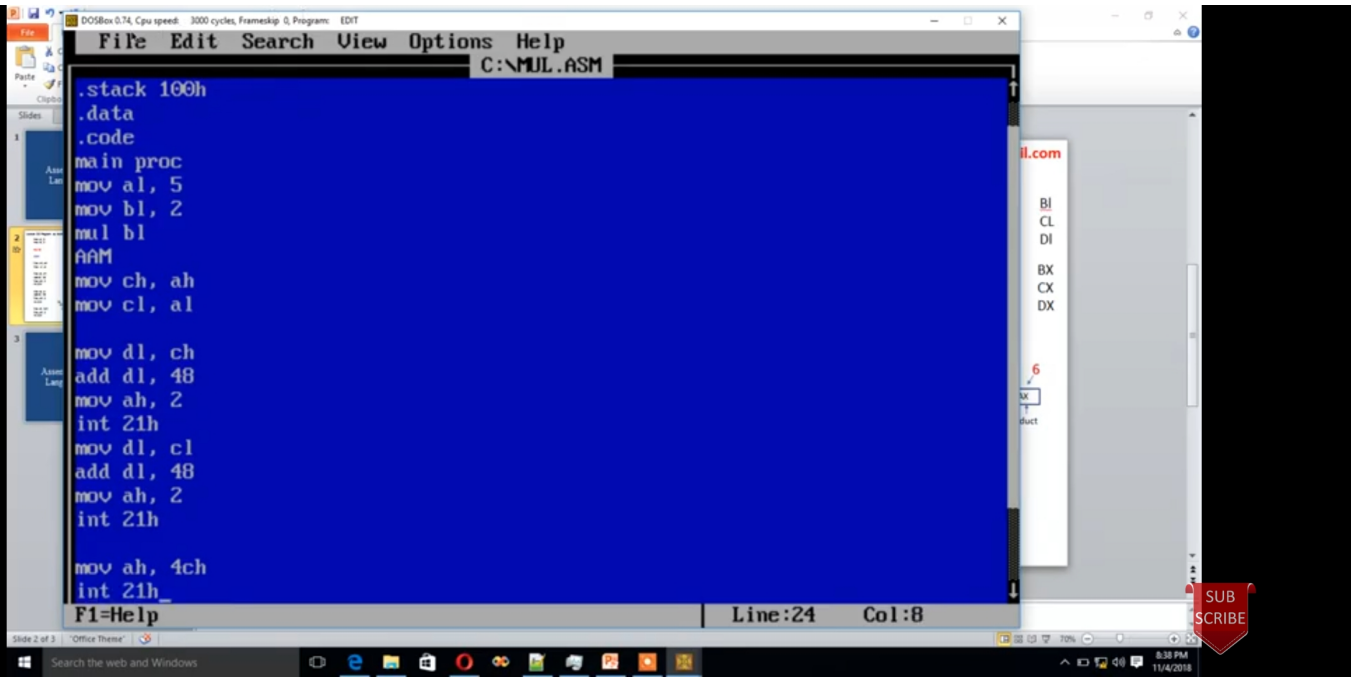
- 00h: Set video mode
- 01h: Set cursor lines
- 02h: Set cursor position
- 03h: Get cursor position and size
- 06h: Scroll window up
- 07h: Scroll window down
- 08h: Read character and attribute
- 09h: Write character and attribute
- 0Ah: Write character
- 10h (AL = 03h): Toggle blinking/intensity bit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

Mov ah,
Int 21h

Mov ah, 2
Int 21h

Ah sets functions/Service routine

Interrupt for Text


DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: EDIT
File Edit Search View Options Help
C:\MUL.ASM
.stack 100h
.data
.code
main proc
mov al, 5
mov bl, 2
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
F1=Help
Line:24 Col:8
SUB SCRIBE

Lecture 34: Program to multiply two numbers and print the product

alisoomro666@gmail.com

```
mov al, 3
mov bl, 2

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 dl, 4ch
Mov ah, 2
Int 21h
```

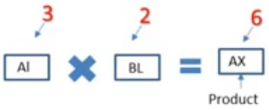
```
Mov al, 3
Mov bl, 2

Mul bl

mov dx, ax
add dx, 48
mov ah, 2
Int 21h
```

5 x 2 = 10

3 * 2 → multiplier (BL, CL, DL)

multiplicand: AL

AX (BX, CX, DX)



AAM

ASCII **A**djust after **M**ultiplication

10 → AX = AH : AL ← 1, 0

Lecture 31: Multiplication, Multiplication for Unsigned (Positive Numbers) with Example alisoomro666@gmail.com

```
mov al, 3
mov bl, 2

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 dl, 4ch
Mov ah, 2
Int 21h
```

```
Mov al, 3
Mov bl, 2

Mul bl

mov dx, ax
add dx, 48
mov ah, 2
Int 21h
```

5 x 2 = 10

Why do we learn Multiplication in Assembly?

- ✓ To know how computer perform multiplication

3 * 2 → multiplier

multiplicand

AL

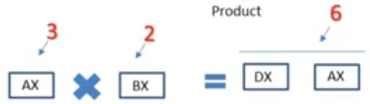
AX

BI
CL
DI

BX
CX
DX

How to perform Multiplication in assembly?

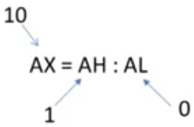
mul multiplier

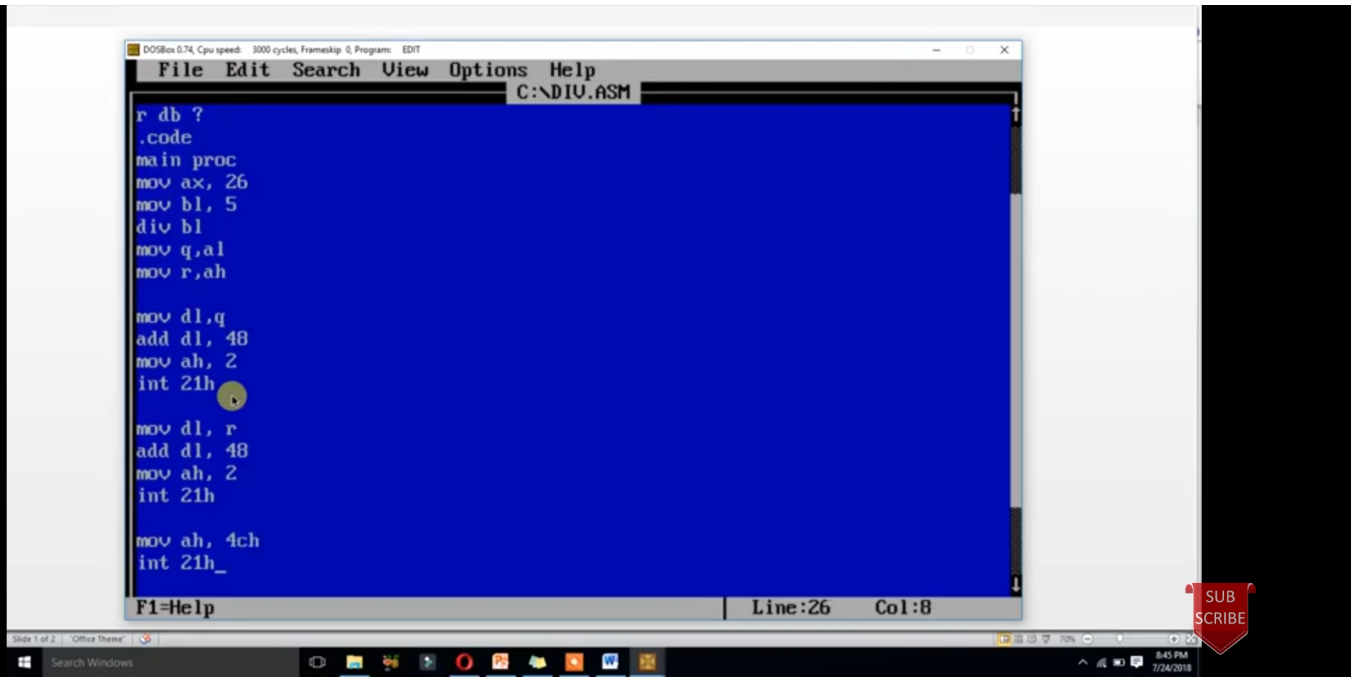


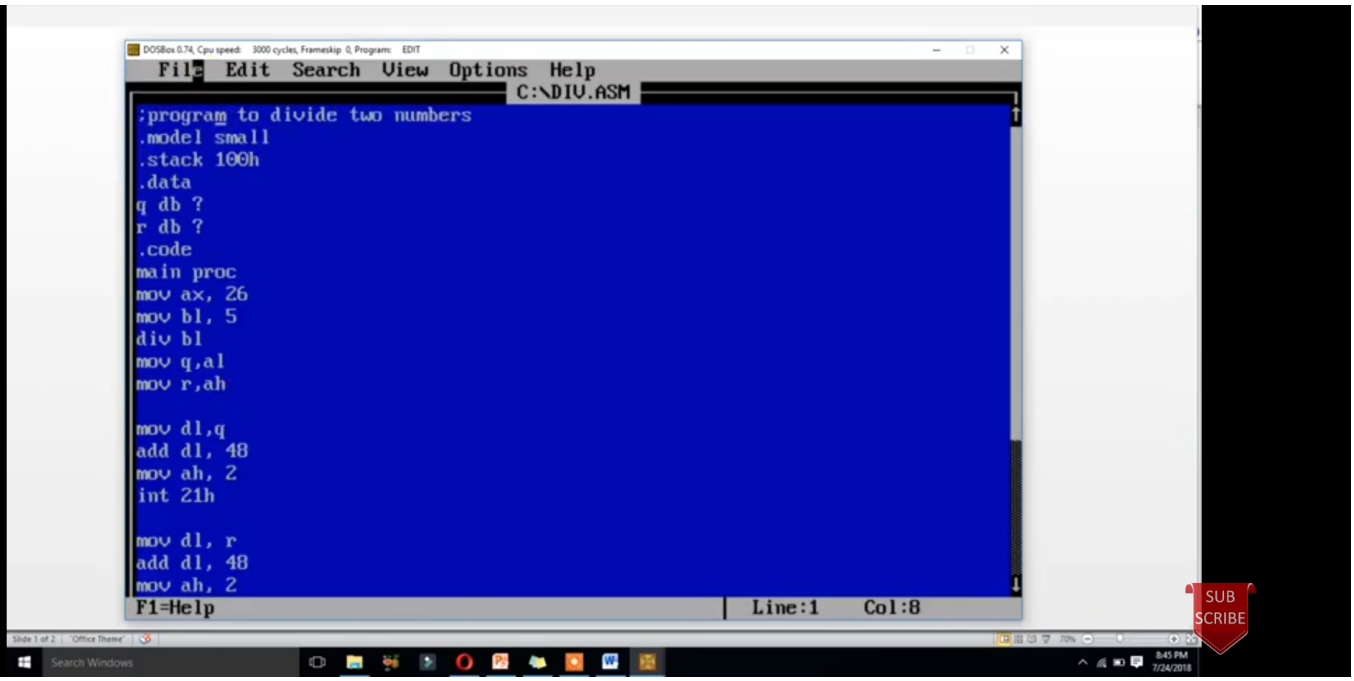
0000000000000000 0000000000000110

AAM

ASCII **A**djust after **M**ultiplication

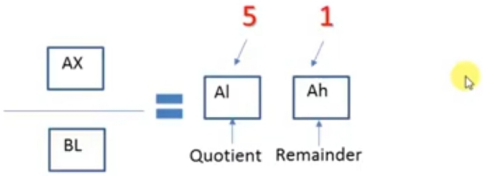



DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: EDIT
File Edit Search View Options Help
C:\DIV.ASM
r db ?
.code
main proc
mov ax, 26
mov bl, 5
div bl
mov q,al
mov r,ah
mov dl,q
add dl, 48
mov ah, 2
int 21h
mov dl, r
add dl, 48
mov ah, 2
int 21h
mov ah, 4ch
int 21h_
F1=Help
Line:26 Col:8
SUB SCRIBE


DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: EDIT
File Edit Search View Options Help
C:\DIV.ASM
;program to divide two numbers
.model small
.stack 100h
.data
q db ?
r db ?
.code
main proc
mov ax, 26
mov bl, 5
div bl
mov q,al
mov r,ah
mov dl,q
add dl, 48
mov ah, 2
int 21h
mov dl, r
add dl, 48
mov ah, 2
F1=Help
Line:1 Col:8
SUB
SCRIBE
Search Windows
8:45 PM
7/24/2018

Lecture 32: Program to divide two numbers and print quotient and remainder

alisoomro666@gmail.com



Lecture 31: Division for Unsigned (Positive Numbers) with Example

alisoomro666@gmail.com

```
.data
Quo dw ?
Rem dw ?

.code
Main proc
Mov ax,@data
Mov ds, ax

Mov Ax, 26
Mov DX, 0

Mov BX, 5
Div BX

Mov Quo, AX
Mov Rem, DX

Mov DX, Quo
Mov ah, 2
Int 21h

Mov DX, Rem
Mov ah, 2
Int 21h
```

26 / 5 → Divisor: Bl, CL, Dl, BX, CX, DX

Dividend: Ax, DX : AX

```
Mov Ax, 26
Mov bl, 5
Div bl

mov cl, al
Mov ch, ah

Mov dl, cl
Mov ah, 2
Int 21h

Mov dl, ch
Mov ah, 2
Int 21h
```

If divisor is 8 bit e.g. bl, cl, dl

Why do we learn Division in Assembly?

- ✓ Convert decimal to binary

Other benefits of learning this?

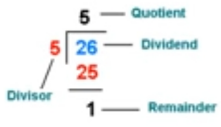
- Print decimal of many bits

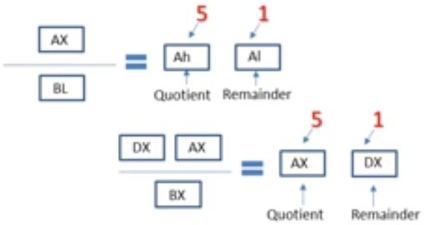
```
Mov bl, 15
Add bl, 10
```

- Given number is odd or even?
- Division program and many more programs..

How to perform Division in assembly?

Div divisor





SUB SCRIBE

Lecture 30 : Signed and Unsigned Numbers, convert unsigned to signed numbers

alisoomro666@gmail.com

- 6 — **Negative Number**

6 — **Positive Number**

-6 * -6 = 36 -6 / -6 = 1

-6 * 6 = -36 -6 / 6 = -1

6 * 6 = 36 6 / 6 = 1

One's Compliment

If 1, it is negative
If 0, it is positive

Most Significant Bit **MSB**

Two's Compliment

How to convert unsigned to signed numbers?

1	0001h	0001
-1	FFFFh	- ffff = FFFE + f = ffff
1	0001b	
-1	1111b	

0001

LSB
Least Significant Bit

0001
- 1111
1110
+1
1111

Signed Number

Mov bl, 2
Sub bl, 4

Bl

-2

Bl - 4

2 - 4

Mov bl, 3
add bl, -2

Bl

1

Bl + (- 2)

3 + (- 2)

Lecture 29 : Macro, example program for macro

alisoomro666@gmail.com

```
print macro p1
Mov dx, offset p1
Mov ah, 9
Int 21h
endm

.model small
.stack 100h
.data
Str1 db 'hello$'
Str2 db 'it is a test program$'
.code
Main proc
Mov ax, @data
Mov ds, ax

print str1
print str2

Mov ah, 4ch
Int 21h
Main endp
End main
```

What is macro?

It is just a block of code that can be used with input parameters anywhere in the program with name
It is a perfect function.

```
.data
Str1 db 'hello$'
Str2 db 'it is test program$'

print macro p1
Mov dx, offset p1
Mov ah, 9
Int 21h
endm

print str1
print str2

print proc
Ret
print endp
```

For Print

Why do we need it?

- Reusability with input parameters
- Reduce complexity

How to use macro?

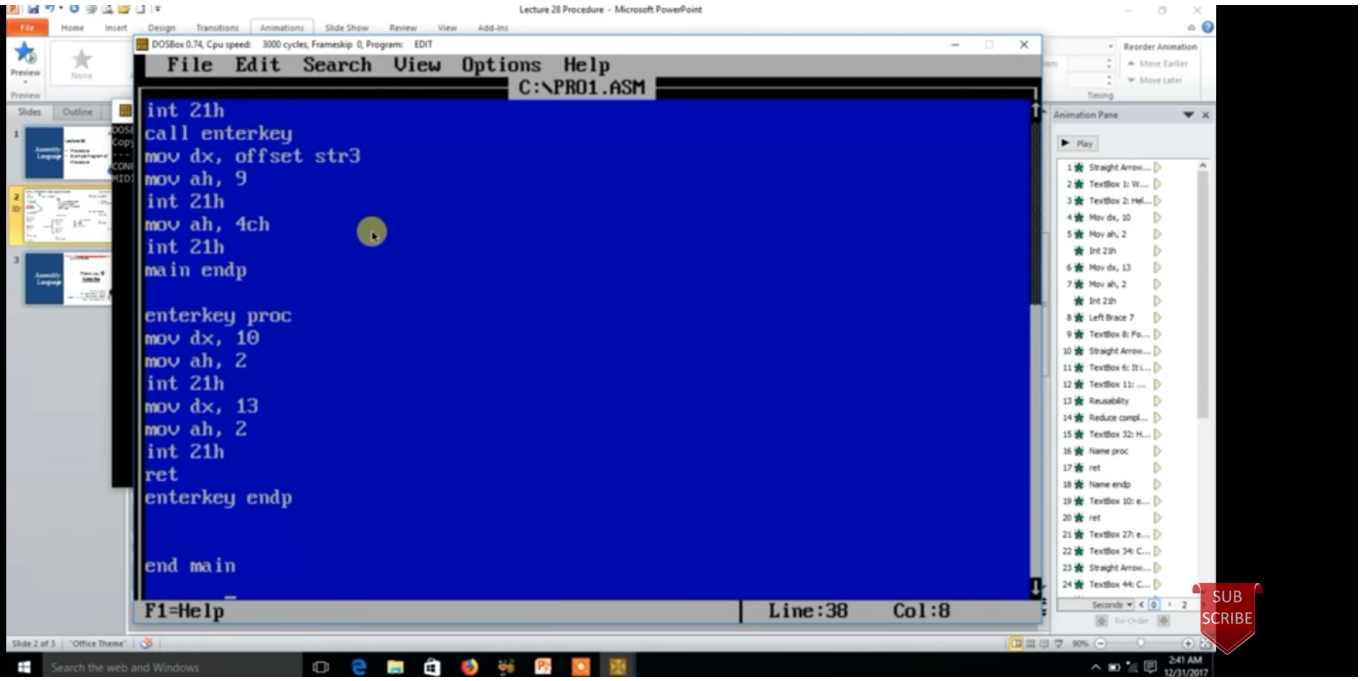
```
Name macro p1, p2, ...

endm
```

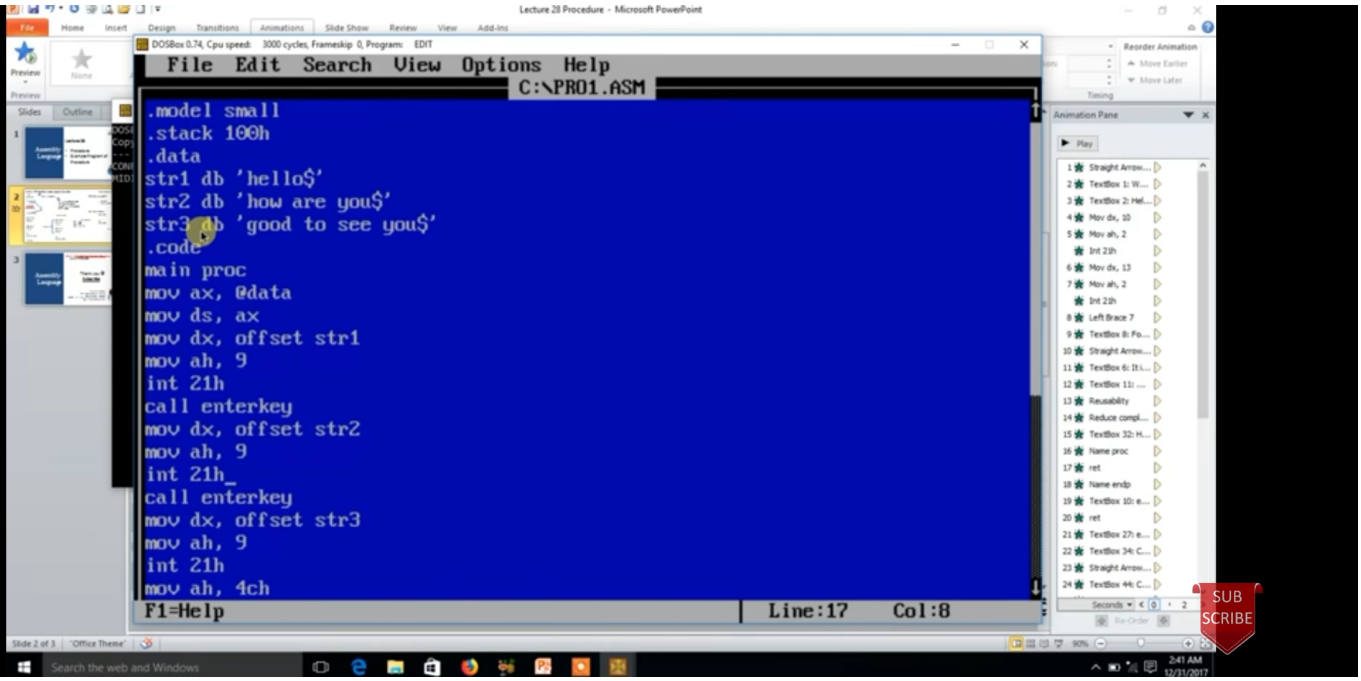
Name p1, p2, ...

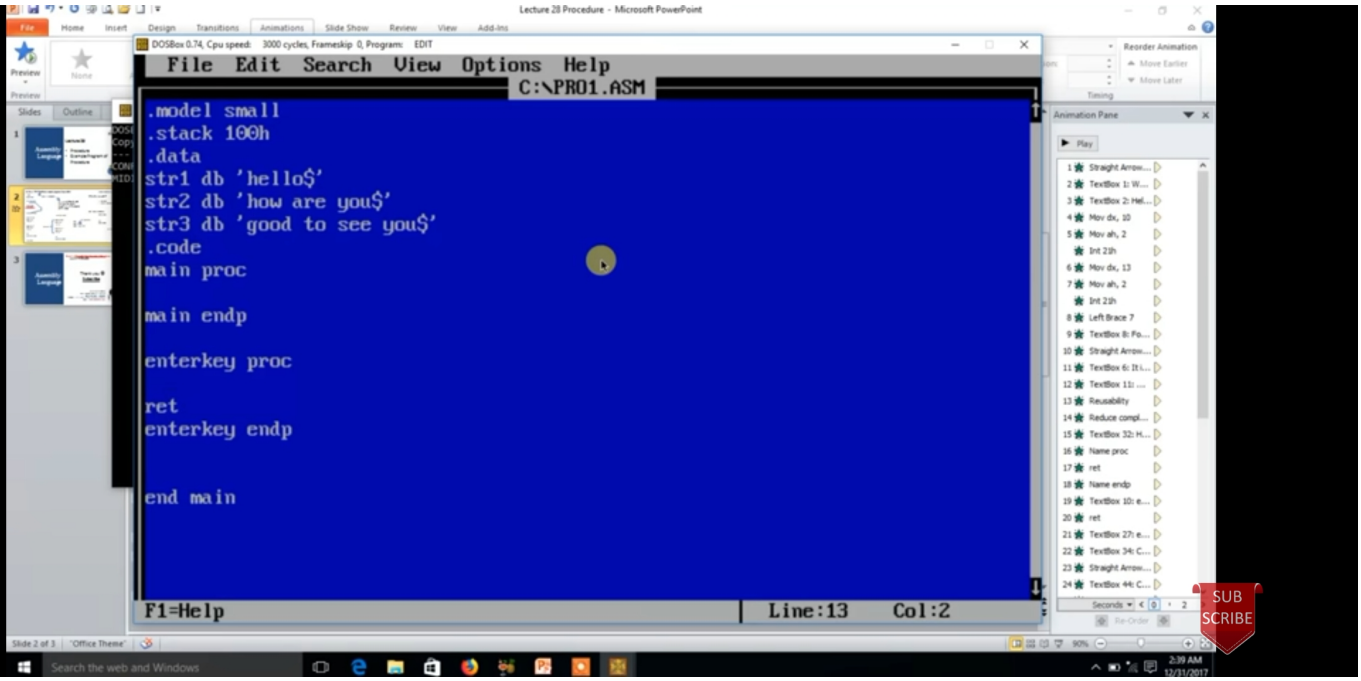
What is Difference b/w Procedure and Macro

Proc	Macro
No input parameters	Input parameters
Ret is used	No 'ret' is used
Slow, goes and run code	Fast, replace with code

SUB SCRIBE


Lecture 28 Procedure - Microsoft PowerPoint
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: EDIT
File Edit Search View Options Help
C:\PRO1.ASM
int 21h
call enterkey
mov dx, offset str3
mov ah, 9
int 21h
mov ah, 4ch
int 21h
main endp
enterkey proc
mov dx, 10
mov ah, 2
int 21h
mov dx, 13
mov ah, 2
int 21h
ret
enterkey endp
end main
F1=Help
Line:38 Col:8
SUB SCRIBE


Lecture 28 Procedure - Microsoft PowerPoint
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: EDIT
File Edit Search View Options Help
C:\PRO1.ASM
.model small
.stack 100h
.data
str1 db 'hello$'
str2 db 'how are you$'
str3 db 'good to see you$'
.code
main proc
mov ax, @data
mov ds, ax
mov dx, offset str1
mov ah, 9
int 21h
call enterkey
mov dx, offset str2
mov ah, 9
int 21h
call enterkey
mov dx, offset str3
mov ah, 9
int 21h
mov ah, 4ch
F1=Help
Line:17 Col:8
SUB SCRIBE


Lecture 28 Procedure - Microsoft PowerPoint
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: EDIT
File Edit Search View Options Help
C:\PRO1.ASM
.model small
.stack 100h
.data
str1 db 'hello$'
str2 db 'how are you$'
str3 db 'good to see you$'
.code
main proc
main endp
enterkey proc
ret
enterkey endp
end main
F1=Help
Line:13 Col:2
Animation Pane
SUB SCRIBE

Lecture 28 : Procedure, example program of procedure

alisoomro666@gmail.com

```
.code
Main proc

Call enterkey

Main endp
End main


enterkey proc

Mov dx, 10
Mov ah, 2
Int 21h

Mov dx, 13
Mov ah, 2
Int 21h

Ret
enterkey endp

End main
```

What is Procedure?

It is just a block of code that can be called anywhere in the program with name

```
enterkey proc

Mov dx, 10
Mov ah, 2
Int 21h

Mov dx, 13
Mov ah, 2
Int 21h

ret
enterkey endp
```

For Enter

Call enterkey

Hello
How are you
Good to see you

Why do we need it?

- Reusability
- Reduce complexity

How to use Procedure?

Name **proc**

ret
Name **endp**

Call Name

Lecture 27 : Nested Loop, program to print pyramid

alisoomro666@gmail.com

What is Nested Loop?

Loop within Loop

```
mov bx,1
Mov cx, 5
L1:
Push cx
Mov cx, 2
Mov cx, bx
L2:
Mov dx, '*'
Mov ah, 2
Int 21h

Mov dx, 10
mov ah, 2
Int 21h
Mov dx, 13
Mov ah, 2
Int 21h

Loop l2
Inc bx
POP cx
Loop l1
```

```
*
**
***
****
*****
```

Math	Eng	Physics
12	13	30
42	12	40
43	42	12

Why do we need it?

- Reduce complexity
- Maintained Program

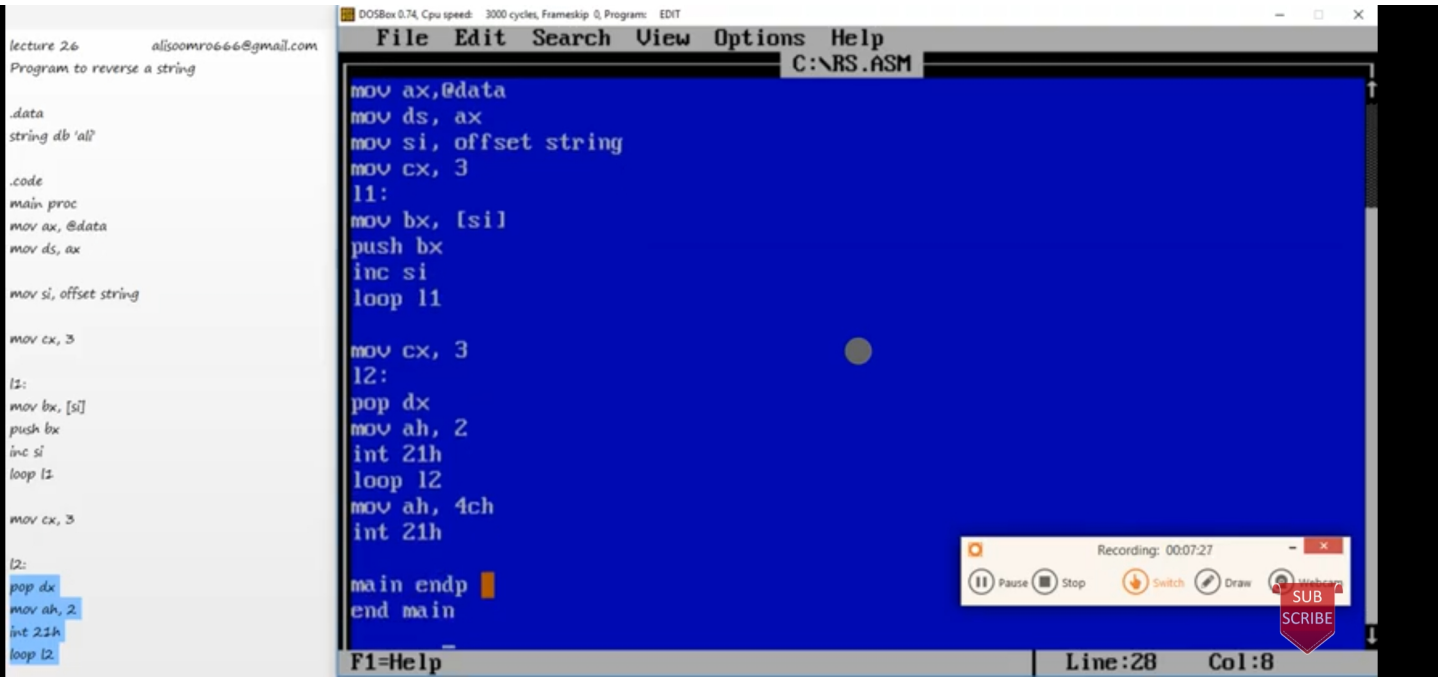
How to use Nested Loop?

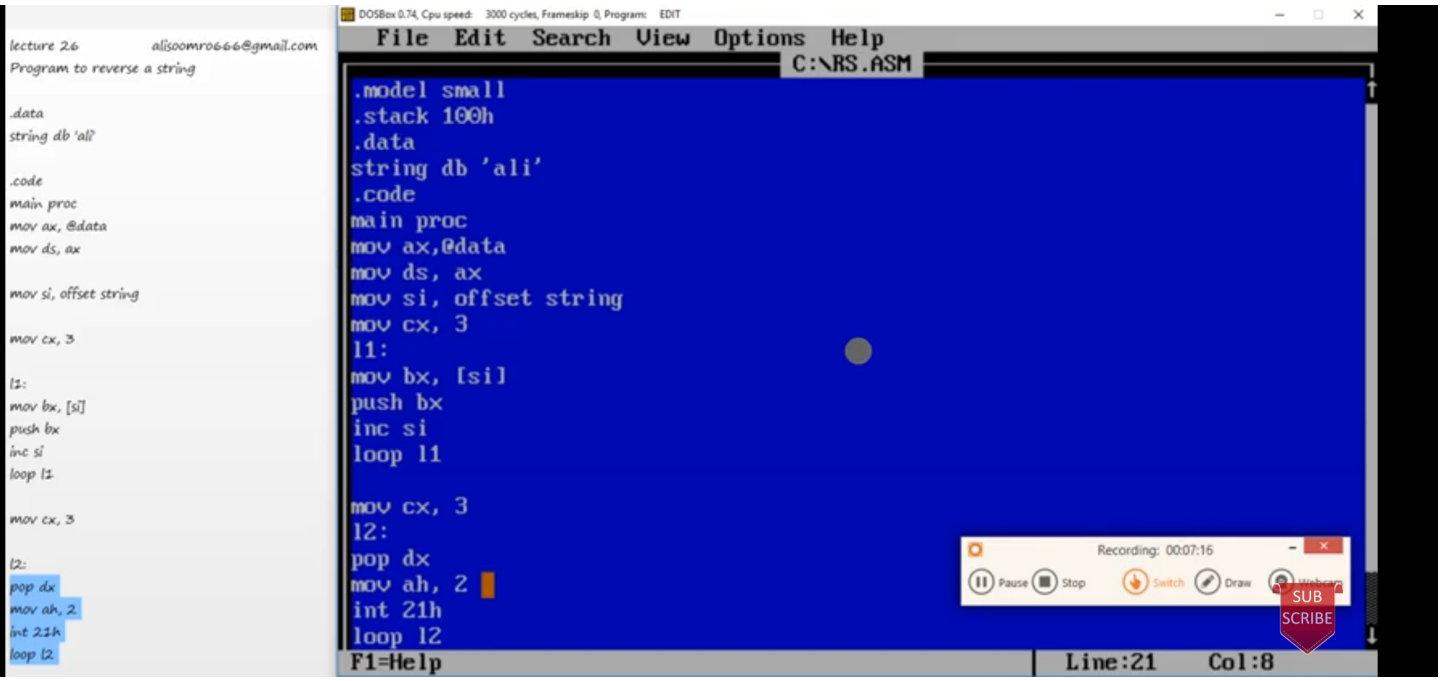
- With the help of PUSH and POP

```
Mov cx, 4
L1:
PUSH cx

Mov cx, 3
l2:

Loop l2

POP cx

Loop l1
```

Main Loop

Nested Loop


lecture 26 alisoomro666@gmail.com
Program to reverse a string
.data
string db 'ali'
.code
main proc
mov ax, @data
mov ds, ax
mov si, offset string
mov cx, 3
l1:
mov bx, [si]
push bx
inc si
loop l1
mov cx, 3
l2:
pop dx
mov ah, 2
int 21h
loop l2
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: EDIT
File Edit Search View Options Help
C:\RS.ASM
mov ax,@data
mov ds, ax
mov si, offset string
mov cx, 3
l1:
mov bx, [si]
push bx
inc si
loop l1
mov cx, 3
l2:
pop dx
mov ah, 2
int 21h
loop l2
mov ah, 4ch
int 21h
main endp
end main
Recording: 00:07:27
Pause
Stop
Switch
Draw
SUB
SCRIBE
F1=Help
Line:28 Col:8


lecture 26 alisoomro666@gmail.com
Program to reverse a string
.data
string db 'ali'
.code
main proc
mov ax, @data
mov ds, ax
mov si, offset string
mov cx, 3
l1:
mov bx, [si]
push bx
inc si
loop l1
mov cx, 3
l2:
pop dx
mov ah, 2
int 21h
loop l2
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: EDIT
File Edit Search View Options Help
C:\RS.ASM
.model small
.stack 100h
.data
string db 'ali'
.code
main proc
mov ax,@data
mov ds, ax
mov si, offset string
mov cx, 3
l1:
mov bx, [si]
push bx
inc si
loop l1
mov cx, 3
l2:
pop dx
mov ah, 2
int 21h
loop l2
Recording: 00:07:16
Pause
Stop
Switch
Draw
SUB
SCRIBE
F1=Help
Line:21 Col:8

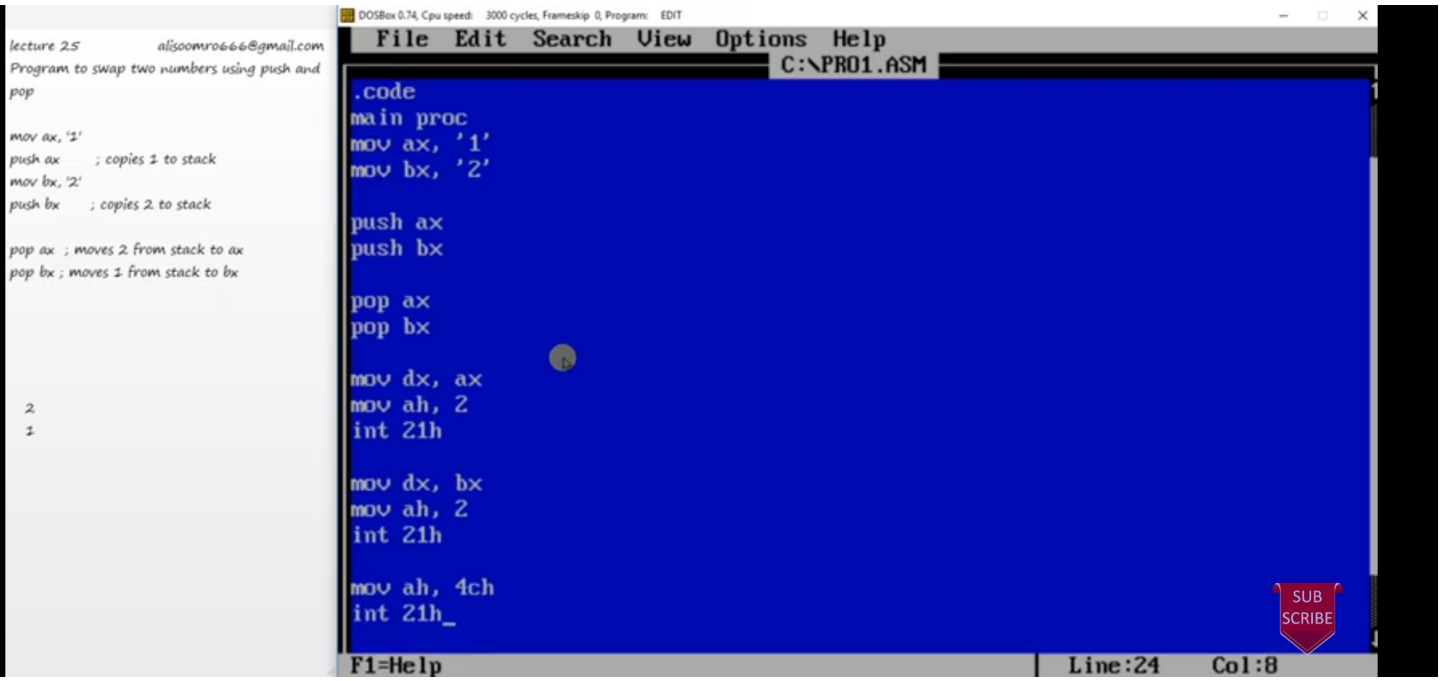
Projects with Codes in Assembly Language Programming

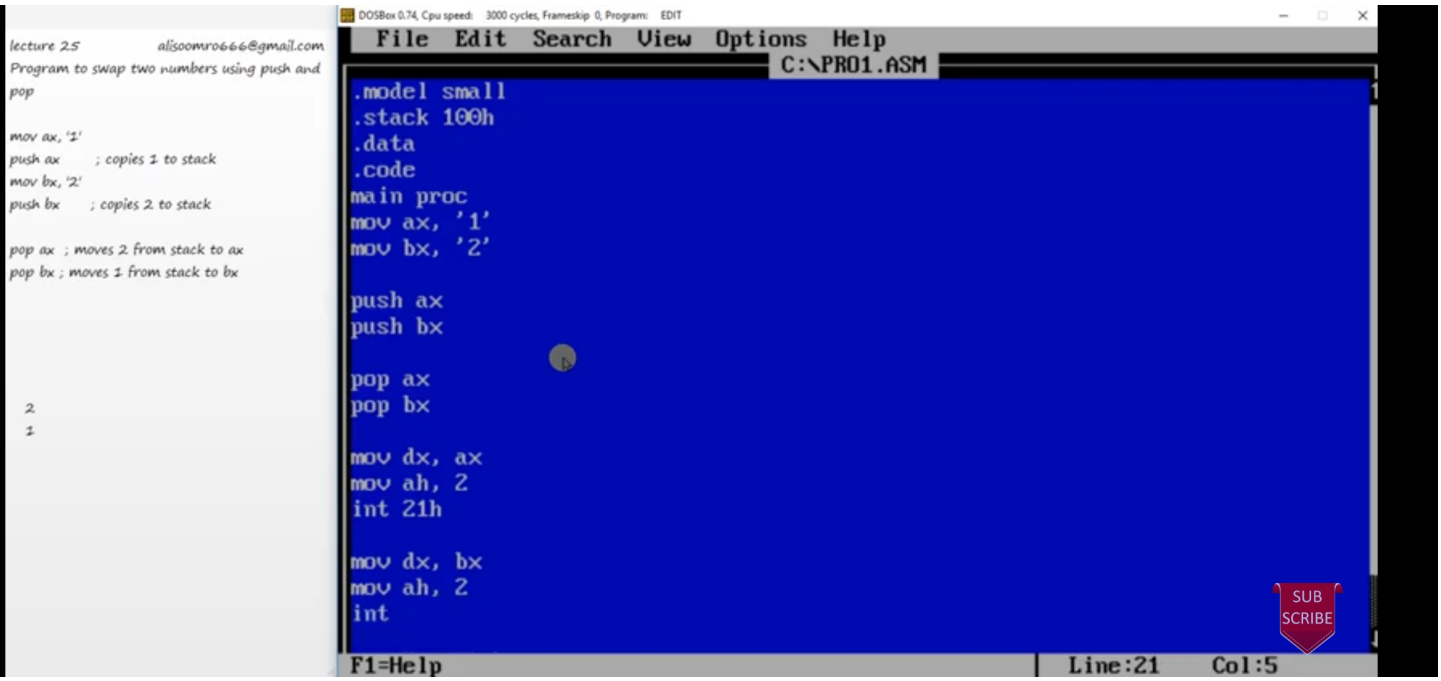
alisoomro666@gmail.com

Links

www.assemblylanguageprojects.blogspot.com

https://drive.google.com/drive/folders/1cEmZX7hlRB0kLKPQ1BrnGIG5U0VTAOz2?usp=sharing


lecture 25 alisoomro666@gmail.com
Program to swap two numbers using push and pop
mov ax, '1'
push ax ; copies 1 to stack
mov bx, '2'
push bx ; copies 2 to stack
pop ax ; moves 2 from stack to ax
pop bx ; moves 1 from stack to bx
2
1
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: EDIT
File Edit Search View Options Help
C:\PRO1.ASM
.code
main proc
mov ax, '1'
mov bx, '2'
push ax
push bx
pop ax
pop bx
mov dx, ax
mov ah, 2
int 21h
mov dx, bx
mov ah, 2
int 21h
mov ah, 4ch
int 21h
SUB SCRIBE
F1=Help
Line:24 Col:8


lecture 25 alisoomro666@gmail.com
Program to swap two numbers using push and pop
mov ax, '1'
push ax ; copies 1 to stack
mov bx, '2'
push bx ; copies 2 to stack
pop ax ; moves 2 from stack to ax
pop bx ; moves 1 from stack to bx
2
1
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: EDIT
File Edit Search View Options Help
C:\PRO1.ASM
.model small
.stack 100h
.data
.code
main proc
mov ax, '1'
mov bx, '2'
push ax
push bx
pop ax
pop bx
mov dx, ax
mov ah, 2
int 21h
mov dx, bx
mov ah, 2
int
SUB SCRIBE
F1=Help
Line:21 Col:5

Lecture 24 : Exam Preparation Material For Assembly Language

alisoomro666@gmail.com
how to get hidden pho...

1. All Assembly Language Programs
2. All Assembly Important Theory Questions
3. Past Papers of Different Universities for Practice
4. My All PowerPoint Presentations
5. Best Book for Assembly Language

Links:

http://bit.do/Assembly

https://goo.gl/6ooV1D

https://drive.google.com/drive/folders/1apAsIvOU9mb1ijshHWwzv0mDODkI0zm1?usp=sharing

By: Ali Hassan Soomro
BSCS from UBIT, University of Karachi
DAE in Electronics from SBTE
Facebook: https://web.facebook.com/AliiHassanSoomro
Gmail: alisoomro666@gmail.com

SUB SCRIBE

Lecture 23

Stack, PUSH, POP and Example Program of Push and Pop

alisoomro666@gmail.com

A stack is a data structure that works on LIFO principle

How to add value to stack and take out ?

Copies content from Operand to top of stack

PUSH Register/Variable

PUSH AX

PUSH Var1

> 16 bits

~~Push Al~~

~~Pop al~~

Pop Register/Variable

Copies content from top of stack to Operand

POP AX

POP Var1

```
.MODEL SMALL
.STACK 100H
.DATA
.CODE
MAIN PROC

mov AX, 2

PUSH AX

POP AX

mov DX, AX
mov ah, 2
Int 21h


mov AH, 4CH
INT 21H
MAIN ENDP
END MAIN
```

Stack Pointer Register
Point the top of space reserved for stack

SS : SP

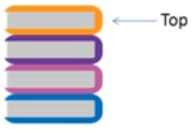
Stack Segment Register
Hold address of space reserved for stack

What's the use of Stack in Computer Science?

1. Undo/Redo
2. Back/Forward
3. Solving mathematical problems with precedence

Why do we study Stack in Assembly Language ?

1. Swap two numbers
2. To reverse a string
3. Helps in Nested Loops (Loop within loop)



How to use stack in assembly program?

.stack 100h

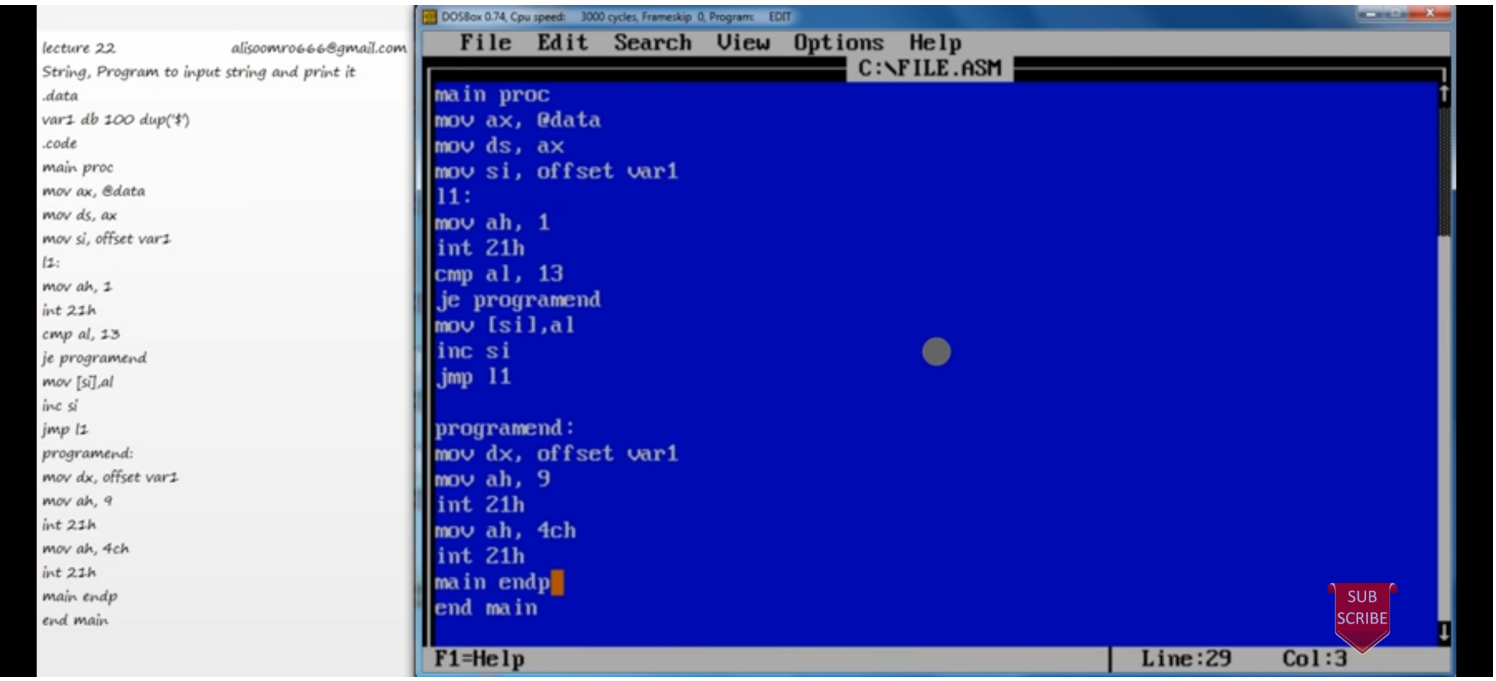
It's a directive/command that reserves 100h bytes for stack

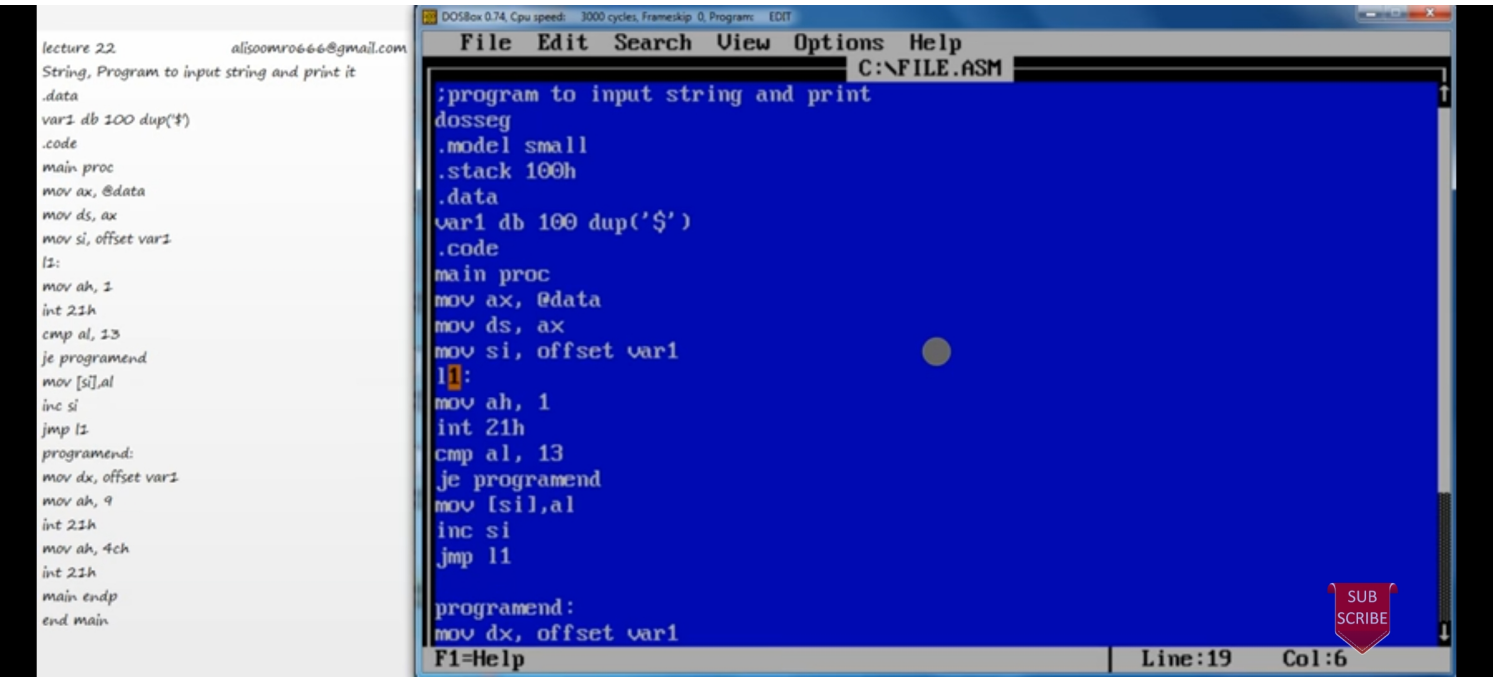
Last In, first out (LIFO)

PUSH → Rakhna

POP → Nikalna

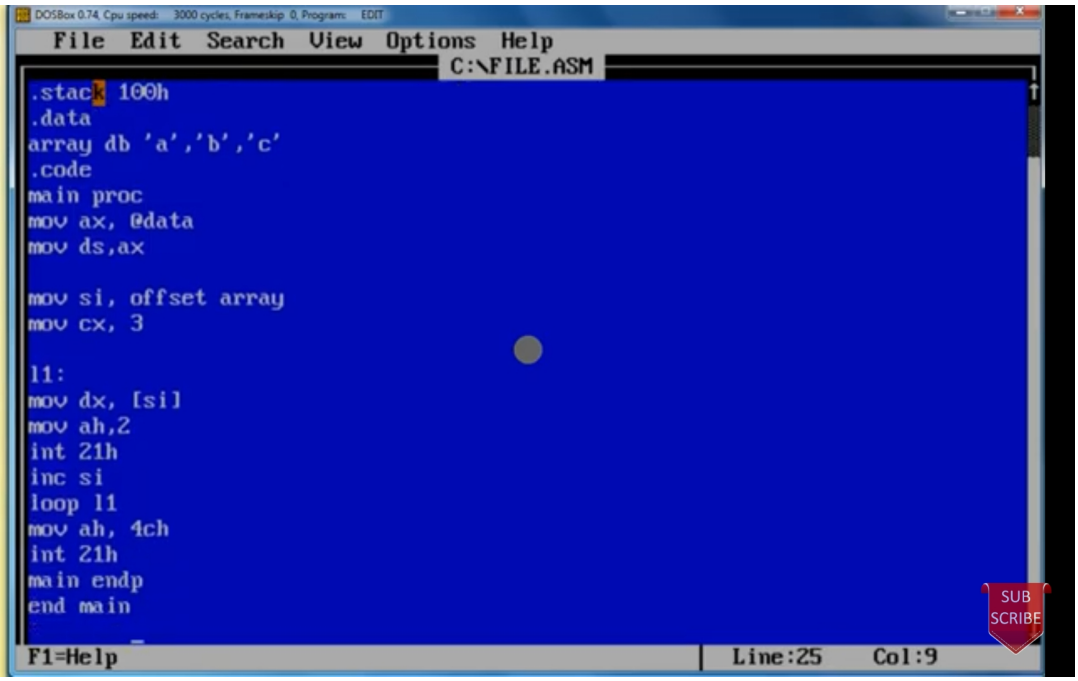
SUB SCRIBE


lecture 22 alisoomro666@gmail.com
String, Program to input string and print it
.data
var1 db 100 dup('$')
.code
main proc
mov ax, @data
mov ds, ax
mov si, offset var1
l1:
mov ah, 1
int 21h
cmp al, 13
je programend
mov [si],al
inc si
jmp l1
programend:
mov dx, offset var1
mov ah, 9
int 21h
mov ah, 4ch
int 21h
main endp
end main
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: EDIT
File Edit Search View Options Help
C:\FILE.ASM
main proc
mov ax, @data
mov ds, ax
mov si, offset var1
l1:
mov ah, 1
int 21h
cmp al, 13
je programend
mov [si],al
inc si
jmp l1
programend:
mov dx, offset var1
mov ah, 9
int 21h
mov ah, 4ch
int 21h
main endp
end main
SUB SCRIBE
F1=Help
Line:29 Col:3


lecture 22 alisoomro666@gmail.com
String, Program to input string and print it
.data
var1 db 100 dup('$')
.code
main proc
mov ax, @data
mov ds, ax
mov si, offset var1
l1:
mov ah, 1
int 21h
cmp al, 13
je programend
mov [si],al
inc si
jmp l1
programend:
mov dx, offset var1
mov ah, 9
int 21h
mov ah, 4ch
int 21h
main endp
end main
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: EDIT
File Edit Search View Options Help
C:\FILE.ASM
;program to input string and print
dosseg
.model small
.stack 100h
.data
var1 db 100 dup('$')
.code
main proc
mov ax, @data
mov ds, ax
mov si, offset var1
l1:
mov ah, 1
int 21h
cmp al, 13
je programend
mov [si],al
inc si
jmp l1
programend:
mov dx, offset var1
SUB SCRIBE
F1=Help
Line:19 Col:6

Lecture 21
Program to print an array using loop

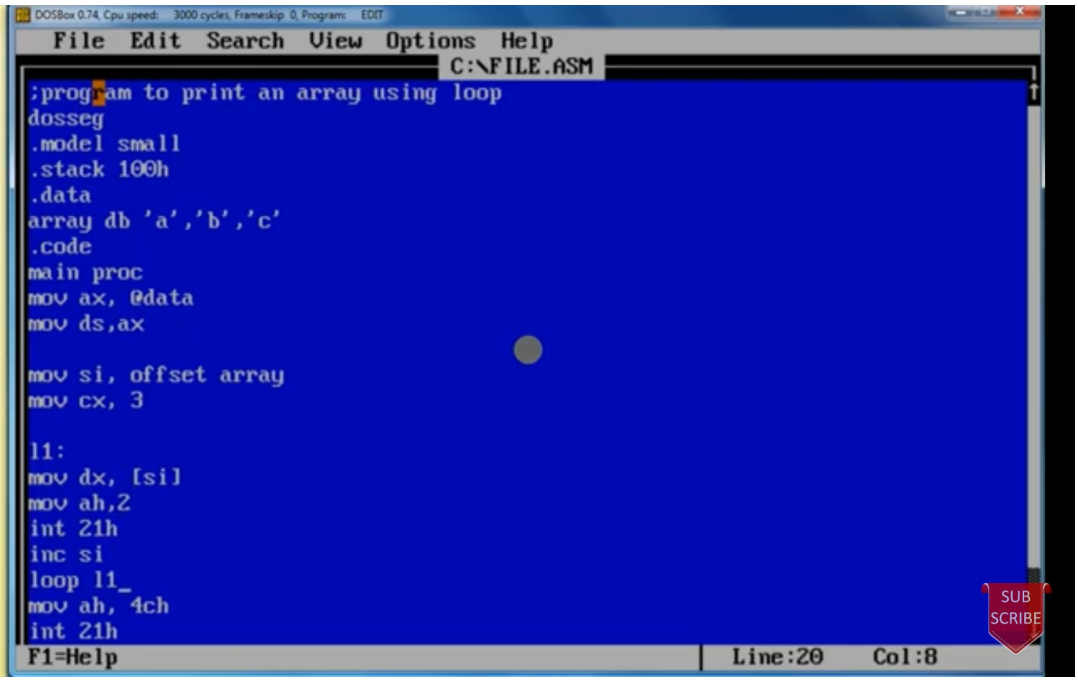
```
.data
array db 'a','b','c'

.code
main proc
mov ax, @data
mov ds,ax

mov si, offset array

mov cx, 3

l1:
mov dx, [si]
mov ah,2
int 21h
inc si
loop l1
mov ah,4ch
int 21h

main endp
end main
```



SUB SCRIBE

Lecture 21
Program to print an array using loop

```
.data
array db 'a','b','c'

.code
main proc
mov ax, @data
mov ds,ax

mov si, offset array

mov cx, 3

l1:
mov dx, [si]
mov ah,2
int 21h
inc si
loop l1
mov ah,4ch
int 21h

main endp
end main
```



Lecture 20

Array, dup and Source Index

alisoomro666@gmail.com

Collection of characters in sequence

Where to initialize?

Array is defined in **.data** directive of program as variable

How to initialize?

In same way as variable but with multiple values

```
Arr1   db   1,2,3,4
Arr1   db   'a', 'b', 'c'
Arr1   db   'abc'
Arr1   db   'a', 'a', 'a'
Arr1   db   ?, ?, ?, ?
Arr1   db   3   Dup('a')
Arr1   db   4   Dup(?)
```

Number of characters

Duplicates the value

SI
Source index register, used as pointer to access array

1, 2, 3, 4

```
Var1 db 1
Var2 db 2
Var3 db 3
Var4 db 4
```

1
2
3
4

a
a
a

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

arr1 db 1, 2, 3, 4

.code
Main proc
Mov ax, @data
Mov ds, ax

Mov si, offset arr1
Mov dx, si

Mov dx, [si]
Mov ah, 2
Int 21h

; mov dx, [si + 1]
Inc si
Mov dx, [si]
Mov ah, 2
Int 21h

Main endp
End main
```

Why do we need to learn Array?

To store many characters with single variable name in sequence in memory

Var1 db 1, 2, 3, 4

How to access array?

Starting address
/
Address of first character

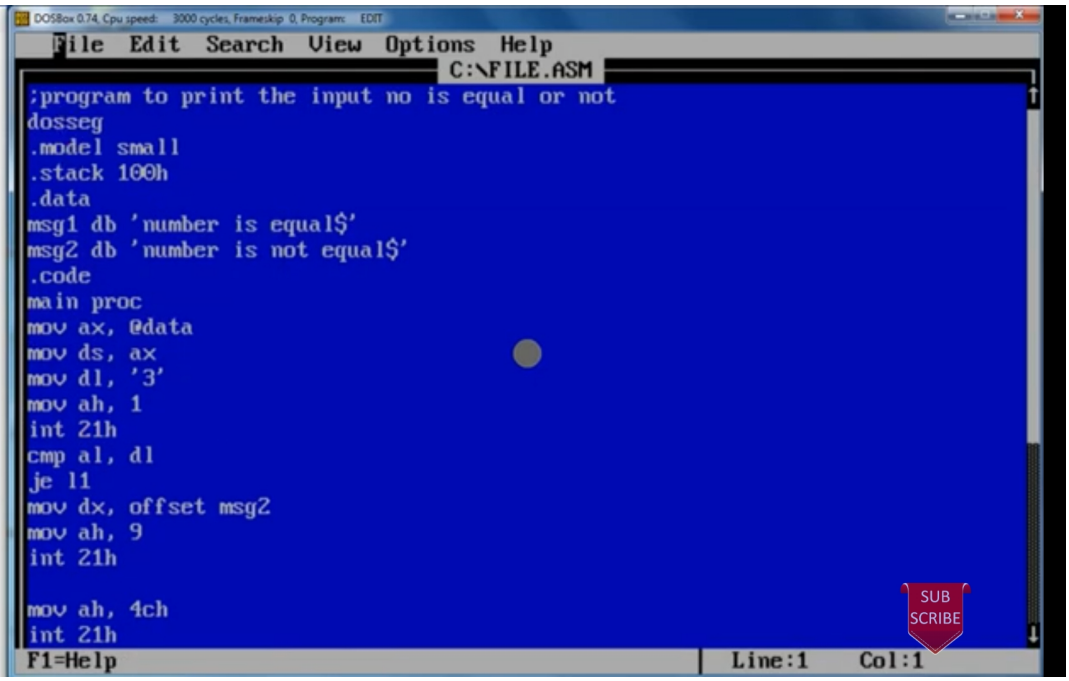
Bracket form to access value at address

Ah000	1
Ah001	2
Ah002	3
Ah003	4

SUB SCRIBE

lecture 19
Program to print the input number is equal or not to given number in program

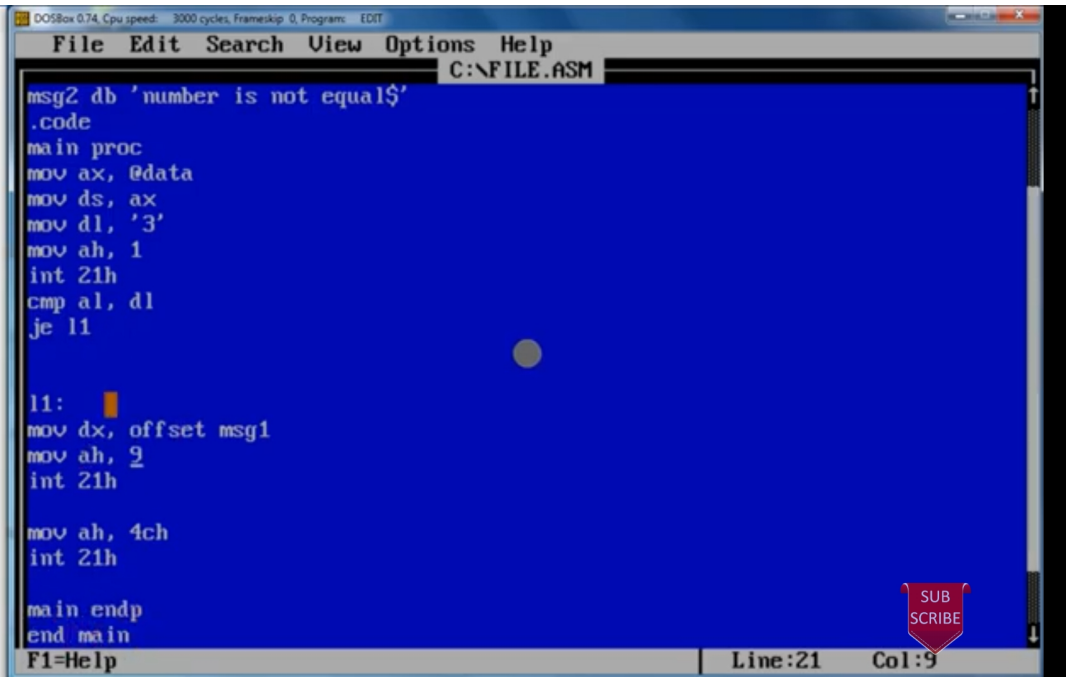
```
.data
msg1 db 'number is equal$'
msg2 db 'number is not equal$'

.code
main proc
mov ax, @data
mov ds, ax
mov dl, '3'
mov ah, 1
int 21h
cmp al, dl
je l1
mov dx, offset msg2
mov ah, 9
int 21h
mov ah, 4ch
int 21h

l1
mov dx, offset msg1
mov ah, 9
int 21h
```



SUB SCRIBE

lecture 19
Program to print the input number is equal or not to given number in program

```
.data
msg1 db 'number is equal$'
msg2 db 'number is not equal$'

.code
main proc
mov ax, @data
mov ds, ax
mov dl, '3'
mov ah, 1
int 21h
cmp al, dl
je l1
mov dx, offset msg2
mov ah, 9
int 21h
mov ah, 4ch
int 21h

l1
mov dx, offset msg1
mov ah, 9
int 21h
```



Lecture 18

alisoomro666@gmail.com

Jump, unconditional jump, conditional jump and Compare

Is a instruction to control the program flow

Unconditional Jump Jump to Label without any condition

Syntax JMP label

Conditional Jump Jump to label when condition occur

Syntax	Opcode Label
JE , JZ	Jump if equal , jump if zero
JNE , JNZ	Jump if not equal , jump if not zero
JL , JB	Jump if less, jump if below
JLE , JBE	Jump if less or equal, jump if below or equal
JG , JA	Jump if greater, jump if above
JGE , JAE	Jump if greater or equal, jump if above of equal

JC , JP , JA , JZ , JS , JT , JI , JD , JO

```
L1:
Mov dl, 'a'
Mov ah, 2
Int 21h

Jmp L1
```

Subtracts operand1 from operand2, but does not store the result; only changes the flags

```
L1:
Mov ah, 1
Int 21h

Mov dl, 3
Cmp al, dl
JE  L1

Mov ah, 4ch
Int 21h
```

Compare

Syntax:

Cmp reg, reg	Cmp dl, al
Cmp reg, constant	Cmp dl, '3'
Cmp reg, [memory address]	Cmp dl, [si]

Al, 3
Dl, 3

Al, 5
Dl, 3

Al, 1
Dl, 3

Jump if ZF = 1

				OF	DF	IF	TF	SF	ZF		AF		PF		CF

SUB SCRIBE

Lecture 17

alisoomro666@gmail.com

Flag Register, Carry flag, parity flag, Auxiliary flag, zero flag, sign flag, trap flag, interrupt flag, direction and overflow flag

Is a register that contains the current state of the processor

				OF	DF	IF	TF	SF	ZF		AF		PF		CF

Useful bits = 9

Status Flags :
To handle the result of an operation.

1. Carry : CF
2. Parity : PF
3. Auxiliary : AF
4. Zero: ZF
5. Sign: Flag

Controls Flags:
To Control the operations of CPU

1. Trap: TF
2. Direction: DF
3. Interrupt: IF

Trap Flag : TF
System use it when debugging is required;
1 : When single step mode (debugging) is needed
0: When single step mode (debugging) is not needed

Zero Flag: ZF
1 : When result is zero
0: when result is not zero

```
00000001  <- 1
00000001  <- 1
--------
00000000  <- 0
```

Interrupt Flag : IF
1 : When interrupt is called
0: when interrupt is not called

Int 21h

```
00000011  <- 3
00000111  <- 7
--------
00000110  <- -4
```

Sign Flag : SF
1 : When result is negative
0: When result is positive

Overflow Flag : OF
1 : When result is too big to fit in the destination
0: When there is not too big to fit in the destination

Add dx, ax

```
Dx  1111111111111111  <- 65536
Ax  1111111111111111  <- 65536
```

Auxiliary Flag
1: When 3rd bit carry exits
0: When 3rd bit carry doesn't exit

Every 3rd bit carry

Carry Flag : CF
1 : When there is last carry out
0: When there is not last carry out

Last carry out

```
1 1 1 1 1 1 1
11111111  <- 255
00000011  <- 3
--------
00000010
```

'hello$'

Direction Flag : DF
1 : Strings automatically decrements the address
0: String does not automatically decrement the address

Parity: PF
1 : When there is even number of bits
0: When there is not even number of bits

Why do we study flag register basically?

Theoretically?

1. What controls the operations of CPU?
2. What handles the status of operations?

In programming?

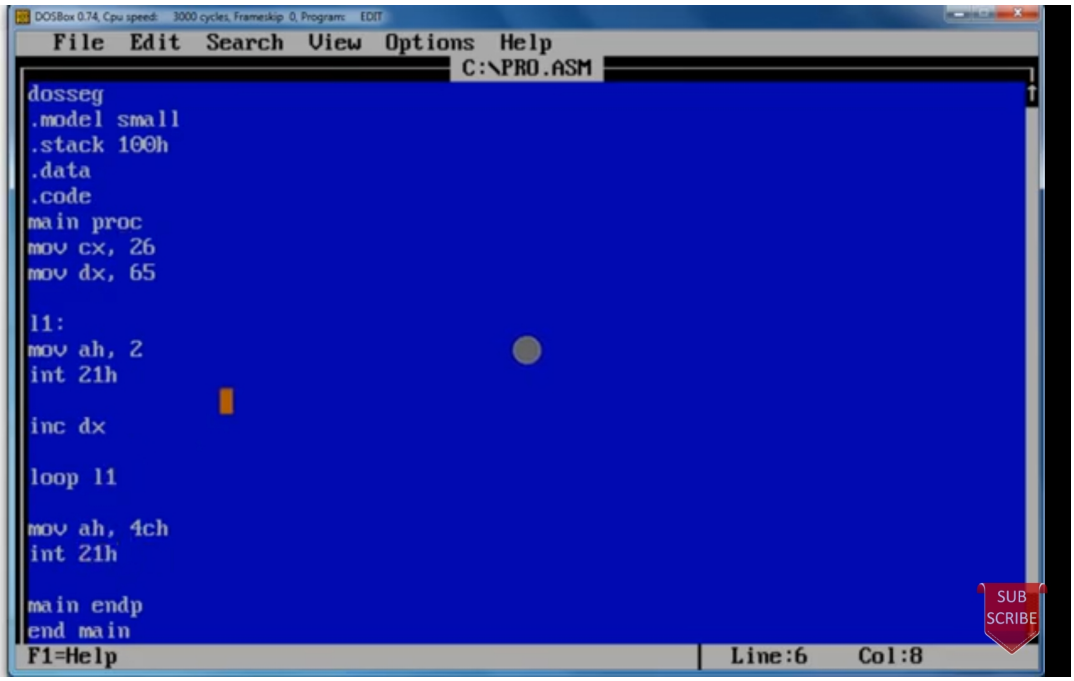
1. Conditional Jump
2. Which number is lesser, greater, or equal

```
Mov dl, 12
Mov al, 10

Mov dx, 'a'
Mov ax, 2
Int 21h
```

SUB SCRIBE

Lecture 16
Program to print capital letters from A to Z Using Loop

```
.code
main proc

mov cx, 26
mov dx, 65

l1:
mov ah, 2
int 21h

inc dx

loop l1
```



Lecture 15:

alisoomro666@gmail.com

Loop, Label, Counter Register, Inc and Program to print 0 to 9

```
dosseg
.model small
.stack 100h
.data
.code
main proc

Mov cx,10
mov dx, 48

L1:
Mov dx, 48

Mov ah, 2
Int 21h

Add dx, 1

Loop L1

Mov ah,4ch
Int 21h

main endp
end main
```

Series of instructions that is repeated until a terminating condition is reached.

```
Mov dx, 'a'
Mov ah, 2
Int 21h
```

Increment by 1

Inc dx

Mov:

Label Syntax

1Test:

General purpose registers,
Main purpose is to be used for a loop

LabelName: → **Counter Register**

Mov CX, 10

```
Mov dx, 'a'
Mov ah, 2
Int 21h

Loop LabelName
```

Works on Decrement By 1

Cx = 10
Cx = 9
Cx = 8
↓
Cx = 0

Label Rules:

Test:
Test1:
T1:

1. A **label** can be placed at the beginning of a statement, because the label is assigned the current value of line

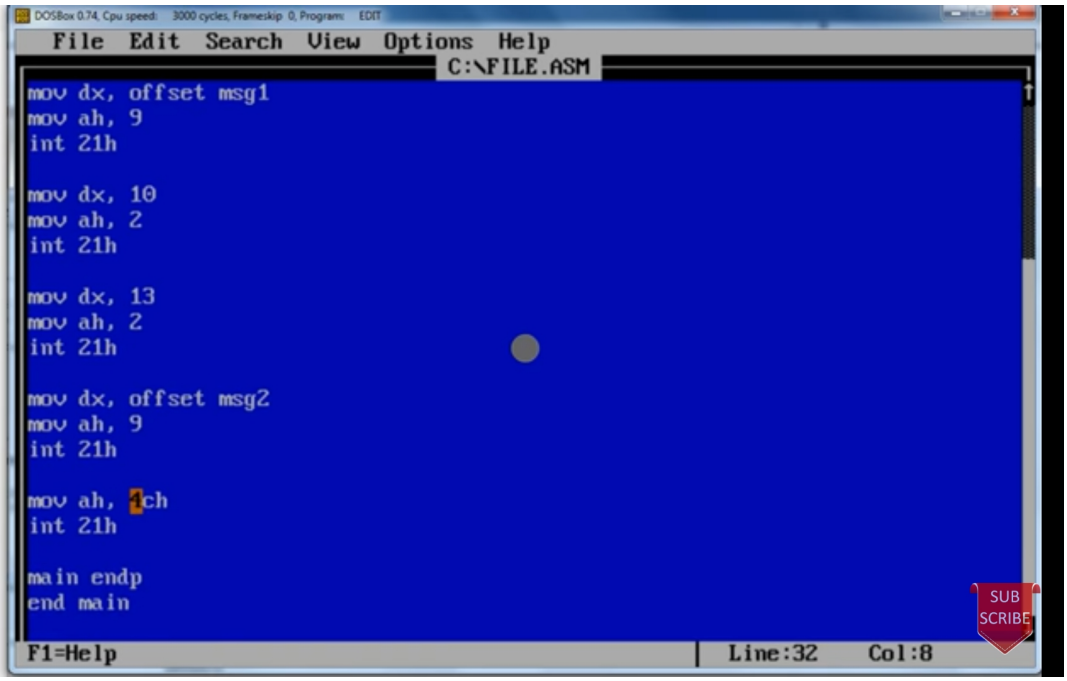
2. Label name must not be a reserved word e.g. Mov, Add, DB and DW

3. Colon : must be used with Label While initializing, but not while calling

SUB SCRIBE

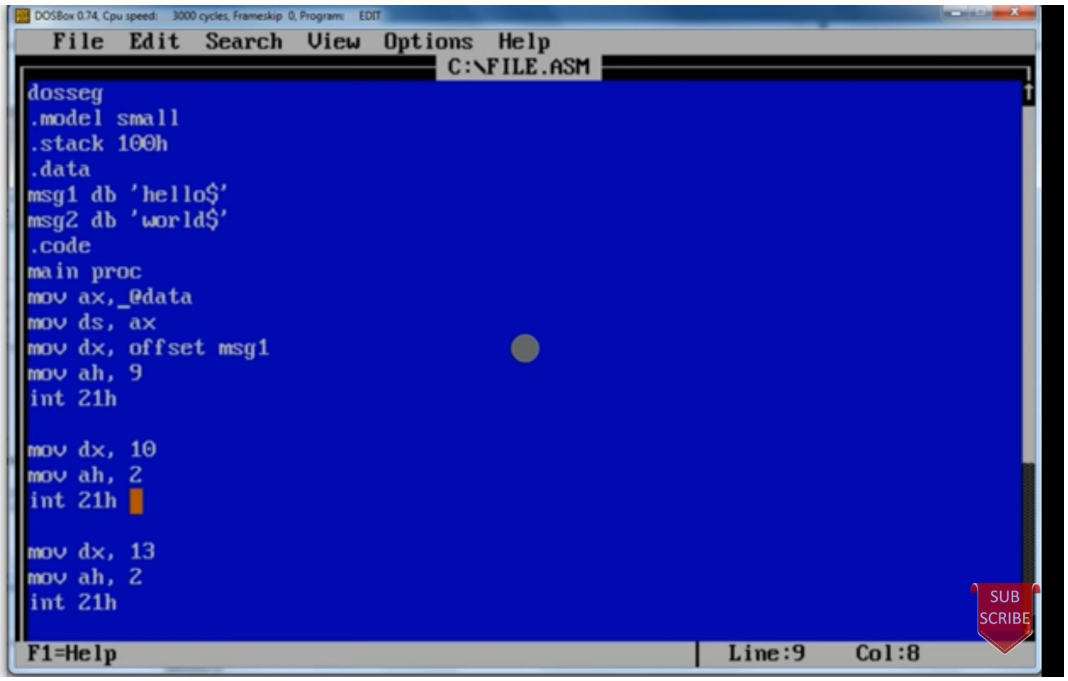
Lecture 14
Program to print two different strings on two different lines
hello
world

```
.data
msg1 db 'hello$'
msg2 db 'world$'

.code
mov ax, @data
mov ds, ax

mov dx, offset msg1
mov ah, 9
int 21h

mov dx, 10
mov ah, 2
int 21h

new line feed : 10
carriage return: 13
```



Lecture 14
Program to print two different strings on two different lines
hello
world

```
.data
msg1 db 'hello$'
msg2 db 'world$'

.code
mov ax, @data
mov ds, ax

mov dx, offset msg1
mov ah, 9
int 21h

mov dx, 10
mov ah, 2
int 21h

new line feed : 10
carriage return: 13
```



Lecture 13:

alisoomro666@gmail.com

Variables, Data Types, Offset and LEA

Where to initialize variables in program?

Variables are defined in **.data** directive of program structure

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

.code
main proc

main endp
end main
```

How to initialize variables?

Initializer Directive

VariableName DataSize Value ← Initializer

```
Var1    db    49
Var1    db    ?
Var1    db    '1'
Var1    db    'A'
Var1    db    '1235$'
Var1    db    'hello world$'
```

AL, BL, CL, DL...
Sub, Add, DIV, MUL
Mov
POP, PUSH

Don't use reserved keywords as VariableName

$ must be used in end of string

$: Terminator, end point of string

Initializer Directive

DB	Define Byte	1 byte, 8 bits
DW	Define Word	2 bytes, 16 bits
DD	Define Double word	4 bytes, 32 bits
DQ	Define QuadWord	8 bytes, 64 bits
DT	Define TenBytes	10 bytes, 80 bits

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

Moves data address to ds so that data segment gets initialized as heap memory to access variables fast

Offset

Holds the beginning address of Variable as 16 bits

Type Mismatch

16 bits — Db : 8 bits

```
Mov dx, var1
```

LEA

Load Effective Address

It is an indirect instruction used as a pointer in which first variable Points the address of second variable

```
Dosseg
.model small
.stack 100h
.data
Var1 db '1'
Var2 db ?
Var3 db '1235$'
.code
Main proc
Mov ax, @data
Mov ds, ax
Mov dl, Var1
Mov ah, 2
Int 21h
Mov Var2, bl
Mov dl, Var3
Mov dx, offset Var3
lea dx, Var3
Mov ah, 9
Int 21h
Main endp
End main
```

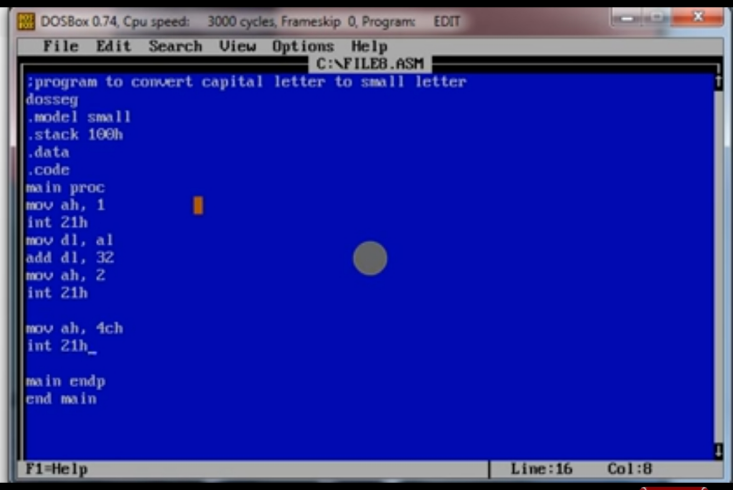
Not for string

First Character From string

Lecture 12 alisoomro666@gmail.com

Program to convert capital letter to small letter

```
mov ah, 1
int 21

A = 65
B = 66
a = 97
b= 98

97-65 = 32
```



SUB SCRIBE

Lecture 11 alisoomro666@gmail.com

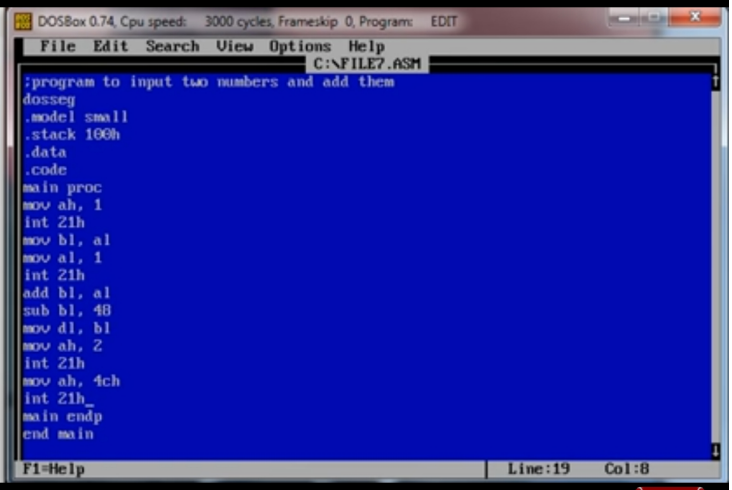
Program to input two numbers and add them

```
mov ah, 1
int 21h
mov bl, al
mov ah, 1
int 21h
add bl, al
sub bl, 48
```

1 = 49

2= 50

3 = 99 - 48 = 51



SUB SCRIBE

lecture 10 alisoomro666@gmail.com

Program to subtract two numbers

3 - 1 = 2

```
mov bl, 3
mov cl, 1
sub bl, cl

add bl, 48

mov dl, bl
```

SUB SCRIBE

lecture 10 alisoomro666@gmail.com

Program to subtract two numbers

3 - 1 = 2

```
mov bl, 3
mov cl, 1
sub bl, cl

add bl, 48

mov dl, bl
```

SUB SCRIBE

lecture 9 alisoomro666@gmail.com

Program to add two numbers

1 +2 = 3

mov bl, 1

mov cl, 2

ADD bl, cl

add bl, 2

add bl, 48

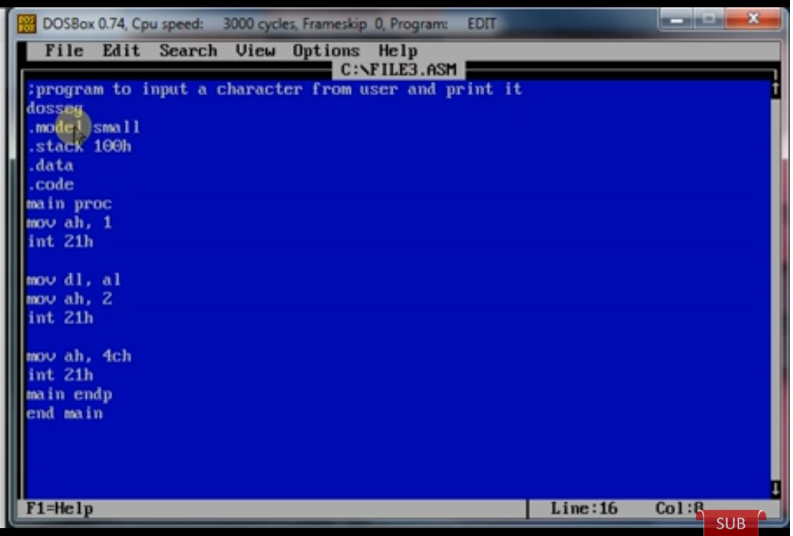
3 + 48 = 51

SUB SCRIBE

Lecture 8

program to input a character from user and print it

```
mov ah, 1
int 21h

mov dl, al
mov ah, 2
int 21h
```



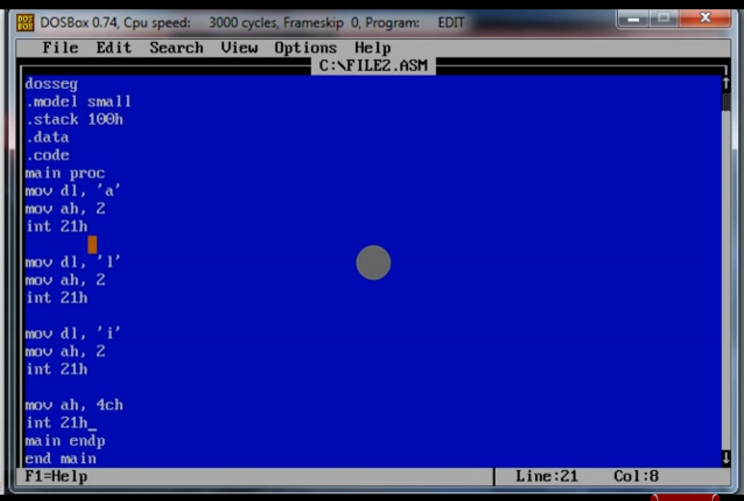
SUB SCRIBE

what can a hacker do...

Lecture 7 alisoomro666@gmail.com

Program to print a name with characters

a l i



SUB SCRIBE

Lecture 6:

alisoomro666@gmail.com

DosBox, MASM, LINK, DOSbox Commands and Program to Print a single character

What is DosBox and why are we installing it?

→ Emulator → In July 22, 2002

→ By: Peter Veenstra

Why DosBox? Why not MASM Emulator?
Why not Emu8086 ?
Why not Visual Studio with Kip Irvine Library?

→ Free Software

→ Light, Simple and easy to use.

→ Run on every environment – Windows, mac, linux and Android

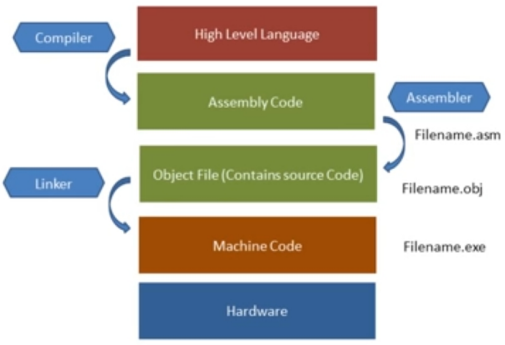
→ Can learn debugging

→ Can grip on syntax

Download Link?

→ Programologysoftwares.blogspot.com

Convert Assembly Code to executable code

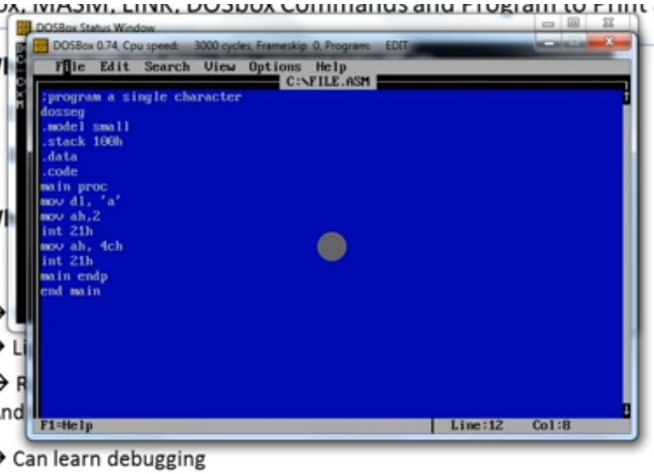


DOSbox Commands:

→ Edit Filename.asm
→ MASM Filename.asm;
→ LINK Filename.obj;
→ Filename.exe

SUB SCRIBE

DOSBox, MASM, LINK, DOSbox Commands and Program to Print a single character



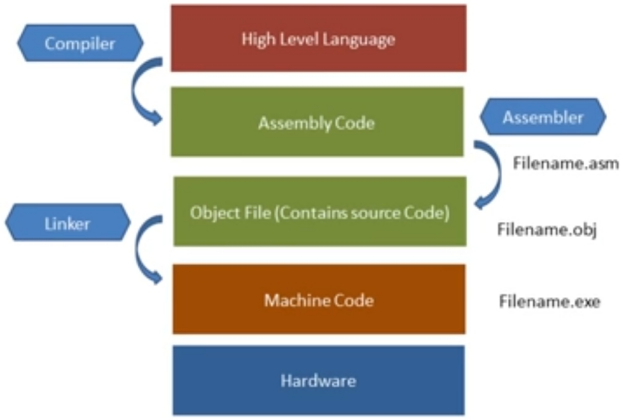
→ Can learn debugging

→ Can grip on syntax

Download Link?

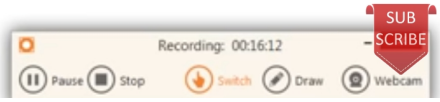
→ Programologysoftwares.blogspot.com

Convert Assembly Code to executable code



DOSbox Commands:

→ Edit Filename.asm
→ MASM Filename.asm;
→ LINK Filename.obj;

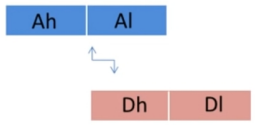


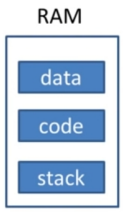
Lecture 5:
Program Structure, Syntax and Program to print a single character on screen

alisoomro666@gmail.com

```
;Program to print a single character on screen
dosseg          <-- DOS Segment <-- Manages the arrangement of segments in a program
.model small    <-- Model Directive <-- Specifies the total amount of memory the program would take.
.stack 100h     <-- Stack Segment Directive <-- Specifies the storage for stack
.data           <-- Data Segment Directive
;variables are defined here
.code           <-- Code Segment Directive
Main proc
Mov dl, 'A'
Mov ah, 2
INT 21h         ;here we write our program, executable instructions

Mov ah, 4CH
INT 21h

Main endp
End Main
```

Mov 'B', 'A' ✗
Mov 2, 3 ✗
Mov dl, AX ✗
Mov dl, 'A'
Mov dx, AX
Mov dl, 2
Mov dh, al





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

Syntax Rules

→ Space after Opcode
→ One operand must be general purpose register
→ Operands must be of same size
→ Comma , between operands
→ Comment must start with a semi colon ;

SUB SCRIBE

Lecture 4:

alisoomro666@gmail.com

Addressing Modes, Mov Instruction, Service Routine, ASCII Code and Interrupt

Addressing Modes

↳ Ways/Models to access data

Registers Addressing: Both operands are registers

Immediate Addressing: One Operand is constant term

Memory Addressing: Access static data directly

Opcode (Operational Code)

2 + 2 Operands

Opcode Reg1, Reg2
Add DL, Al

Opcode Reg, Value
Add Dl, 2

Opcode Reg, [Address]
Add Dl, [address]

Data Transfer Instruction

Mov DL, 2 DL, 'A'

Mov Ah, 2 ← **Service Routine**

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

4ch = Exit

String

Interrupt

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 Video/Graphics Handling

Example 1: Output

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

Example 2: Input

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

ASCII Code

(American Standard Code for Information Interchange)

↳ Character Encoding Scheme

↳ By: American Standards Association (ASA)
Published in; 1963

A = 65 B = 66 → Z=90
a = 91 b = 92 → z=122
0 = 48 1 = 49 → 9=57

Next Line Feed = 10
Carriage Return = 13

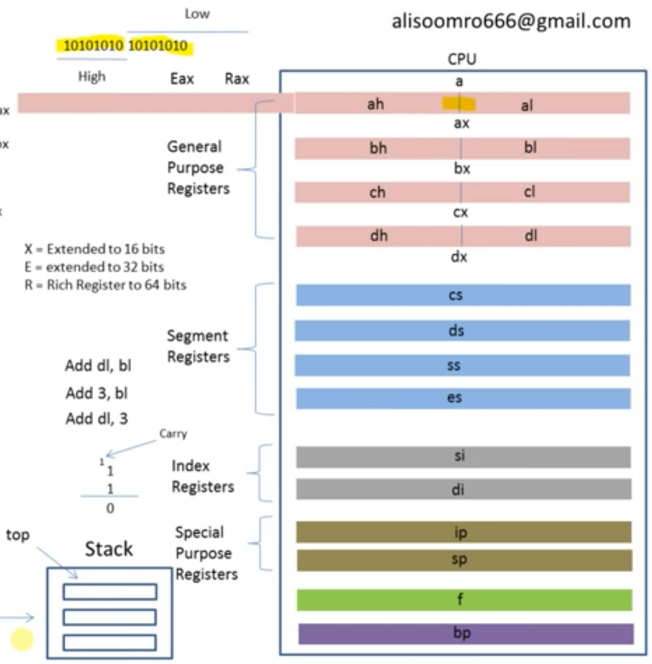
SUB SCRIBE

Lecture 3: Types of Registers

alisoomro666@gmail.com

There are 14 types of registers

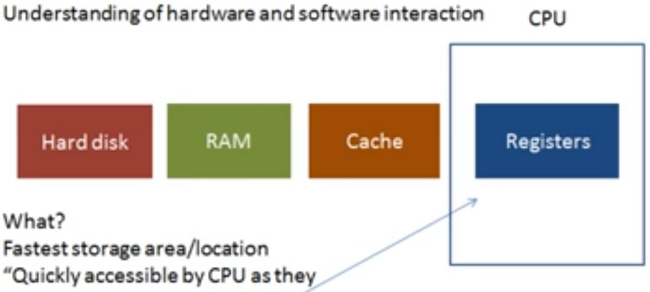
1. **Accumulator** ← Input / output, Operations, a, ax, eax, rax
2. **Base** ← Holds address of data b, bx, ebx, rbx
3. **Counter** ← Counts, used in loop cx, ecx, rcx
4. **Data** ← Holds data for output d, dx, edx, rdx
5. **Code Segment** ← Holds address of code segment
6. **Data Segment** ← Holds address of data segment
7. **Stack segment** ← Holds address of stack segment
8. **Extra segment** ← Holds address of data segment
9. **Source Index** ← Points the source operand
10. **Destination Index** ← Points the destination operand
11. **Instruction Pointer** ← Holds the next instruction
12. **Stack Pointer** ← Points current top of stack
13. **Flag registers** <- Holds current status of the program
14. **Base pointer** ← Base of the top of stack



Lecture 2: Registers

alisoomro666@gmail.com

Why?
Optimization of Processing time
Understanding of hardware and software interaction



What?
Fastest storage area/location
"Quickly accessible by CPU as they are built into CPU.

Record or collection of information

What's the origin of Registers?

Intel 4004 in 1971, Federico Faggin

SUB
SCRIBE

Lecture 1. Introduction to Assembly

alisoomro666@gmail.com

Why?

1. Better/deep understanding of software and hardware interaction
2. Optimization of processing time
3. Embedded Programming
4. Course requirement

What? → Language used to make computer programs → Set of instructions

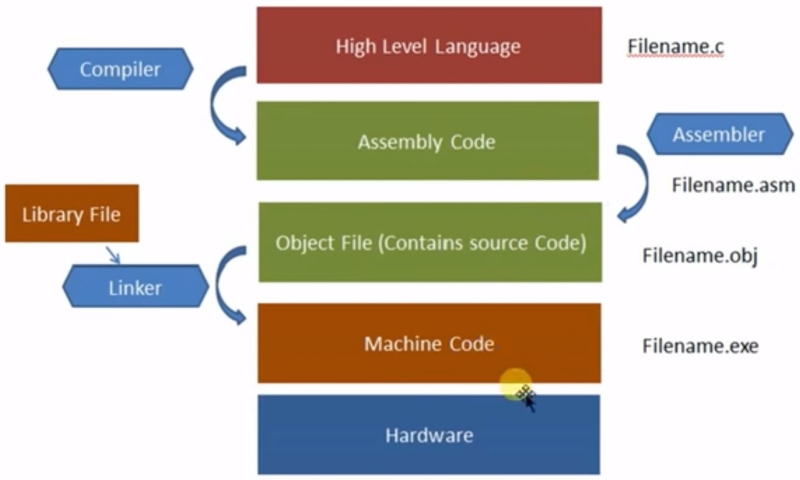
1. Computer Programming Language
2. Low level → Close to Machine
3. Mnemonics/keywords

What's the meaning of Assembly language? **David John Wheeler**

SUB SCRIBE

Lecture 1. Introduction to Assembly

alisoomro666@gmail.com



SUB
SCRIBE