- 1. Install Assembly Language Compiler (emu8086) and get familiar with it.
- 2. Basic Structure –

.MODEL SMALL //for code size .STACK 100H //memory

.DATA .CODE

**MAIN PROC** 

MOV AH,4CH INT 21H PROGRAM HAS RETURNED CONTROL
TO THE OPERATING SYSTEM

MAIN ENDP END MAIN

3. Input/Output –

.MODEL SMALL //for code size .STACK 100H //memory

.DATA .CODE

MAIN PROC

MOV AH,1 ;input

INT 21H

MOV AH,2 ;output

MOV DL,AL ;keyboard input automatically stored in AL register

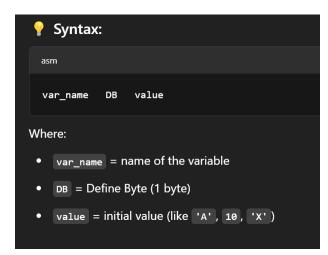
;MOV DL,'M' ;DL register used for output

INT 21H

MOV AH,4CH INT 21H

MAIN ENDP END MAIN

## 4. Variables -



Directive	Meaning
DB	Define Byte (1 byte)
DW	Define Word (2 bytes)
DD	Define Double Word (4 bytes)
DS	Data Segment Register
MOV	Move data into a register or memory

### Code:

.MODEL SMALL .STACK 100H

.DATA

char1 DB 'X' ; variable to hold a character

.CODE

MAIN PROC

MOV AX, @DATA

MOV DS, AX ; initialize data segment

MOV DL, char1 ; load char1 into DL

MOV AH, 02H

INT 21H ; print the character

MOV AH, 4CH

INT 21H ; exit

MAIN ENDP

**END MAIN** 

5. String Output -

.MODEL SMALL //for code size

.STACK 100H //memory

.DATA

msg DB 'Hello, world!\$' ;String must end with \$ sign

.CODE

MAIN PROC

MOV AH, 09H ;09 for string output

LEA DX, msg ; Load address of string

INT 21H ; Display string

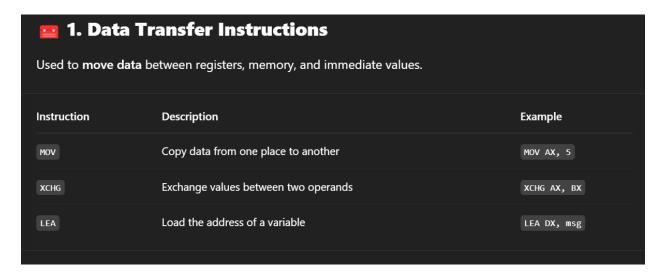
MOV AH,4CH

INT 21H

MAIN ENDP

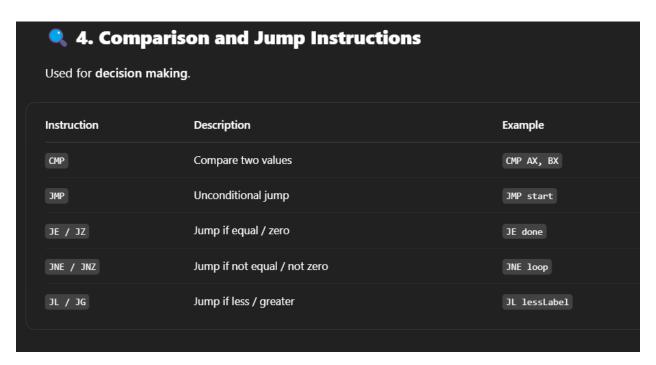
**END MAIN** 

### 6. BASIC Instructions



# + 2. Arithmetic Instructions Used to perform calculations. Instruction Description Example Add two values ADD ADD AX, BX SUB Subtract SUB AX, 10 Increment (add 1) INC CX INC Decrement (subtract 1) DEC DEC CX **Unsigned multiply** MUL BL $(AX = AL \times BL)$ MUL Unsigned divide DIV DIV CL

🔼 3. Logi	<b>3. Logical Instructions</b>			
Used for bit-leve	Used for bit-level operations.			
Instruction	Description	Example		
AND	Bitwise AND	AND AL, 0Fh		
OR	Bitwise OR	OR AL, 0Fh		
XOR	Bitwise XOR	XOR AL, AL (clears AL)		
NOT	Bitwise NOT (inverts bits)	NOT AL		



☐ 7. Program Control Instructions		
Instruction	Description	
INT 21H	Call DOS service	
RET	Return from procedure	
CALL	Call a procedure	
HLT	Stop the processor	
NOP	Do nothing (No Operation)	

# 7. STACK segment –



The stack segment is a special area of memory used to store data temporarily, following the LIFO principle:

LIFO = Last In, First Out

So, the last item pushed into the stack will be the first one popped out.

# Why Use the Stack?

- Store return addresses for function calls
- Store register values temporarily
- Store parameters and local variables
- Allow for nested function calls

# Register Purpose Stack Segment register SP Stack Pointer (points to top of stack) BP Base Pointer (often used to access stack variables)