8086 ASSEMBLY LANGUAGE PROGRAMMING

INSTRUCTION SET OF 8086
The 8086 instructions are categorized into the following main types

(i) Data copy /transfer instructions: These types of instructions are used to transfer data from source operand
to destination operand. All the store, load, move, exchange input and output instructions belong to this
category.

(ii) Arithmetic and Logical instructions: All the instructions performing arithmetic, logical, increment,
decrement, compare and ASCII instructions belong to this category.

(iii) Branch Instructions: These instructions transfer control of execution to the specified address. All the call,
jump, interrupt and return instruction belong to this class.

(iv) Loop instructions: These instructions can be used to implement unconditional and conditional loops. The
LOOP, LOOPNZ, LOOPZ instructions belong to this category.

(v) Machine control instructions: These instructions control the machine status. NOP, HLT, WAIT and LOCK
instructions belong to this class.

(vi) Flag manipulation instructions: All the instructions which directly affect the flag register come under this
group of instructions. Instructions like CLD, STD, CLI, STI etc.., belong to this category of instructions.
(vii) Shift and Rotate instructions: These instructions involve the bit wise shifting or rotation in either direction
with or without a count in CX.

(viii) String manipulation instructions: These instructions involve various string manipulation operations like
Load, move, scan, compare, store etc.,

1. Data Copy/ Transfer Instructions:

The following instructions come under data copy / transfer instructions:

MOV	PUSH	POP	IN	OUT	PUSHF	POPF	LEA	LDS/LES	XLAT
XCHG	LAHF	SAHF							

Data Copy/ Transfer Instructions:
MOV:
MOVE: This data transfer instruction transfers data from one register / memory location to another register /
memory location. The source may be any one of the segment register or other general purpose or special purpose
registers or a memory location and another register or memory location may act as destination. وظيفته نقل البيانات من المصدر الى الوجهة

Syntax:

1) MOV mem/reg1, mem/reg2

[mem/reg1] $\leftarrow$ [mem/reg2]

Ex: MOV BX, 0210H
MOV AL, BL
MOV [SI], [BX] ◊ is not valid Memory uses DS as segment register. No memory-to-memory operation is allowed. It won't affect flag bits in the flag register.

2)

MOV mem,data

[mem] $\Rightarrow$ data

Ex: MOV [BX], 02H

MOV [DI], 1231H

فكر وأجب؟

ما معنى الاقواس في المسجل؟
نفذها بدون اقواس واجب

MOV reg, data

[reg] $\Leftarrow$ data

Ex: MOV AL, 11H

MOV CX, 1210H

MOV A, mem

[A] $\Leftarrow$ [mem]

Ex: MOV AL, [SI]

MOV AX, [DI]

MOV mem, A

[mem] $\Leftarrow$ A

A $\Leftarrow$: AL/AX

Ex: MOV [SI], AL

MOV [SI], AX

Ex: MOV DX, SS

In the case of immediate addressing mode, a segment register cannot be destination register. In other words, direct loading of the segment registers with immediate data is not permitted. To load the segment registers with immediate data, one will have to load any general-purpose register with the data and then it will have to be moved to that particular segment register.

Ex: Load DS with 5000H

1) MOV DS, 5000H; Not permitted (invalid) Thus to transfer an immediate data into the segment register, the convert procedure is given below:

2) MOV AX, 5000H MOV DS, AX

Both the source and destination operands cannot be memory locations (Except for string instructions) Other MOV instructions examples are given below with the corresponding addressing modes

3) MOV AX, 5000H; Immediate
4) MOV AX, BX; Register
5) MOV AX, [SI]; Indirect
6) MOV AX, [2000H]; Direct
7) MOV AX, 50H[BX]; Based relative, 50H displacement

PUSH:

Push to Stack: This instruction pushes the contents of the specified register/memory location on to the stack. The stack pointer is decremented by 2, after each execution of the instruction. The actual current stack-top is always occupied by the previously pushed data. Hence, the push operation decrements SP by two and this store the two-byte contents of the operand

onto the stack. The higher byte is pushed first and then the lower byte. Thus, out of the two decremental stack addresses the higher byte occupies the higher address and the lower byte occupies the lower address.
وظيفتها وضع البيانات في المكدس والمكدس هو منطقة ذاكرة تخزن فيها البيانات مؤقتا بطريقة اخر\ داخل,اول خارج

Syntax:

PUSH reg
$[SP] \leftarrow [SP]-2$
$[[S]] \leftarrow [reg]$

Ex:

1) PUSH AX
2) PUSH DS
3) PUSH [5000H]; content of location 5000H & 5001H in DS are pushed onto the stack.

بمقدار 2 (لان كل عملية بوش تحفظ 16 بت=2 sp\كل مرة يتم تنفيذ العملية يقل المؤشر بايت)ويخزن العنصر في المكدس

POP: **Pop from stack:** This instruction when executed, loads the specified register / memory location with the contents of the memory location of which address is formed using the current stack segment and stack pointer as usual. The stack pointer is incremented by 2. The POP instruction serves exactly opposite to the PUSH instruction.
عكس عملية البوب اي اخراج البيانات من المكدس الى سجل او موقع ذاكرة
بمقدار 2 ليسترجع اخر قيمة ادخلت spكل مرة يتم تنفيذ البوب يزيد المؤشر

Syntax:

i) POP mem

$[SP] \Leftarrow [SP] +2$

$[mem] \Leftarrow [[SP]]$

ii) POP reg

$[SP] \Leftarrow [SP] + 2$

$[reg] \Leftarrow [[SP]]$

Ex:

1. POP AX
2. POP DS
3. POP [5000H]

Ex: لنفترض القيم التالية كبداية

القيمة الابتدائية	العنصر
غير محددة	AX
2000H	SP (STACK POINTER)
3000H	SS (STACK SEGMENT)
العنوان 3000:2000H	مكدس الذاكرة يبدا من

MOV AX,1234H : ضع القيمة المذكورة في السجل AX
PUSH AX : اخزن في المكدس وبالتالي ينقص المؤشر بمقدار 2 AX

القيمة	السجل
1234H	AX
1FFEH	SP

1234H	الذاكرة عندSS:1FFEH

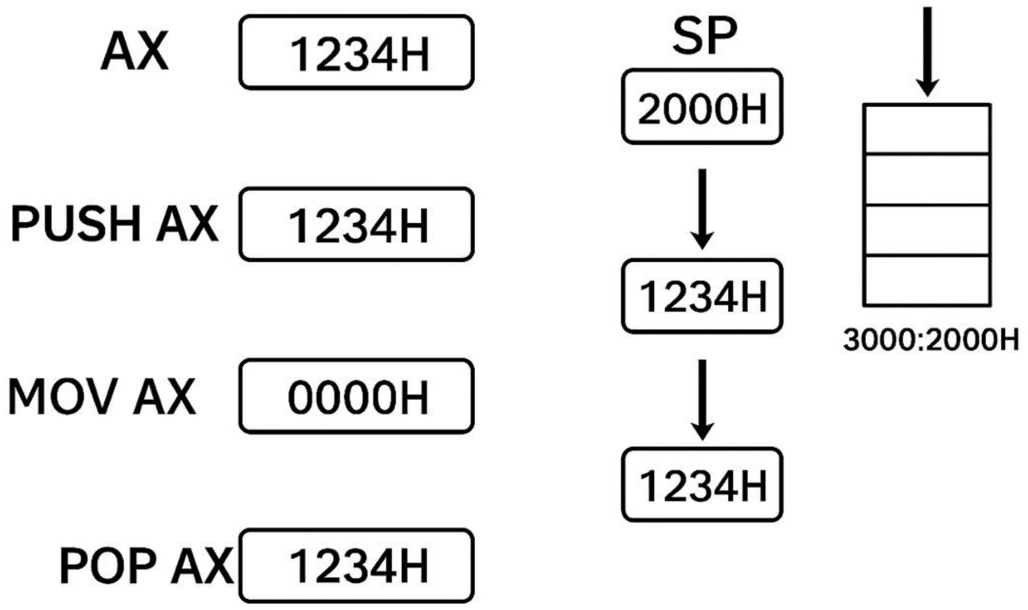
SP=SP-2= 2000H=1FFEH
MEMORY(SS:1FFEH) =1234H

MOV AX,0 : اجعلAX=0

القيمة	السجل
0000H	AX
1FFEH	SP

POP AX : استرجع القيمة الاصلية 1234
وبالتالي يزيد المؤشر بمقدار 2

القيمة	السجل
1234H	AX
2000H (عاد كما كان)	SP


AX
1234H
PUSH AX
1234H
MOV AX
0000H
POP AX
1234H
SP
2000H
1234H
1234H
3000:2000H