University of Diyala
College of engineering
Department of computer Engineering
Second class

microprocessor Programming

Lecture 5

8086 instructions set

Presented by

Lecturer : Abdullah Thair Abdalsatir
Department of computer Engineering
University of Diyala

Lecture 5

Microprocessor programming

content

❖8086 Supports 6 Types Of Instructions.

1. Data Transfer Instructions
2. Arithmetic Instructions
3. Logical Instructions
4. String Manipulation Instructions
5. Process Control Instructions
6. Control Transfer Instructions

1. Data Transfer Instructions

Instructions that are used to transfer data/ address in to registers, memory locations and I/O ports.

Generally involve two operands: Source operand and Destination operand of the same size.

Source: Register or a memory location or an immediate data
Destination : Register or a memory location.

The size should be a either a byte or a word.

A 8-bit data can only be moved to 8-bit register/ memory and a 16-bit data can be moved to 16-bit register/ memory.

1. Data Transfer Instructions

Mnemonics: **MOV, XCHG, PUSH, POP, IN, OUT …**

MOV reg2/ mem, reg1/ mem	
MOV reg2, reg1 **MOV mem, reg1** **MOV reg2, mem**	**(reg2) ← (reg1)** **(mem) ← (reg1)** **(reg2) ← (mem)**
MOV reg/ mem, data	
MOV reg, data **MOV mem, data**	**(reg) ← data** **(mem) ← data**

Example:

```
ORG 100h
MOV AX, 0B800h    ; set AX = B800h (VGA
memory).
MOV DS, AX        ; copy value of AX to DS.
MOV CL, 'A'       ; CL = 41h (ASCII code).
MOV CH, 01011111b ; CL = color attribute.
MOV BX, 15Eh      ; BX = position on screen.
RET               ; returns to operating system.
```

XCHG reg2/ mem, reg1	
XCHG reg2, reg1 **XCHG mem, reg1**	**(reg2) ↔ (reg1)** **(mem) ↔ (reg1)**

Example:

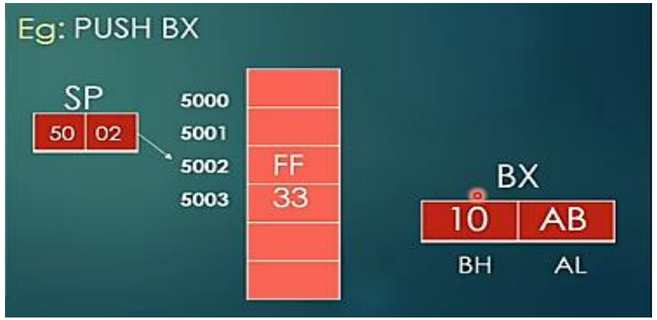
```
MOV AL, 5
MOV AH, 2
XCHG AL, AH   ; AL = 2, AH = 5
XCHG AL, AH   ; AL = 5, AH = 2
RET
```

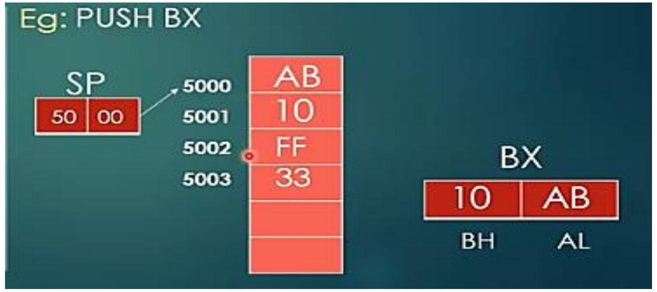
1. Data Transfer Instructions

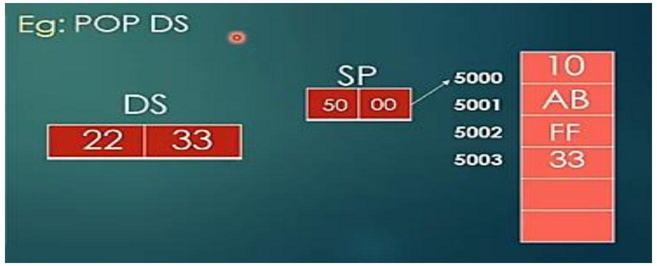
[Instruction Set]

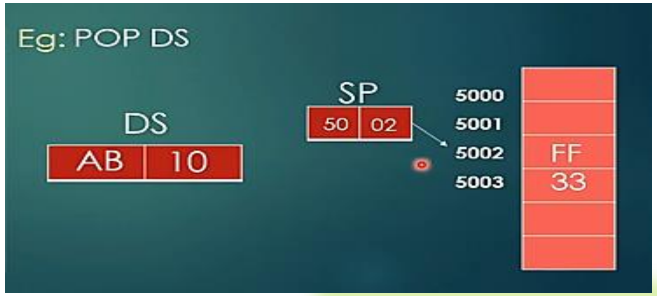
Mnemonics: MOV, XCHG, PUSH, POP, IN, OUT …

PUSH reg16/ mem	
PUSH reg16	$(SP) \leftarrow (SP) - 2$ $MA_S = (SS) \times 16_{10} + SP$ $(MA_S ; MA_S + 1) \leftarrow (reg16)$
PUSH mem	$(SP) \leftarrow (SP) - 2$ $MA_S = (SS) \times 16_{10} + SP$ $(MA_S ; MA_S + 1) \leftarrow (mem)$
POP reg16/ mem	
POP reg16	$MA_S = (SS) \times 16_{10} + SP$ $(reg16) \leftarrow (MA_S ; MA_S + 1)$ $(SP) \leftarrow (SP) + 2$
POP mem	$MA_S = (SS) \times 16_{10} + SP$ $(mem) \leftarrow (MA_S ; MA_S + 1)$ $(SP) \leftarrow (SP) + 2$


Eg: PUSH BX


Eg: PUSH BX


Eg: POP DS

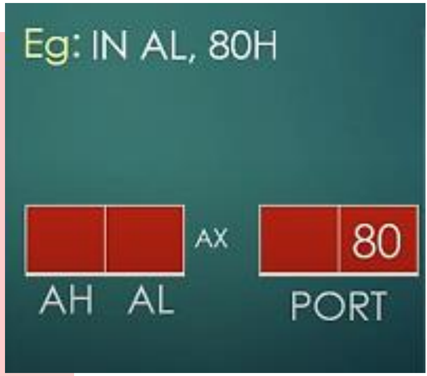

Eg: POP DS

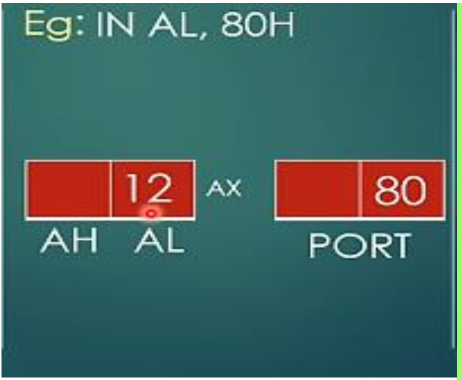
1. Data Transfer Instructions

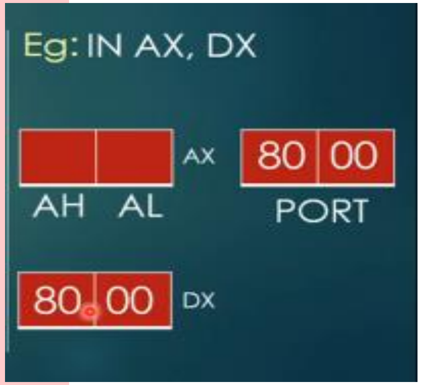
[Instruction Set]

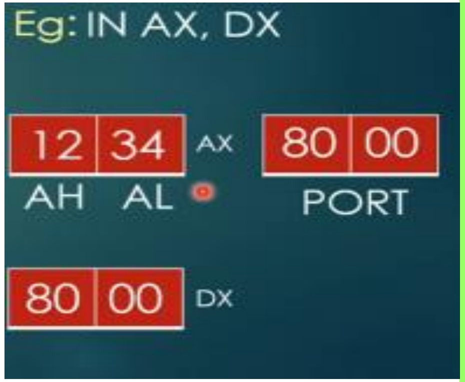
Mnemonics: **MOV, XCHG, PUSH, POP, IN, OUT …**

IN A, [DX]	
IN AL, [DX]	**PORT$_{addr}$ = (DX)** **(AL) ← (PORT)**
IN AX, [DX]	**PORT$_{addr}$ = (DX)** **(AX) ← (PORT)**
IN A, addr8	
IN AL, addr8	**(AL) ← (addr8)**
IN AX, addr8	**(AX) ← (addr8)**









OUT [DX], A	
OUT [DX], AL	**PORT$_{addr}$ = (DX)** **(PORT) ← (AL)**
OUT [DX], AX	**PORT$_{addr}$ = (DX)** **(PORT) ← (AX)**
OUT addr8, A	
OUT addr8, AL	**(addr8) ← (AL)**
OUT addr8, AX	**(addr8) ← (AX)**

2. Arithmetic Instructions

Mnemonics: **ADD, ADC, SUB, SBB, INC, DEC, MUL, DIV, IDIV, CMP…**

ADD reg2/ mem, reg1/mem ADD reg2, reg1 ADD reg2, mem ADD mem, reg1	 (reg2) ← (reg1) + (reg2) (reg2) ← (reg2) + (mem) (mem) ← (mem)+(reg1)
ADD reg/mem, data ADD reg, data ADD mem, data	 (reg) ← (reg)+data (mem) ← (mem)+data
ADD A, data ADD AL, data8 ADD AX, data16	 (AL) ← (AL) + data8 (AX) ← (AX) +data16

Example:

```
MOV AL, 5   ; AL = 5
ADD AL, -3  ; AL = 2
RET
```

2. Arithmetic Instructions

[Instruction Set]

Mnemonics: ADD, ADC, SUB, SBB, INC, DEC, MUL, DIV, IDIV, CMP…

ADC reg2/ mem, reg1/mem	
ADC reg2, reg1 ADC reg2, mem ADC mem, reg1	(reg2) ← (reg1) + (reg2)+CF (reg2) ← (reg2) + (mem)+CF (mem) ← (mem)+(reg1)+CF
ADC reg/mem, data	
ADC reg, data ADC mem, data	(reg) ← (reg)+ data+CF (mem) ← (mem)+data+CF
ADDC A, data	
ADD AL, data8 ADD AX, data16	(AL) ← (AL) + data8+CF (AX) ← (AX) +data16+CF

Example:

```
STC         ; set CF = 1
MOV AL, 5  ; AL = 5
ADC AL, 1  ; AL = 7
RET
```

2. Arithmetic Instructions

Mnemonics: **ADD, ADC, SUB, SBB, INC, DEC, MUL, DIV, IDIV, CMP…**

SUB reg2/ mem, reg1/mem	
SUB reg2, reg1 **SUB reg2, mem** **SUB mem, reg1**	**(reg2) ← (reg1) -(reg2)** **(reg2) ← (reg2) -(mem)** **(mem) ← (mem) -(reg1)**
SUB reg/mem, data	
SUB reg, data **SUB mem, data**	**(reg) ← (reg) -data** **(mem) ← (mem) -data**
SUB A, data	
SUB AL, data8 **SUB AX, data16**	**(AL) ← (AL) - data8** **(AX) ← (AX) - data16**

Example:

```
MOV AL, 5
SUB AL, 1        ; AL = 4
RET
```

2. Arithmetic Instructions

Mnemonics: **ADD, ADC, SUB, SBB, INC, DEC, MUL, DIV, IDIV, CMP…**

SBB reg2/ mem, reg1/mem **SBB reg2, reg1** **SBB reg2, mem** **SBB mem, reg1**	Subtract with Borrow. **(reg2) ← (reg1) - (reg2) -CF** **(reg2) ← (reg2) - (mem)-CF** **(mem) ← (mem) - (reg1)–CF**
SBB reg/mem, data **SBB reg, data** **SBB mem, data**	**(reg) ← (reg) – data -CF** **(mem) ← (mem) - data -CF**
SBB A, data **SBB AL, data8** **SBB AX, data16**	**(AL) ← (AL) - data8 - CF** **(AX) ← (AX) - data16 - CF**

Example:

```
STC
MOV AL, 5
SBB AL, 3    ; AL = 5 - 3 - 1 = 1
RET
```

2. Arithmetic Instructions

Mnemonics: **ADD, ADC, SUB, SBB, INC, DEC, MUL, DIV, IDIV, CMP…**

INC reg/ mem	
INC reg8	**(reg8) ← (reg8) + 1**
INC reg16	**(reg16) ← (reg16) + 1**
INC mem	**(mem) ← (mem) + 1**
DEC reg/ mem	
DEC reg8	**(reg8) ← (reg8) - 1**
DEC reg16	**(reg16) ← (reg16) - 1**
DEC mem	**(mem) ← (mem) - 1**

```
INC : Example
MOV AL, 4
INC AL       ; AL = 5
RET
```

```
DEC : Example
MOV AL, 255  ; AL = 0FFh
DEC AL       ; AL = 0FEh
RET
```

2. Arithmetic Instructions

Mnemonics: **ADD, ADC, SUB, SBB, INC, DEC, MUL, DIV, IDIV,CMP…**

MUL reg/ mem	
MUL reg	**For byte : (AX) ← (AL) x (reg8)** **For word : (DX)(AX) ← (AX) x (reg16)**
MUL mem	**For byte : (AX) ← (AL) x (mem8)** **For word : (DX)(AX) ← (AX) x (mem16)**
IMUL reg/ mem	
IMUL reg	**For byte : (AX) ← (AL) x (reg8)** **For word : (DX)(AX) ← (AX) x (reg16)**
IMUL mem	**For byte : (AX) ← (AX) x (mem8)** **For word : (DX)(AX) ← (AX) x (mem16)**

Example:

```
MOV AL, 200   ; AL = 0C8h
MOV BL, 4
MUL BL        ; AX = 0320h (800)
RET
```

2. Arithmetic Instructions

Mnemonics: **ADD, ADC, SUB, SBB, INC, DEC, MUL, DIV, IDIV, CMP…**

DIV reg/ mem	
DIV reg	**For 16-bit :- 8-bit :** **(AL) ← (AX) :- (reg8) Quotient** **(AH) ← (AX) MOD(reg8) Remainder** **For 32-bit :- 16-bit :** **(AX) ← (DX)(AX) :- (reg16) Quotient** **(DX) ← (DX)(AX) MOD(reg16) Remainder**
DIV mem	**For 16-bit :- 8-bit :** **(AL) ← (AX) :- (mem8) Quotient** **(AH) ← (AX) MOD(mem8) Remainder** **For 32-bit :- 16-bit :** **(AX) ← (DX)(AX) :- (mem16) Quotient** **(DX) ← (DX)(AX) MOD(mem16) Remainder**

Example:

```
MOV AX, 203   ; AX = 00CBh
MOV BL, 4
DIV BL        ; AL = 50 (32h), AH = 3
RET
```

2. Arithmetic Instructions

Mnemonics: ADD, ADC, SUB, SBB, INC, DEC, MUL, DIV, IDIV, CMP…

IDIV reg/ mem	
IDIV reg	<u>For 16-bit :- 8-bit</u> : (AL) ← (AX) :- (reg8) Quotient (AH) ← (AX) MOD(reg8) Remainder <u>For 32-bit :- 16-bit</u> : (AX) ← (DX)(AX) :- (reg16) Quotient (DX) ← (DX)(AX) MOD(reg16) Remainder
IDIV mem	<u>For 16-bit :- 8-bit</u> : (AL) ← (AX) :- (mem8) Quotient (AH) ← (AX) MOD(mem8) Remainder <u>For 32-bit :- 16-bit</u> : (AX) ← (DX)(AX) :- (mem16) Quotient (DX) ← (DX)(AX) MOD(mem16) Remainder

```
MOV AX, -203 ; AX = 0FF35h
MOV BL, 4
IDIV BL      ; AL = -50 (0CEh), AH = -3 (0FDh)
RET
```

2. Arithmetic Instructions

Mnemonics: **ADD, ADC, SUB, SBB, INC, DEC, MUL, DIV, CMP…**

CMP reg2/mem, reg1/ mem	
CMP reg2, reg1	Modify flags ← (reg2) – (reg1) If (reg2) > (reg1) then CF=0, ZF=0, SF=0 If (reg2) < (reg1) then CF=1, ZF=0, SF=1 If (reg2) = (reg1) then CF=0, ZF=1, SF=0
CMP reg2, mem	Modify flags ← (reg2) – (mem) If (reg2) > (mem) then CF=0, ZF=0, SF=0 If (reg2) < (mem) then CF=1, ZF=0, SF=1 If (reg2) = (mem) then CF=0, ZF=1, SF=0
CMP mem, reg1	Modify flags ← (mem) – (reg1) If (mem) > (reg1) then CF=0, ZF=0, SF=0 If (mem) < (reg1) then CF=1, ZF=0, SF=1 If (mem) = (reg1) then CF=0, ZF=1, SF=0

2. Arithmetic Instructions

Mnemonics: **ADD, ADC, SUB, SBB, INC, DEC, MUL, DIV, CMP…**

CMP reg/mem, data	
CMP reg, data	**Modify flags ← (reg) – (data)** **If (reg) > data then CF=0, ZF=0, SF=0** **If (reg) < data then CF=1, ZF=0, SF=1** **If (reg) = data then CF=0, ZF=1, SF=0**
CMP mem, data	**Modify flags ← (mem) – (mem)** **If (mem) > data then CF=0, ZF=0, SF=0** **If (mem) < data then CF=1, ZF=0, SF=1** **If (mem) = data then CF=0, ZF=1, SF=0**

Example:

```
MOV AL, 5
MOV BL, 5
CMP AL, BL  ; AL = 5, ZF = 1 (so equal!)
RET
```

2. Arithmetic Instructions

Mnemonics: **ADD, ADC, SUB, SBB, INC, DEC, MUL, DIV, CMP…**

CMP A, data	
CMP AL, data8	**Modify flags ← (AL) −data8** **If (AL) > data8 then CF=0, ZF=0, SF=0** **If (AL) < data8 then CF=1, ZF=0, SF=1** **If (AL) = data8 then CF=0, ZF=1, SF=0**
CMP AX, data16	**Modify flags ← (AX) −data16** **If (AX) > data16 then CF=0, ZF=0, SF=0** **If (mem) < data16 then CF=1, ZF=0, SF=1** **If (mem) = data16 then CF=0, ZF=1, SF=0**

Thank you so much

Any questions ?

Lecturer :Abdullah Thair Abdalsatir
Department of computer Engineering