CMPE 310

Lecture 05 - 8086 Addressing modes and Instructions set

Outline

Addressing modes

Instruction format

Instruction set

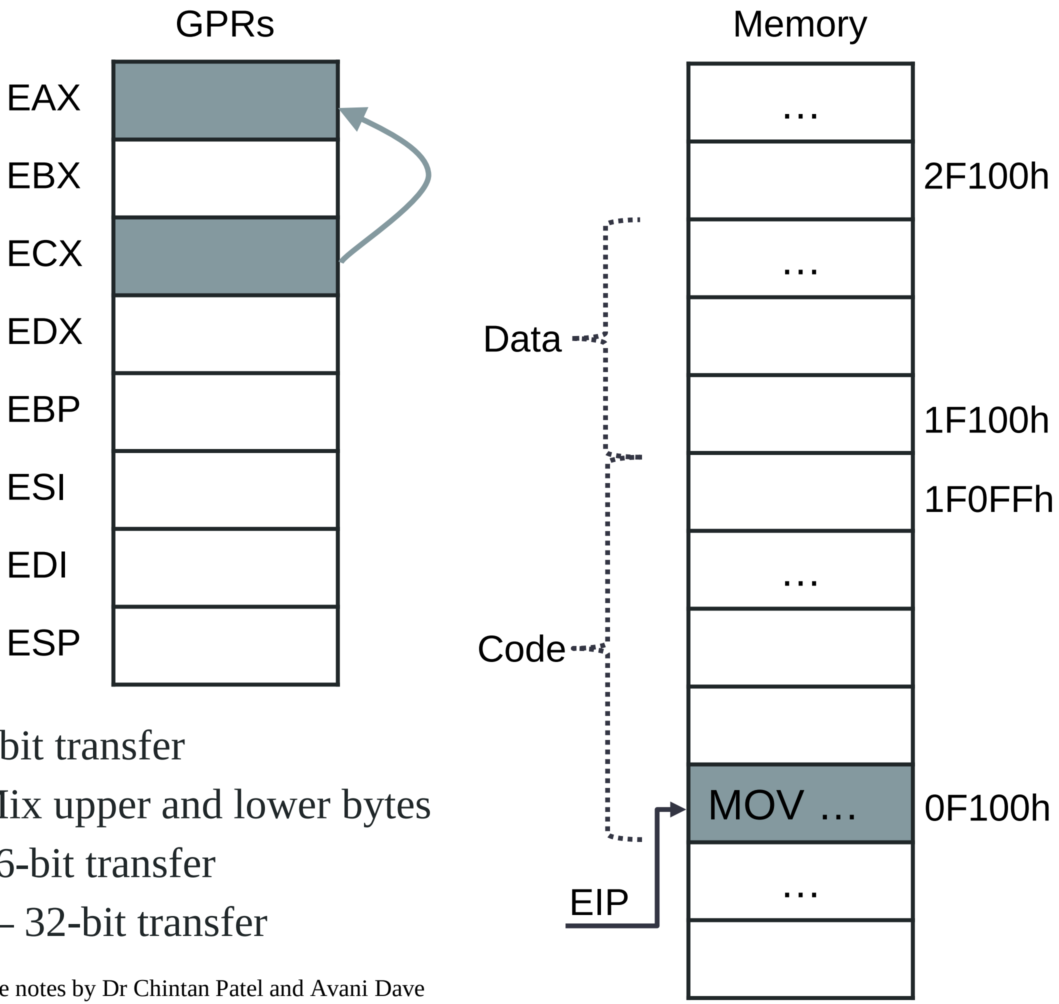
Addressing Modes

1.Register Addressing	Register and immediate data
2.Immediate Addressing	
3.Direct Addressing	Memory data
4.Register Indirect Addressing	
5.Based Addressing	
6.Indexed Addressing	
7.Based Index Addressing	
8.String Addressing	
9.Direct I/O port Addressing	I/O ports
10.Indirect I/O port Addressing	
11.Relative Addressing	
12.Implied Addressing	

Adapted from lecture notes by Dr Chintan Patel and Avani Dave

Addressing Modes

1.Register Addressing
2.Immediate Addressing
3.Direct Addressing
4.Register Indirect Addressing
5.Based Addressing
6.Indexed Addressing
7.Based Index Addressing
8.String Addressing
9.Direct I/O port Addressing
10.Indirect I/O port Addressing
11.Relative Addressing
12.Implied Addressing



Examples

MOV AL,CL - 8 bit transfer

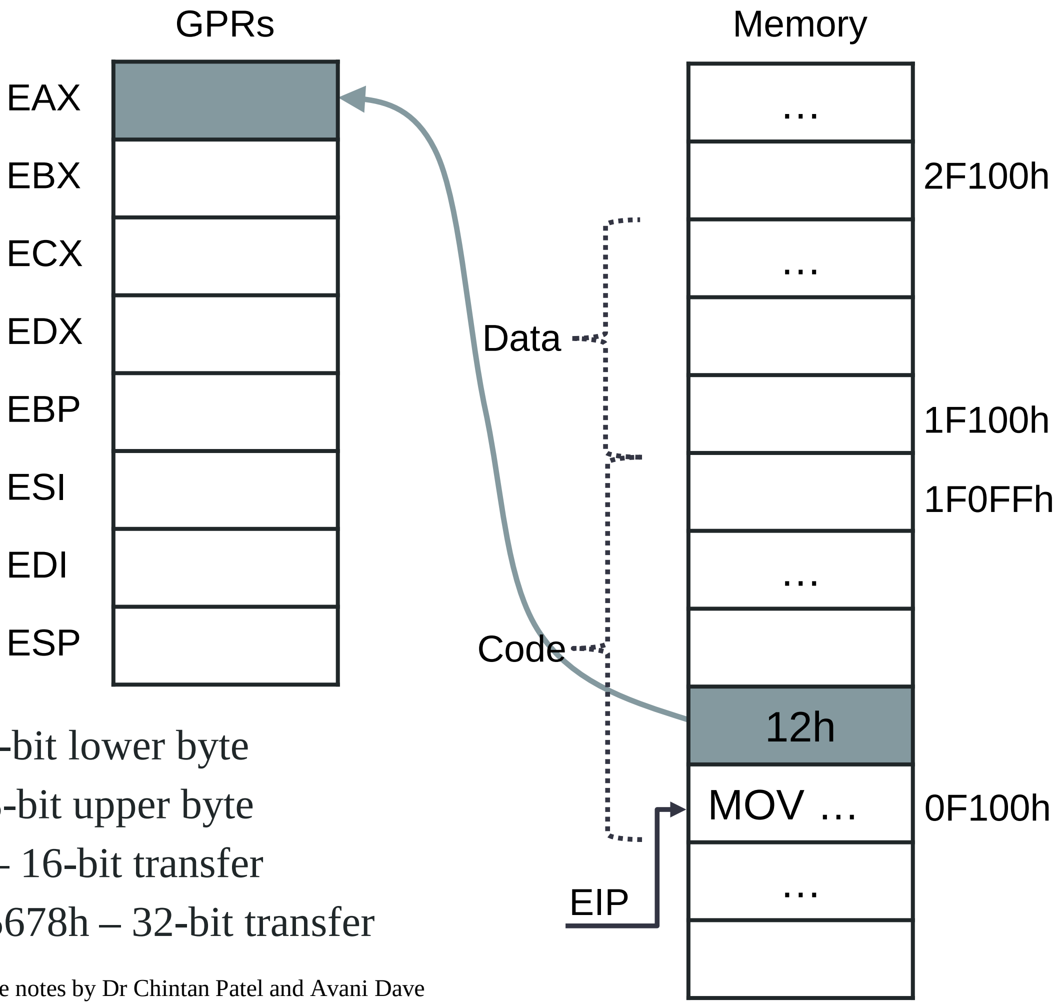
MOV AH,CL - Mix upper and lower bytes

MOV AX, CX – 16-bit transfer

MOV EAX, ECX – 32-bit transfer

Addressing Modes

1.Register Addressing
2.Immediate Addressing
3.Direct Addressing
4.Register Indirect Addressing
5.Based Addressing
6.Indexed Addressing
7.Based Index Addressing
8.String Addressing
9.Direct I/O port Addressing
10.Indirect I/O port Addressing
11.Relative Addressing
12.Implied Addressing



Examples

MOV AL,12h - 8-bit lower byte

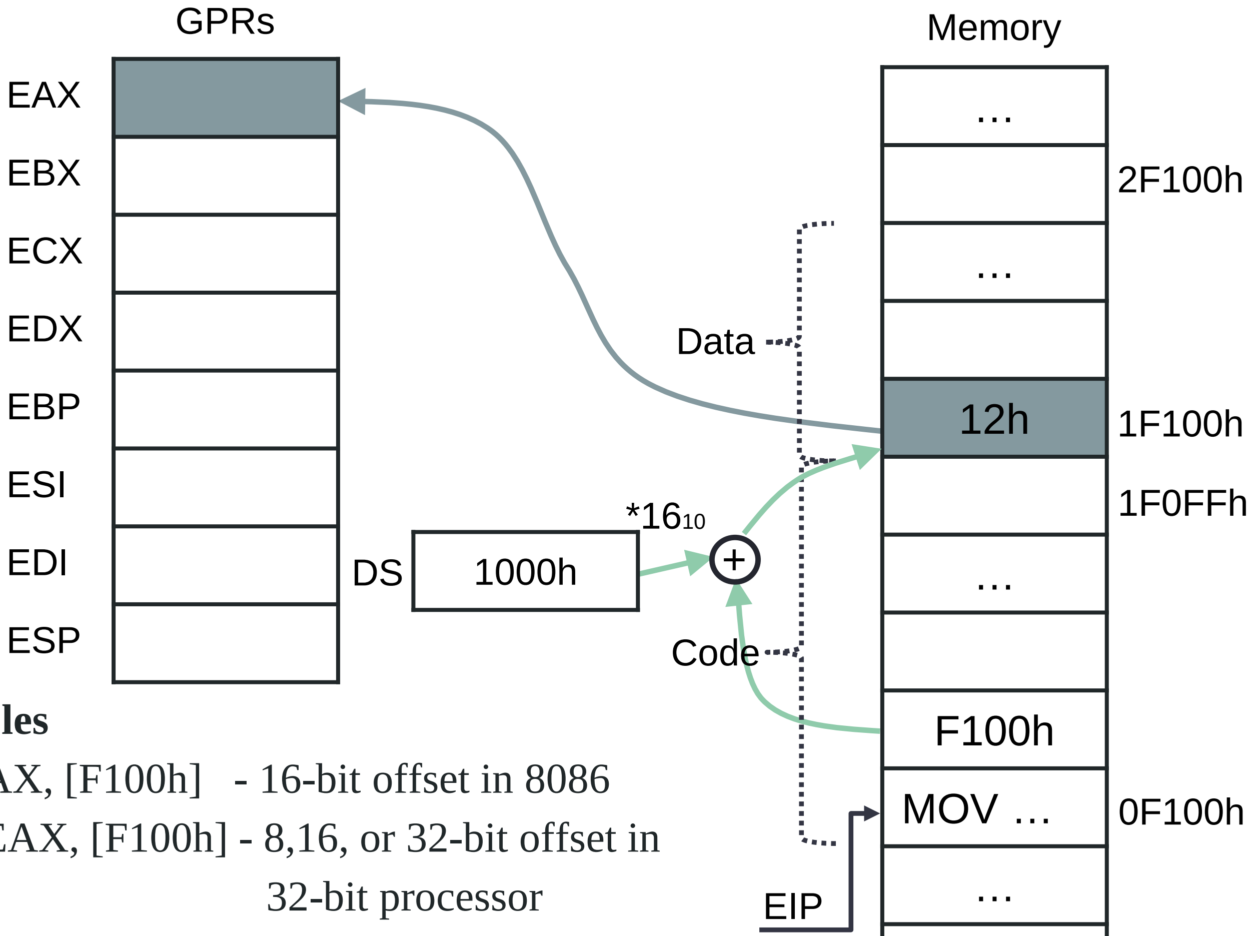
MOV AH,12h - 8-bit upper byte

MOV AX, 1234h – 16-bit transfer

MOV EAX, 12345678h – 32-bit transfer

Addressing Modes

1.Register Addressing
2.Immediate Addressing
3.Direct Addressing
4.Register Indirect Addressing
5.Based Addressing
6.Indexed Addressing
7.Based Index Addressing
8.String Addressing
9.Direct I/O port Addressing
10.Indirect I/O port Addressing
11.Relative Addressing
12.Implied Addressing



Examples

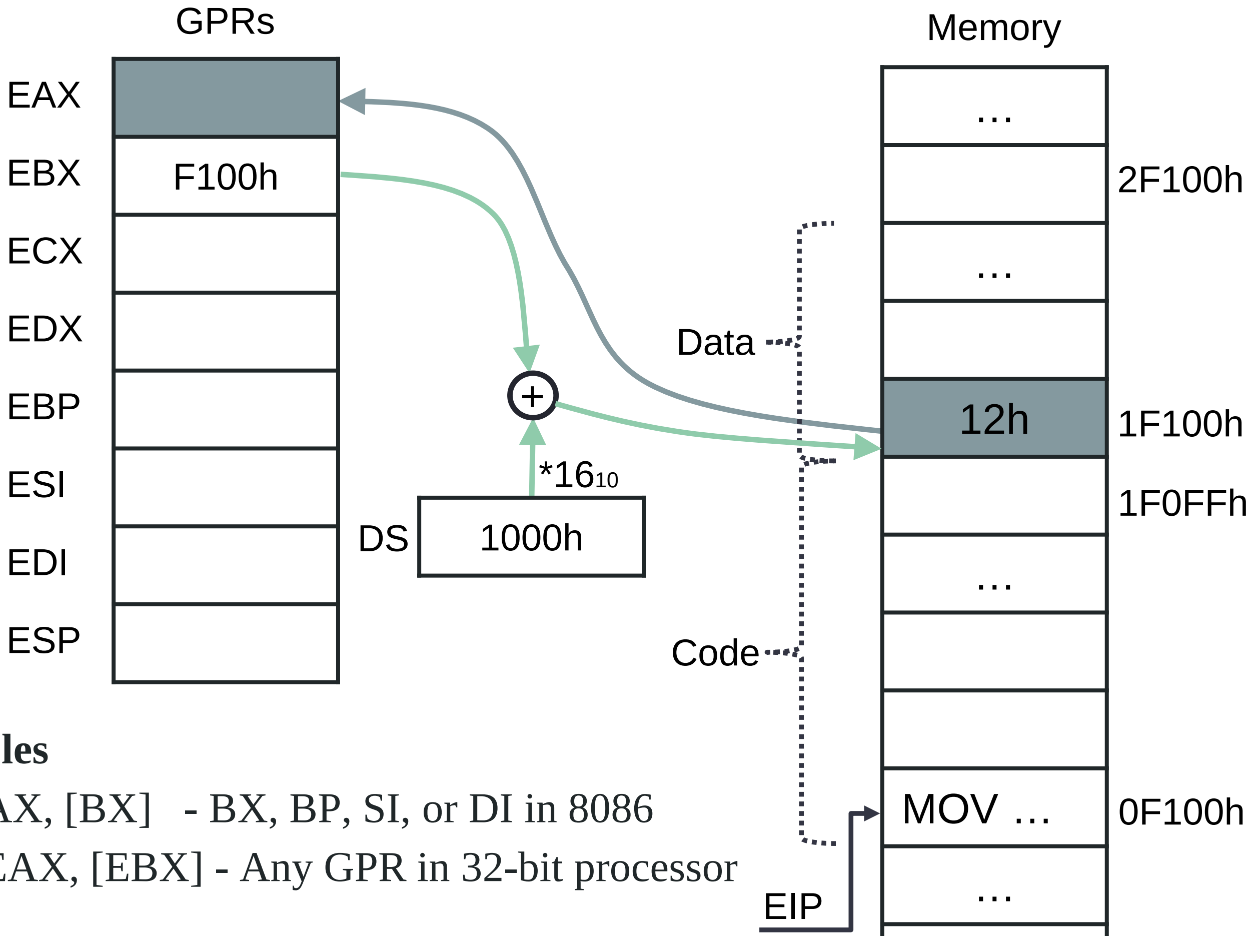
MOV AX, [F100h] - 16-bit offset in 8086

MOV EAX, [F100h] - 8,16, or 32-bit offset in 32-bit processor

Adapted from lecture notes by Dr Chintan Patel and Avani Dave

Addressing Modes

1.Register Addressing
2.Immediate Addressing
3.Direct Addressing
4.Register Indirect Addressing
5.Based Addressing
6.Indexed Addressing
7.Based Index Addressing
8.String Addressing
9.Direct I/O port Addressing
10.Indirect I/O port Addressing
11.Relative Addressing
12.Implied Addressing



Examples

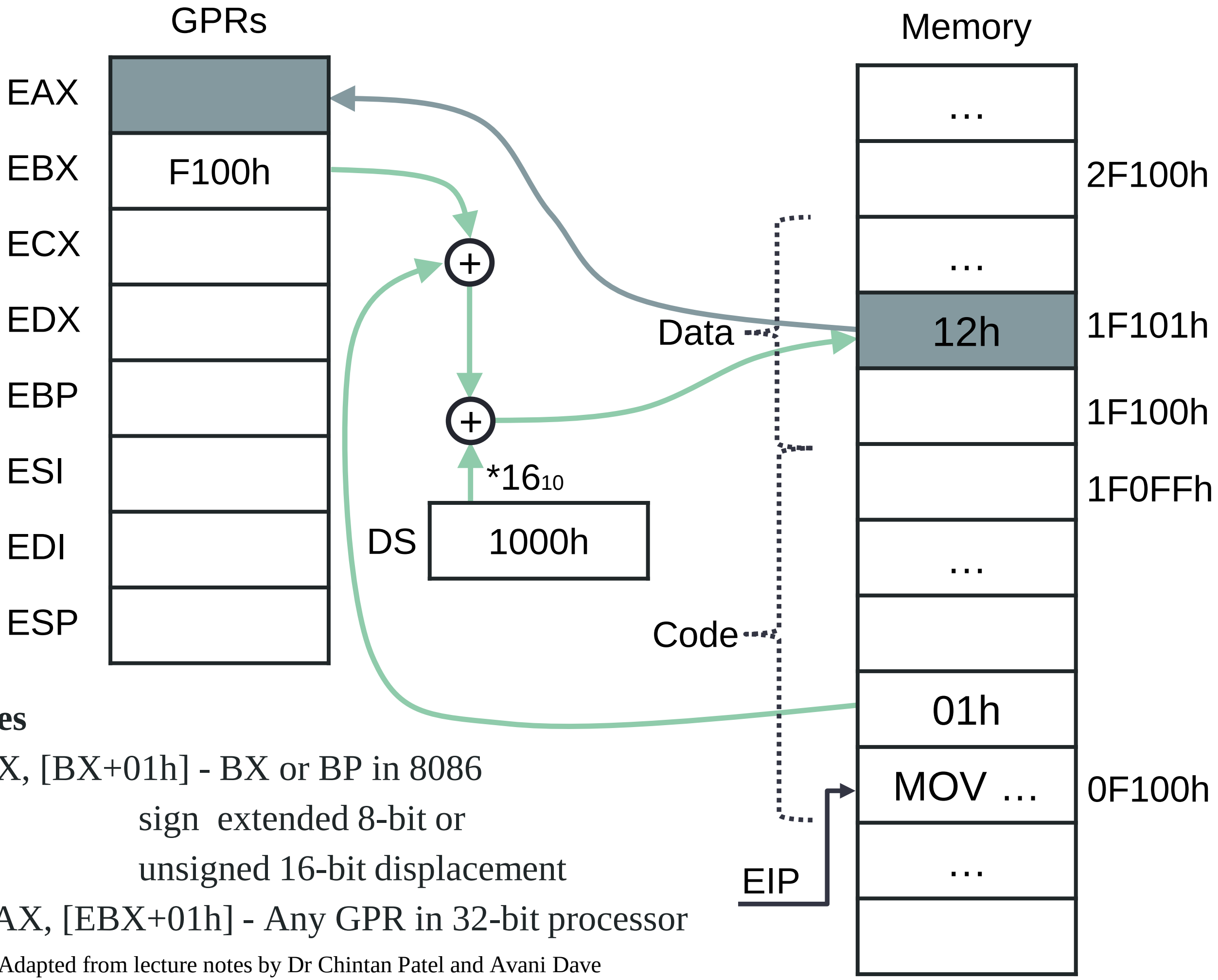
MOV AX, [BX] - BX, BP, SI, or DI in 8086

MOV EAX, [EBX] - Any GPR in 32-bit processor

Adapted from lecture notes by Dr Chintan Patel and Avani Dave

Addressing Modes

1.Register Addressing
2.Immediate Addressing
3.Direct Addressing
4.Register Indirect Addressing
5.Based Addressing
6.Indexed Addressing
7.Based Index Addressing
8.String Addressing
9.Direct I/O port Addressing
10.Indirect I/O port Addressing
11.Relative Addressing
12.Implied Addressing



Examples

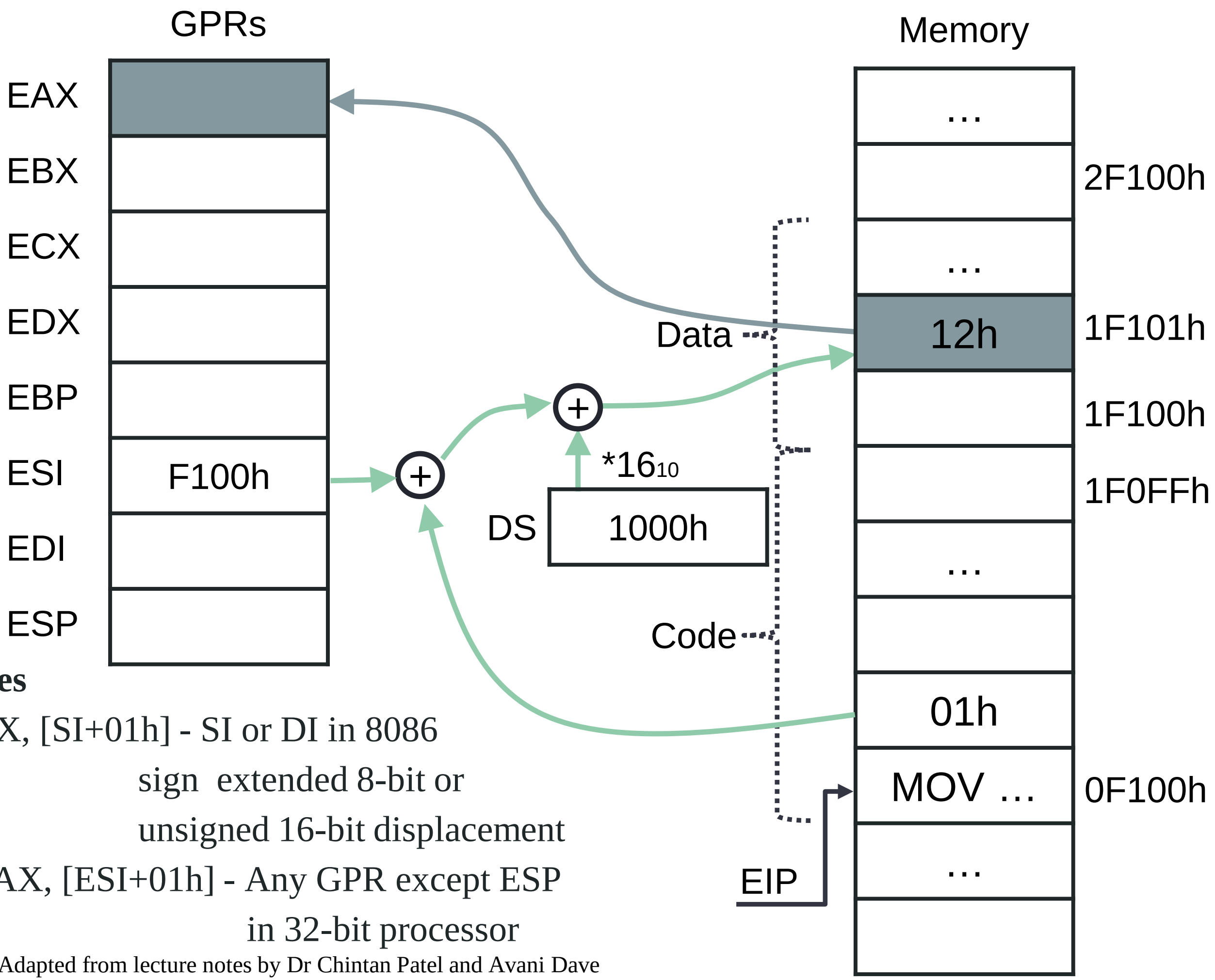
MOV AX, [BX+01h] - BX or BP in 8086
sign extended 8-bit or
unsigned 16-bit displacement

MOV EAX, [EBX+01h] - Any GPR in 32-bit processor

Adapted from lecture notes by Dr Chintan Patel and Avani Dave

Addressing Modes

1.Register Addressing
2.Immediate Addressing
3.Direct Addressing
4.Register Indirect Addressing
5.Based Addressing
6.Indexed Addressing
7.Based Index Addressing
8.String Addressing
9.Direct I/O port Addressing
10.Indirect I/O port Addressing
11.Relative Addressing
12.Implied Addressing



Examples

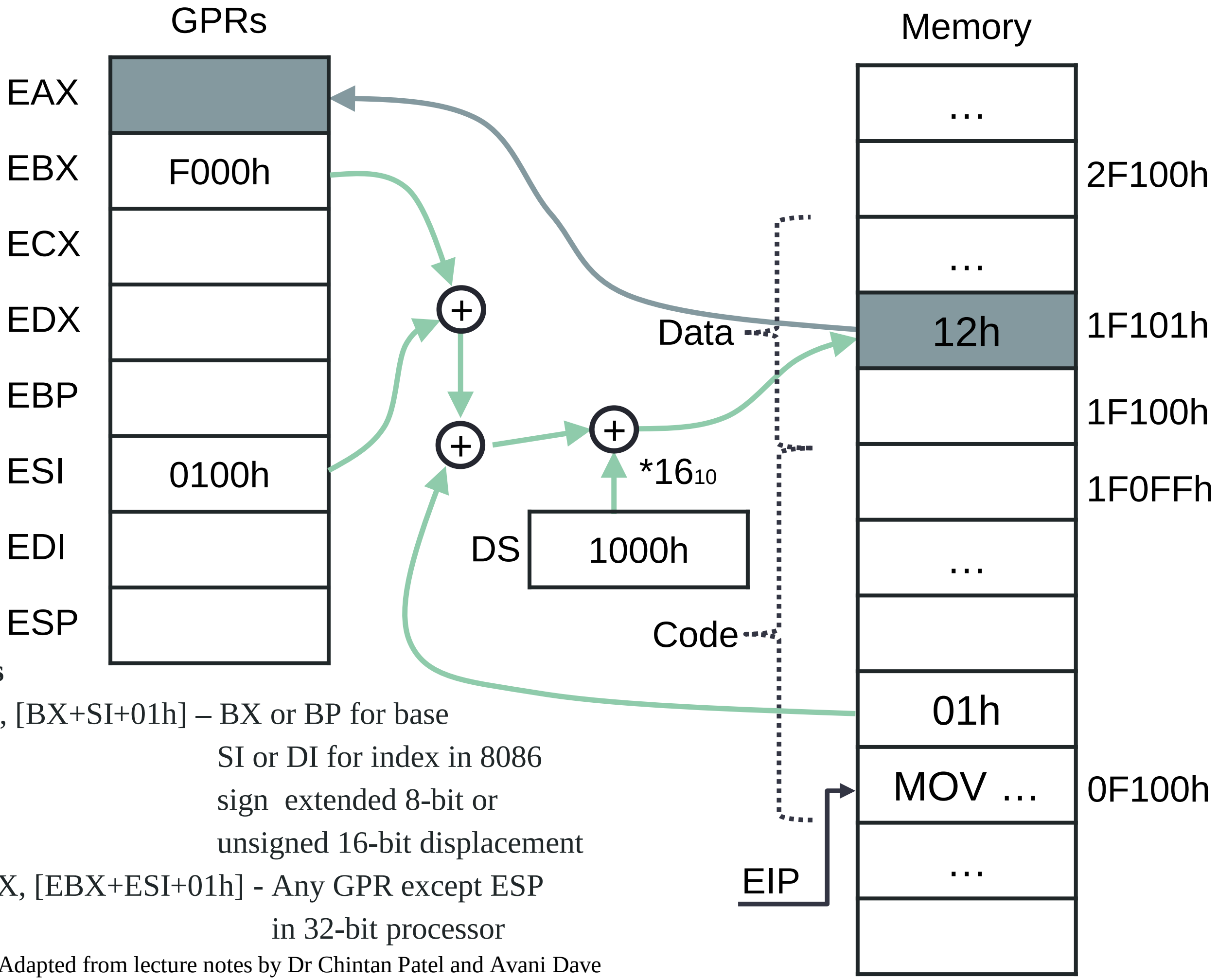
MOV AX, [SI+01h] - SI or DI in 8086
sign extended 8-bit or
unsigned 16-bit displacement

MOV EAX, [ESI+01h] - Any GPR except ESP
in 32-bit processor

Adapted from lecture notes by Dr Chintan Patel and Avani Dave

Addressing Modes

1.Register Addressing
2.Immediate Addressing
3.Direct Addressing
4.Register Indirect Addressing
5.Based Addressing
6.Indexed Addressing
7.Based Index Addressing
8.String Addressing
9.Direct I/O port Addressing
10.Indirect I/O port Addressing
11.Relative Addressing
12.Implied Addressing



Examples

MOV AX, [BX+SI+01h] – BX or BP for base
SI or DI for index in 8086
sign extended 8-bit or
unsigned 16-bit displacement

MOV EAX, [EBX+ESI+01h] - Any GPR except ESP
in 32-bit processor

Adapted from lecture notes by Dr Chintan Patel and Avani Dave

Employed in string operations to operate on string data.
The effective address (EA) of source data is stored in **SI** register and the EA of destination is stored in **DI** register**.**
Segment register for calculating base address of source data is **DS** and that of the destination data is **ES**

Example:

MOVSB ; DS ← 1000h, SI ← 5000h
; ES ← 7000h, DI ← 6000h

MOVSW, MOVSD for word or doubleword respectively

Operations:

BA - Base Address, MA – Memory Address (Physical Address)

Calculation of source memory location:

EA = SI BA = DS * 16_{10} MA = BA + EA

Calculation of destination memory location:

EA = DI BA = ES * 16_{10} MA = BA + EA

If DF = 1, then SI – 1 and DI = DI - 1
If DF = 0, then SI +1 and DI = DI + 1

1.Register Addressing
2.Immediate Addressing
3.Direct Addressing
4.Register Indirect Addressing
5.Based Addressing
6.Indexed Addressing
7.Based Index Addressing
8.String Addressing
9.Direct I/O port Addressing
10.Indirect I/O port Addressing
11.Relative Addressing
12.Implied Addressing

These addressing modes are used to access data from standard I/O mapped devices or ports.

In direct port addressing mode, an 8-bit port address is directly specified in the instruction.

Example: IN AL, [09h]

Operations: Content of port with address 09h is moved to AL register.

1.Register Addressing
2.Immediate Addressing
3.Direct Addressing
4.Register Indirect Addressing
5.Based Addressing
6.Indexed Addressing
7.Based Index Addressing
8.String Addressing
9.Direct I/O port Addressing
10.Indirect I/O port Addressing
11.Relative Addressing
12.Implied Addressing

In indirect port addressing mode, a 16-bit port address is specified in a register.

Example: IN AL, [DX]

Operations: Content of port with address in register DX is moved to AL register.

Can be a byte or word transfer in 8086 (e.g. AL or AX).

In 32-bit systems, can be byte, word or doubleword transfer (e.g. AL, AX, or EAX).

In this addressing mode, the effective address of a program instruction is specified relative to Instruction Pointer (IP) by an 8-bit signed displacement.

Example: JZ 0Ah ; CS ← 7000H, IP ← 6500H

Operations:

$000Ah \longleftarrow 0Ah$ (sign extend)

If ZF = 1, then

$EA = IP + 000A_h$

$BA = CS * 16_{10}$

$MA = BA + EA$

If ZF = 1, then the program control jumps to new address calculated above.

If ZF = 0, then next instruction of the program is executed.

1.Register Addressing
2.Immediate Addressing
3.Direct Addressing
4.Register Indirect Addressing
5.Based Addressing
6.Indexed Addressing
7.Based Index Addressing
8.String Addressing
9.Direct I/O port Addressing
10. Indirect I/O port Addressing
11.Relative Addressing
12.Implied Addressing

Instructions using this mode have **no operands.** The instruction itself will specify the data to be operated by the instruction.

Example: CLC - Clears the carry flag to zero.
CLD - Clears the Direction Flag (DF)
STD - Sets the Direction Flag (DF)

Instruction Format

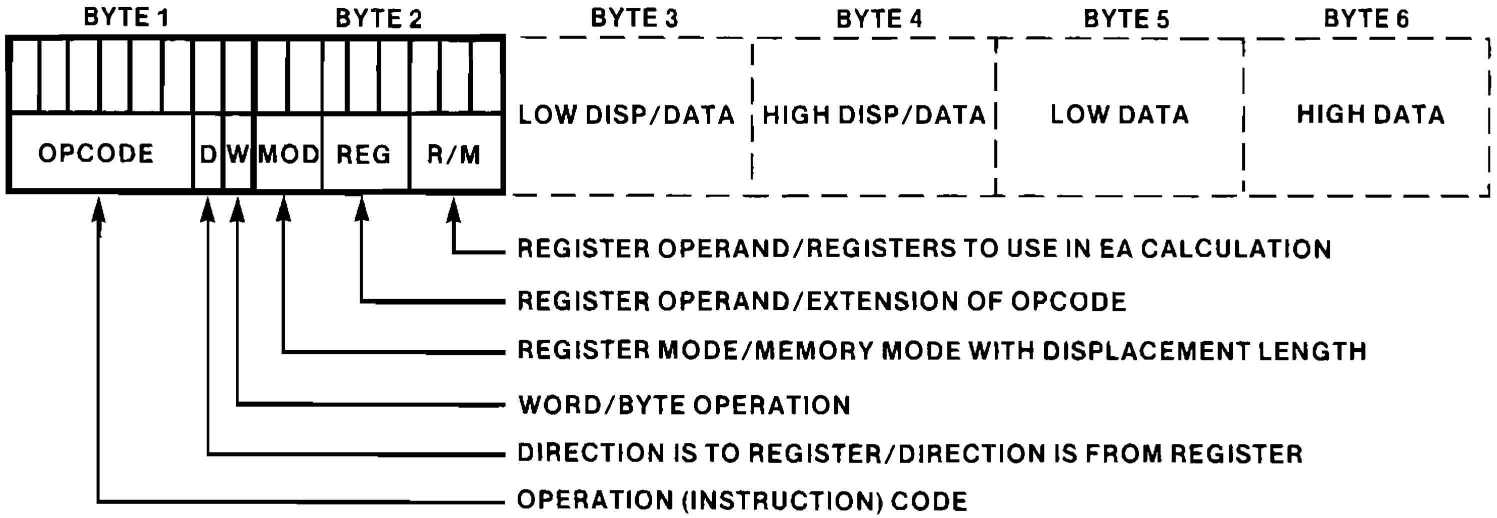
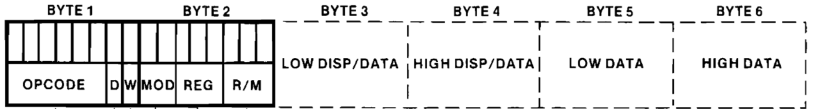
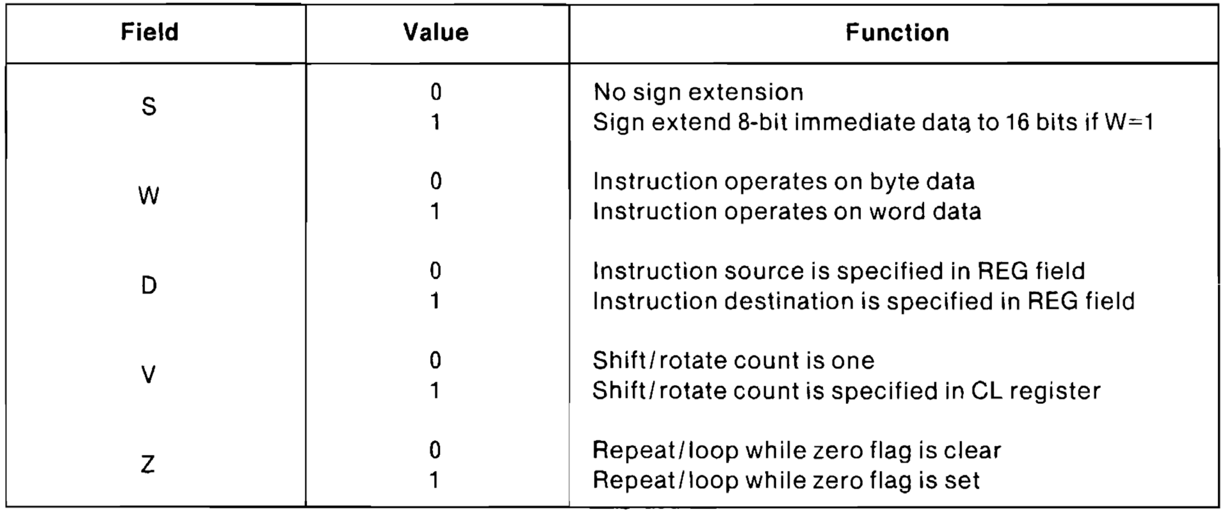

BYTE 1
BYTE 2
BYTE 3
BYTE 4
BYTE 5
BYTE 6
OPCODE
D
W
MOD
REG
R/M
LOW DISP/DATA
HIGH DISP/DATA
LOW DATA
HIGH DATA
REGISTER OPERAND/REGISTERS TO USE IN EA CALCULATION
REGISTER OPERAND/EXTENSION OF OPCODE
REGISTER MODE/MEMORY MODE WITH DISPLACEMENT LENGTH
WORD/BYTE OPERATION
DIRECTION IS TO REGISTER/DIRECTION IS FROM REGISTER
OPERATION (INSTRUCTION) CODE

Image Source: 8086 User Manual, Page 4-19

Instruction Format



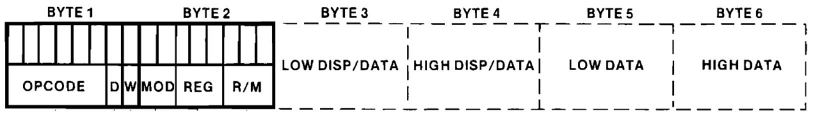


Field	Value	Function
S	0 1	No sign extension Sign extend 8-bit immediate data to 16 bits if W=1
W	0 1	Instruction operates on byte data Instruction operates on word data
D	0 1	Instruction source is specified in REG field Instruction destination is specified in REG field
V	0 1	Shift/rotate count is one Shift/rotate count is specified in CL register
Z	0 1	Repeat/loop while zero flag is clear Repeat/loop while zero flag is set

Image Source: 8086 User Manual, Page 4-19

Adapted from lecture notes by Dr Chintan Patel and Avani Dave

Instruction Format



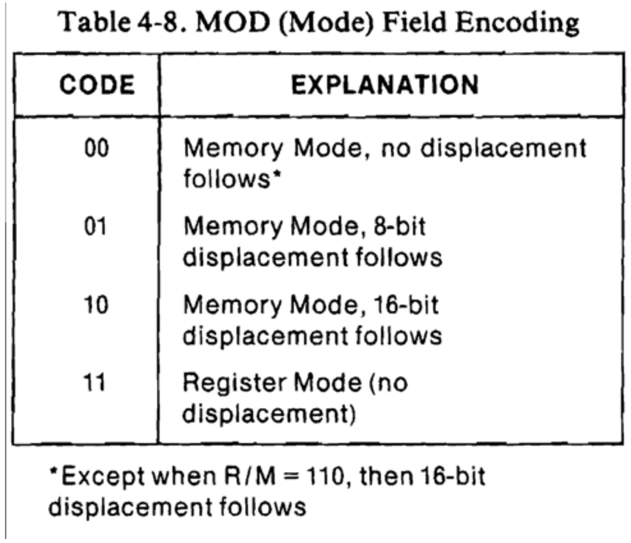


Table 4-8. MOD (Mode) Field Encoding

CODE	EXPLANATION
00	Memory Mode, no displacement follows*
01	Memory Mode, 8-bit displacement follows
10	Memory Mode, 16-bit displacement follows
11	Register Mode (no displacement)

*Except when R/M = 110, then 16-bit displacement follows

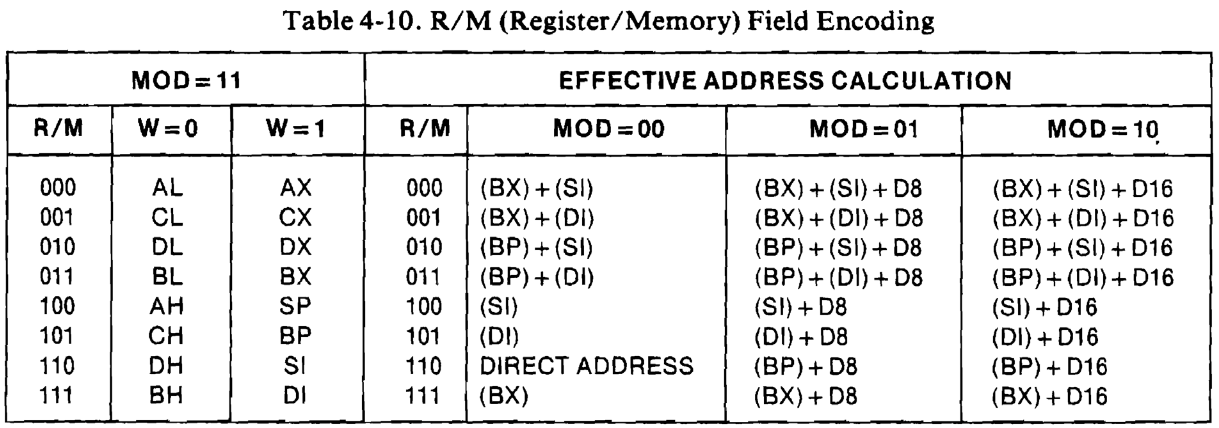


Table 4-10. R/M (Register/Memory) Field Encoding

MOD = 11			EFFECTIVE ADDRESS CALCULATION			
R/M	W = 0	W = 1	R/M	MOD = 00	MOD = 01	MOD = 10
000	AL	AX	000	(BX) + (SI)	(BX) + (SI) + D8	(BX) + (SI) + D16
001	CL	CX	001	(BX) + (DI)	(BX) + (DI) + D8	(BX) + (DI) + D16
010	DL	DX	010	(BP) + (SI)	(BP) + (SI) + D8	(BP) + (SI) + D16
011	BL	BX	011	(BP) + (DI)	(BP) + (DI) + D8	(BP) + (DI) + D16
100	AH	SP	100	(SI)	(SI) + D8	(SI) + D16
101	CH	BP	101	(DI)	(DI) + D8	(DI) + D16
110	DH	SI	110	DIRECT ADDRESS	(BP) + D8	(BP) + D16
111	BH	DI	111	(BX)	(BX) + D8	(BX) + D16

Image Source: 8086 User Manual, Page 4-19, 4-20

Adapted from lecture notes by Dr Chintan Patel and Avani Dave

Instruction Format

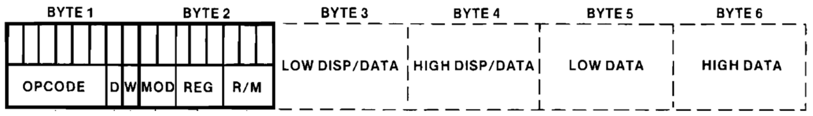
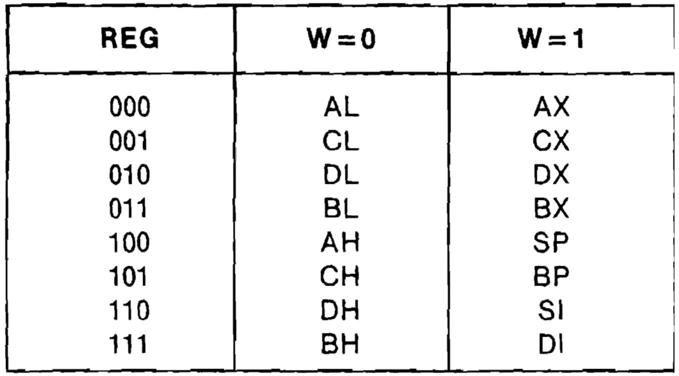
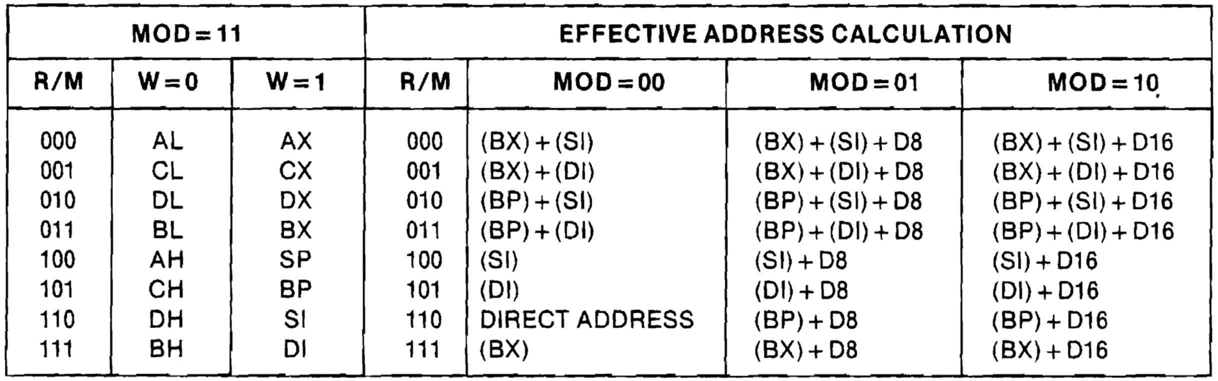


Table 4-9. REG (Register) Field Encoding



REG	W=0	W=1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

Table 4-10. R/M (Register/Memory) Field Encoding



MOD=11			EFFECTIVE ADDRESS CALCULATION			
R/M	W=0	W=1	R/M	MOD=00	MOD=01	MOD=10
000	AL	AX	000	(BX) + (SI)	(BX) + (SI) + D8	(BX) + (SI) + D16
001	CL	CX	001	(BX) + (DI)	(BX) + (DI) + D8	(BX) + (DI) + D16
010	DL	DX	010	(BP) + (SI)	(BP) + (SI) + D8	(BP) + (SI) + D16
011	BL	BX	011	(BP) + (DI)	(BP) + (DI) + D8	(BP) + (DI) + D16
100	AH	SP	100	(SI)	(SI) + D8	(SI) + D16
101	CH	BP	101	(DI)	(DI) + D8	(DI) + D16
110	DH	SI	110	DIRECT ADDRESS	(BP) + D8	(BP) + D16
111	BH	DI	111	(BX)	(BX) + D8	(BX) + D16

Image Source: 8086 User Manual, Page 4-19, 4-20

Adapted from lecture notes by Dr Chintan Patel and Avani Dave

Instruction Set

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

Instruction Set

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

Instruction Set

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

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

MOV reg2/ mem, reg1/ mem MOV reg2, reg1 MOV mem, reg1 MOV reg2, mem	(reg2) $\longleftarrow$ (reg1) (mem) $\longleftarrow$ (reg1) (reg2) $\longleftarrow$ (mem)
MOV reg/ mem, imm MOV reg, imm MOV mem, imm	(reg) $\longleftarrow$ imm (mem) $\longleftarrow$ imm
XCHG reg2/ mem, reg1 XCHG reg2, reg1 XCHG mem, reg1	(reg2) $\longleftrightarrow$ (reg1) (mem) $\longleftrightarrow$ (reg1)

No memory to memory data transfer

Instruction Set

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

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

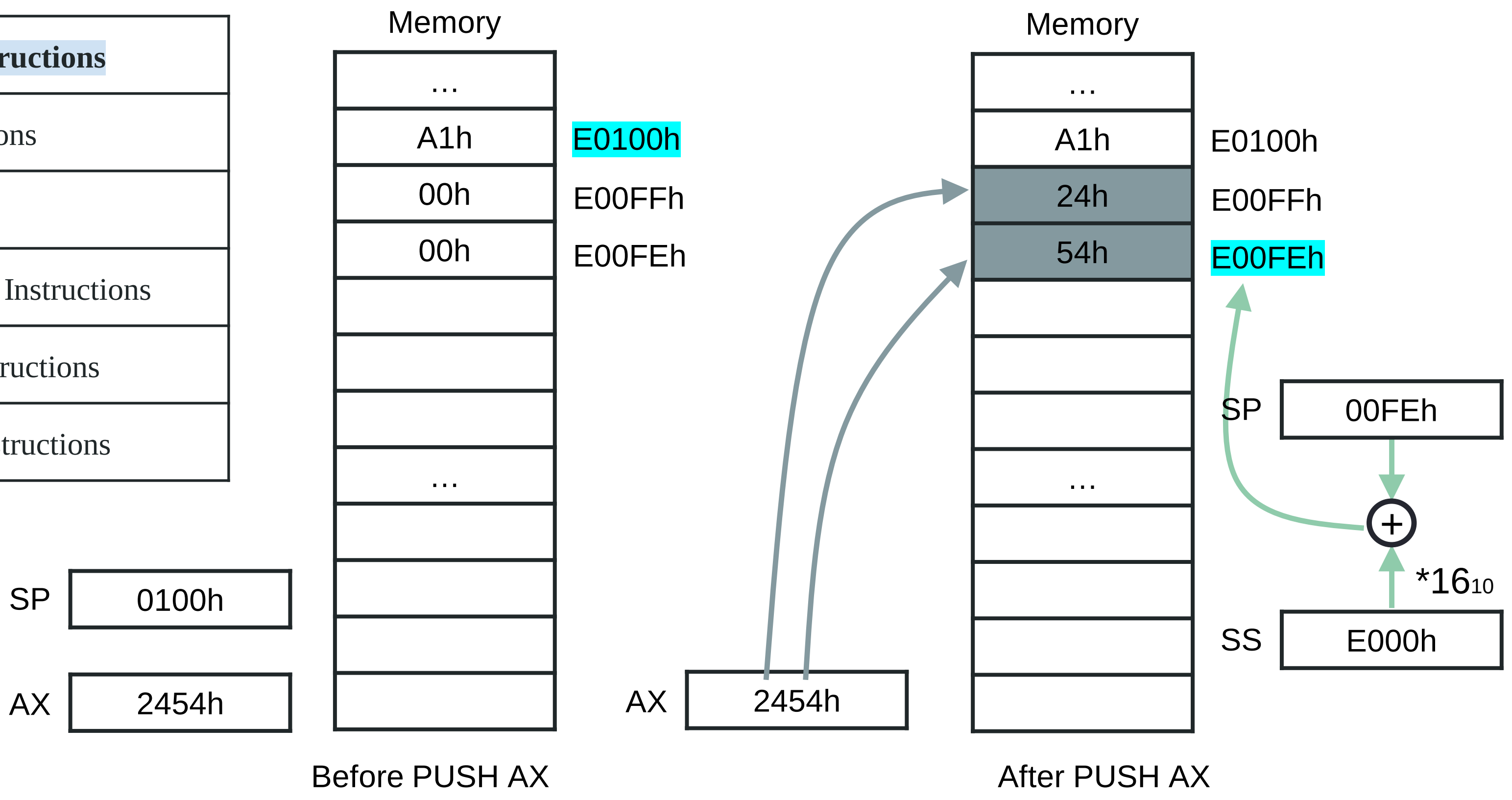
PUSH reg16/ mem PUSH reg16 PUSH mem	SP ← SP – 2 MA_S = SS * 16_{10} + SP [MA_S, MA_S+1] ← reg16 [MA_S, MA_S+1] ← mem

POP reg16/ mem
 	
POP reg16
POP mem
	MA_S = SS * 16_{10} + SP
	reg16 ← [MA_S, MA_S+1]
	
	[mem] ← [MA_S, MA_S+1]
	SP ← SP + 2

Instruction Set

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

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

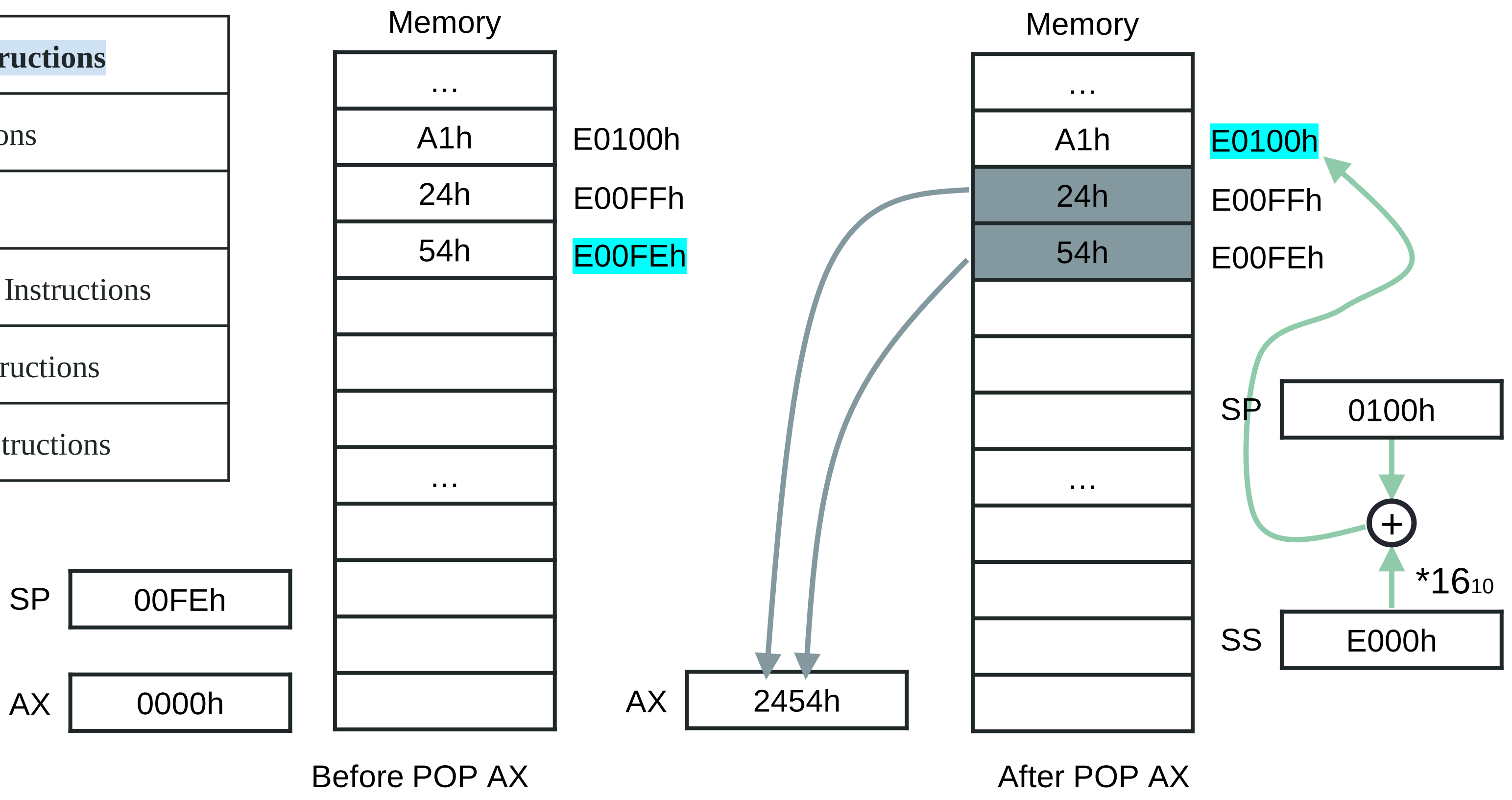


Adapted from lecture notes by Dr Chintan Patel and Avani Dave

Instruction Set

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

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



Adapted from lecture notes by Dr Chintan Patel and Avani Dave

Instruction Set

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

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

IN reg, [DX] IN AL, [DX] IN AX, [DX]	$PORT_{addr}$ = DX AL $\leftarrow$ [PORT] AX $\leftarrow$ [PORT]	**OUT [DX], reg** OUT [DX], AL OUT [DX], AX	$PORT_{addr}$ = DX [PORT] $\leftarrow$ AL [PORT] $\leftarrow$ AX
IN reg, addr8 IN AL, [addr8] IN AX, [addr8]	AL $\leftarrow$ [addr8] AX $\leftarrow$ [addr8]	**OUT addr8, reg** OUT [addr8], AL OUT [addr8], AX	[addr8] $\leftarrow$ AL [addr8] $\leftarrow$ AX

Adapted from lecture notes by Dr Chintan Patel and Avani Dave

Instruction Set

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

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

ADD reg2/ mem, reg1/mem
 	
ADD reg2, reg1
ADD reg2, mem
ADD mem, reg1
	
	reg2 ← reg2 + reg1
	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

Instruction Set

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

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

ADC reg2/ mem, reg1/mem
 	
ADC reg2, reg1
ADC reg2, mem
ADC mem, reg1
	
	reg2 ← reg2 + reg1 + 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
 	
ADC AL, data8
ADC AX, data16
	
	AL ← AL + data8+CF
	AX ← AX +data16+CF

Instruction Set

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

SUB reg2/ mem, reg1/mem	

SUB reg2, reg1
SUB reg2, mem
SUB mem, reg1
	reg2 ← reg2 - reg1
	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

Instruction Set

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

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

SBB reg2/ mem, reg1/mem	

SBB reg2, reg1
SBB reg2, mem
SBB mem, reg1
	reg2 ← reg2 – reg1 - 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

Instruction Set

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

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

INC reg/ mem
 	
INC reg8
INC reg16
INC mem
	
	reg8 ← reg8 + 1
	reg16 ← reg16 + 1
	[mem] ← [mem] + 1

DEC reg/ mem
 	
DEC reg8
DEC reg16
DEC mem
	
	reg8 ← reg8 - 1
	reg16 ← reg16 - 1
	[mem] ← [mem] - 1

Instruction Set

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

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

MUL reg/ mem	
MUL reg	For byte : AX ← AL * reg8 For word : DX:AX ← AX * reg16
MUL mem	For byte : AX ← AL * [mem8] For word : DX:AX ← AX * [mem16]
IMUL reg/ mem	Signed multiplication (Sign-extended)
IMUL reg	For byte : AX ← AL * reg8 For word : DX:AX ← AX * reg16
IMUL mem	For byte : AX ← AX * [mem8] For word : DX:AX ← AX * [mem16]

Adapted from lecture notes by Dr Chintan Patel and Avani Dave

Instruction Set

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

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

DIV reg/ mem
DIV reg
	For 16-bit / 8-bit :
	AL ← AX / reg8 Quotient
	AH ← AX % reg8 Remainder
	For 32-bit / 16-bit :
	AX ← DX:AX / reg16 Quotient
	DX ← DX:AX % reg16 Remainder

DIV mem	For 16-bit / 8-bit :
	AL ← AX / [mem8] Quotient
	AH ← AX % [mem8] Remainder
	For 32-bit / 16-bit :
	AX ← DX:AX / [mem16] Quotient
	DX ← DX:AX % [mem16] Remainder

Instruction Set

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

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

Signed Division

IDIV reg/ mem
IDIV reg
	For 16-bit / 8-bit :
	AL ← AX / reg8 Quotient
	AH ← AX % reg8 Remainder
	For 32-bit / 16-bit :
	AX ← DX:AX / reg16 Quotient
	DX ← DX:AX % reg16 Remainder

IDIV mem	For 16-bit / 8-bit :
	AL ← AX / [mem8] Quotient
	AH ← AX % [mem8] Remainder
	For 32-bit / 16-bit :
	AX ← DX:AX / [mem16] Quotient
	DX ← DX:AX % [mem16] Remainder

Instruction Set

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

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

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

Adapted from lecture notes by Dr Chintan Patel and Avani Dave

Instruction Set

1.Data Transfer Instructions
2.Arithmetic Instructions
3.Logical Instructions
4.String manipulation Instructions
5.Process Control Instructions
6.Control Transfer 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] – data** 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

Instruction Set

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

Instruction Set

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

Mnemonics: AND, OR, XOR, TEST, SHR, SHL, RCR, RCL …

AND A, data
AND AL, data8
AND AX, data16
	AL ← AL & data8
	AX ← AX & data16

AND reg/mem, data
AND reg, data8/16
AND mem, data8/16
	reg ← reg & data8/16
	mem ← mem & data8/16

Instruction Set

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

Mnemonics: AND, OR, XOR, TEST, SHR, SHL, RCR, RCL …

OR reg2/mem, reg1/mem
OR reg2,reg1
OR reg2,mem
OR mem,reg1
	
	reg2 ← reg2 | reg1
	reg2 ← reg2 | mem
	mem ← mem | reg1

OR reg/mem, data
OR reg, data8/16
OR mem, data8
	reg ← reg | data8/16
	mem ← mem | data

OR A, data
OR AL, data8
OR AX, data16
	AL ← AL | data8
	AX ← AX | data16

Instruction Set

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

Mnemonics: AND, OR, XOR, TEST, SHR, SHL, RCR, RCL …

XOR reg2/mem, reg1/mem
XOR reg2,reg1
XOR reg2,mem
XOR mem,reg1
	reg2 ← reg2 ^ reg1
	reg2 ← reg2 ^ mem
	mem ← mem ^ reg1

XOR reg/mem, data
XOR reg, data8/16
XOR mem, data8
	reg ← reg ^ data8/16
	mem ← mem ^ data

XOR A, data
XOR AL, data8
XOR AX, data16
	AL ← AL ^ data8
	AX ← AX ^ data16

Instruction Set

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

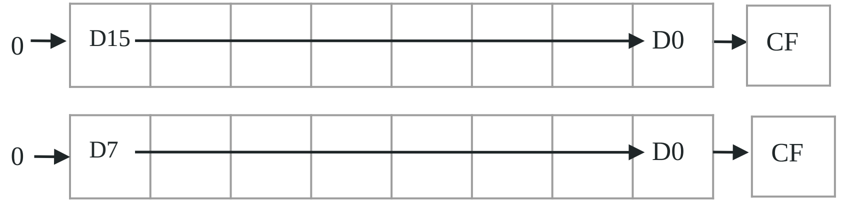
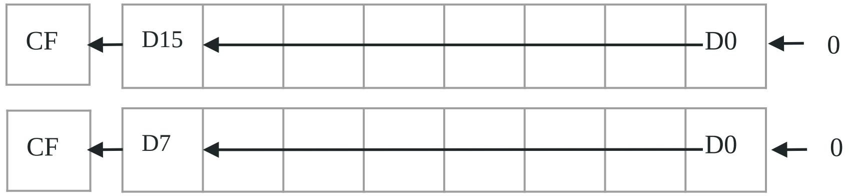
Mnemonics: AND, OR, XOR, TEST, SHR, SHL, RCR, RCL …

TEST reg2/mem, reg1/mem TEST reg2,reg1 TEST reg2,mem TEST mem,reg1	 Modify flags ← reg2 & reg1 Modify flags ← reg2 & mem Modify flags ← mem & reg1
TEST reg/mem, data TEST reg, data8/16 TEST mem, data8	Modify flags ← reg & data8/16 Modify flags ← mem & data
TEST A, data TEST AL, data8/16 TEST AX, data8	Modify flags ← AL & data8 Modify flags ← AX & data16

Instruction Set

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

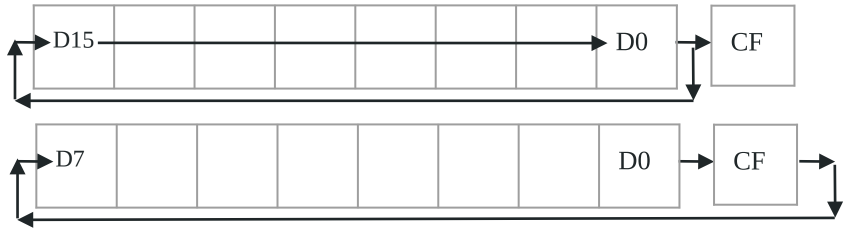
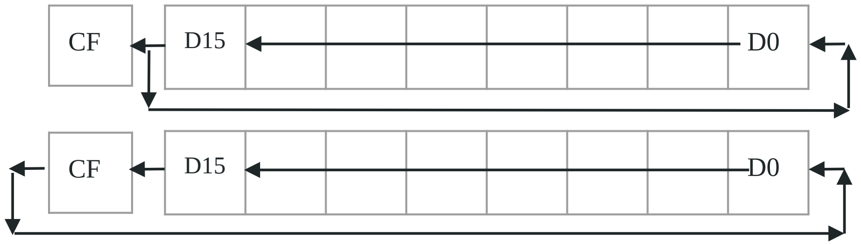
Mnemonics: AND, OR, XOR, TEST, SHR, SHL, RCR, RCL …

SHR reg/mem, reg1/mem SHR reg,1 SHR reg,reg SHR mem, reg	
SHL reg/mem, reg1/mem SHL reg,1 SHL reg,reg SHL mem, reg	

Instruction Set

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

Mnemonics: AND, OR, XOR, TEST, SHR, SHL, RCR, RCL …

ROR data, 12 RCR reg, reg RCR mem, reg	
ROL data, 12 RCL reg, reg RCL mem, reg	

Instruction Set

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

Mnemonics: : REP, MOVS, CMPS, SCAS, LODS, STOS...

REP **REPZ/ REPE** (Repeat CMPS or SCAS until ZF = 0)	While CX = 0 and ZF = 1, repeat execution of string instruction and CX $\leftarrow$ CX – 1
REPNZ/ REPNE (Repeat CMPS or SCAS until ZF = 1)	While CX = 0 and ZF = 0, repeat execution of string instruction and CX $\leftarrow$ CX – 1

Addressing Mode

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

STD	**Set direction flag DF ← 1**
CLD	**Clear direction flag DF ← 0**

Instruction Set

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

Mnemonics: : REP, MOVS, CMPS, SCAS, LODS, STOS...

MOVSB **8bit**	MA = DS * 16_{10} + SI MA_E = ES * 16_{10} + DI (MA_E) ← (MA) If DF = 0, then DI ← DI + 1; SI ← SI + 1 If DF = 1, then DI ← DI – 1; SI ← SI – 1
MOVSW **16bit**	MA = DS * 16_{10} + SI MA_E = ES * 16_{10} + DI (MA_E ; MA_E + 1) ← (MA; MA + 1) If DF = 0, then DI ← DI + 2; SI ← SI + 2 If DF = 1, then DI ← DI – 2; SI ← SI – 2

Instruction Set

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

Mnemonics: : REP, MOVS, CMPS, SCAS, LODS, STOS...

CMPSB **8bit**	$MA = DS * 16_{10} + SI$ $MA_E = ES * 16_{10} + DI$ **Modify flags ← (MA) – (MA$_E$)** If $(MA) > (MA_E)$, then CF = 0; ZF = 0; SF = 0 If $(MA) < (MA_E)$, then CF = 1; ZF = 0; SF = 1 If $(MA) = (MA_E)$, then CF = 0; ZF = 1; SF = 0 <u>For byte operation</u> If DF = 0, then DI ← DI + 1; SI ← SI + 1 If DF = 1, then DI ← DI - 1; SI ← SI – 1
CMPSW **16bit**	<u>For word operation</u> If DF = 0, then DI ← DI + 2; SI ← SI + 2 If DF = 1, then DI ← DI - 2; SI ← SI – 2

Instruction Set

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

Mnemonics: : REP, MOVS, CMPS, SCAS, LODS, STOS...

SCASB **8bit**	$MA_E = ES * 16_{10} + DI$ **Modify flags** $\longleftarrow$ **AL – (MA_E)** If AL > (MA_E), then CF = 0; ZF = 0; SF = 0 If AL < (MA_E), then CF = 1; ZF = 0; SF = 1 If AL = (MA_E), then CF = 0; ZF = 1; SF = 0 If DF = 0, then DI ← DI + 1 If DF = 1, then DI ← DI – 1
SCASW **16bit**	$MA_E = ES * 16_{10} + DI$ **Modify flags** $\longleftarrow$ **AL – (MA_E)** If AX > (MA_E ; MA_E + 1), then CF = 0; ZF = 0; SF = 0 If AX < (MA_E ; MA_E + 1), then CF = 1; ZF = 0; SF = 1 If AX = (MA_E ; MA_E + 1), then CF = 0; ZF = 1; SF = 0 If DF = 0, then DI ← DI + 2 If DF = 1, then DI ← DI – 2

Instruction Set

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

Mnemonics: : REP, MOVS, CMPS, SCAS, LODS, STOS...

LODSB	MA = DS * 16_{10} + SI **AL ← (MA)** If DF = 0, then SI ← SI + 1 If DF = 1, then SI ← SI – 1
LODSW	MA = DS * 16_{10} + SI **AX ← (MA ; MA + 1)** If DF = 0, then SI ← SI + 2 If DF = 1, then SI ← SI – 2

Addressing Mode

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

Mnemonics: : REP, MOVS, CMPS, SCAS, LODS, STOS…

STOSB	MA_E = ES * 16_{10} + DI **(MA_E) ← (AL)** If DF = 0, then DI ← DI + 1 If DF = 1, then DI ← DI – 1
STOSW	MA_E = ES * 16_{10} + DI **(MA_E ; MA_E + 1) ← (AX)** If DF = 0, then DI ← DI + 2 If DF = 1, then DI ← DI – 2

Addressing Mode

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

STC	**Set CF $\leftarrow$ 1**
CLC	**Clear CF $\leftarrow$ 0**
CMC	**Complement carry CF $\leftarrow$ CF'**

Instruction Set

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

STI	**Set interrupt enable flag IF $\leftarrow$ 1**
CLI	**Clear interrupt enable flag IF $\leftarrow$ 0**
NOP	**No operation**
HLT	**Halt after interrupt is set**
WAIT	**Wait for TEST pin active**
ESC opcode mem/ reg	**Used to pass instruction to a coprocessor which shares the address and data bus with the 8086**
LOCK	**Lock bus during next instruction**

Instruction Set

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

8086 Unconditional transfers

Transfer the control to a specific destination or target instruction
Do not affect flags

Mnemonics	Explanation
CALL reg/ mem/ disp16	Call subroutine
RET	Return from subroutine
JMP reg/ mem/ disp8/ disp16	Unconditional jump

Instruction Set

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

8086 conditional transfers

Operands	Comments
CALL	CALL
JMP	unconditional and conditional jump

CALL works in combination with the RET instruction.

To calculate the address, the offset specified by the label L1 is added or subtracted from the current address of the JMP instruction.

If it’s a forward jump, then the offset is added. EIP will hold this value.
If it’s a backward jump, the offset is subtracted. EIP will hold this value.

Instruction Set

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

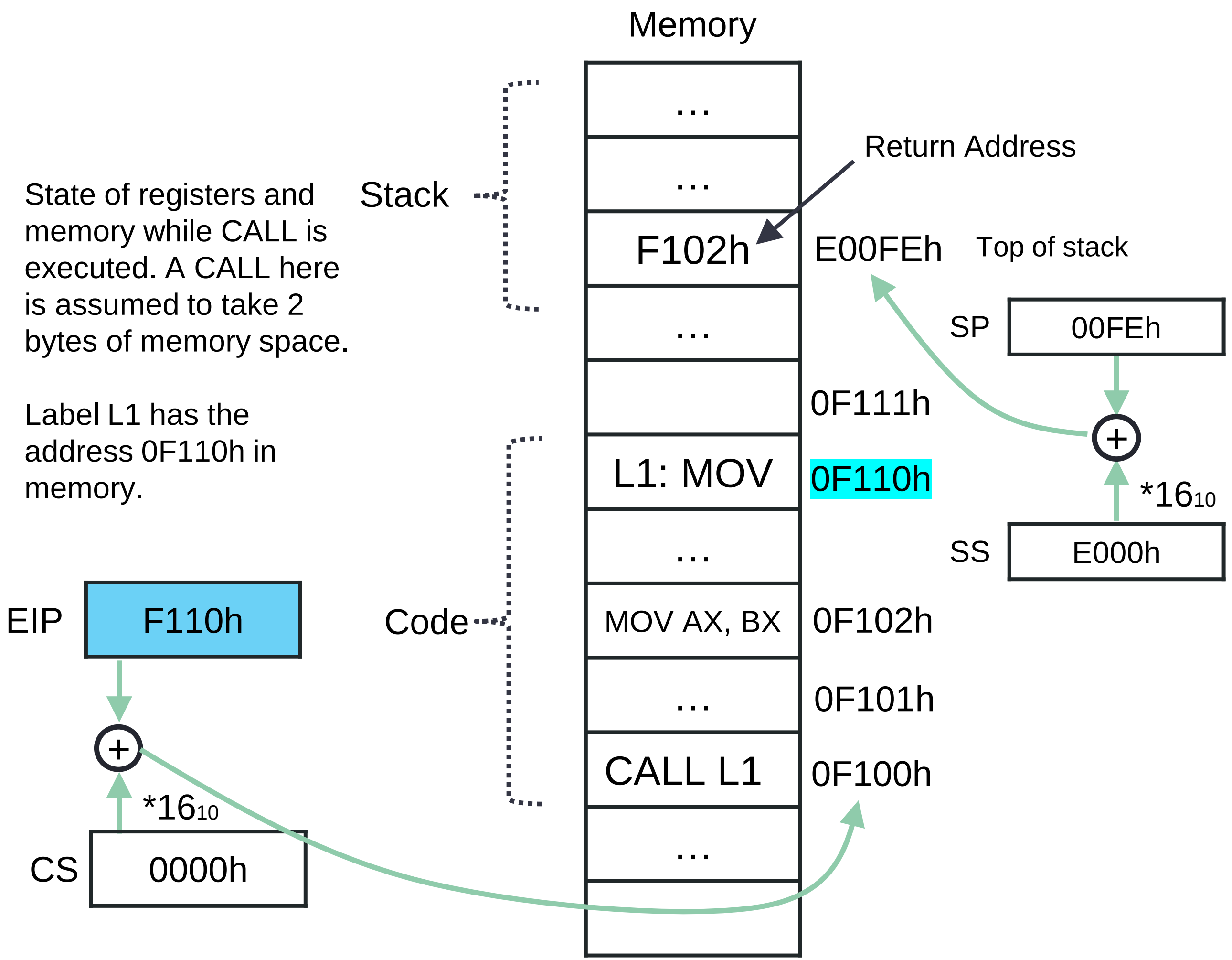
Code being executed

```
CALL    L1
MOV     AX, BX


L1:  MOV CX, AX
     RET
```

State of registers and memory while CALL is executed. A CALL here is assumed to take 2 bytes of memory space.

Label L1 has the address 0F110h in memory.



Adapted from lecture notes by Dr Chintan Patel and Avani Dave

Instruction Set

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

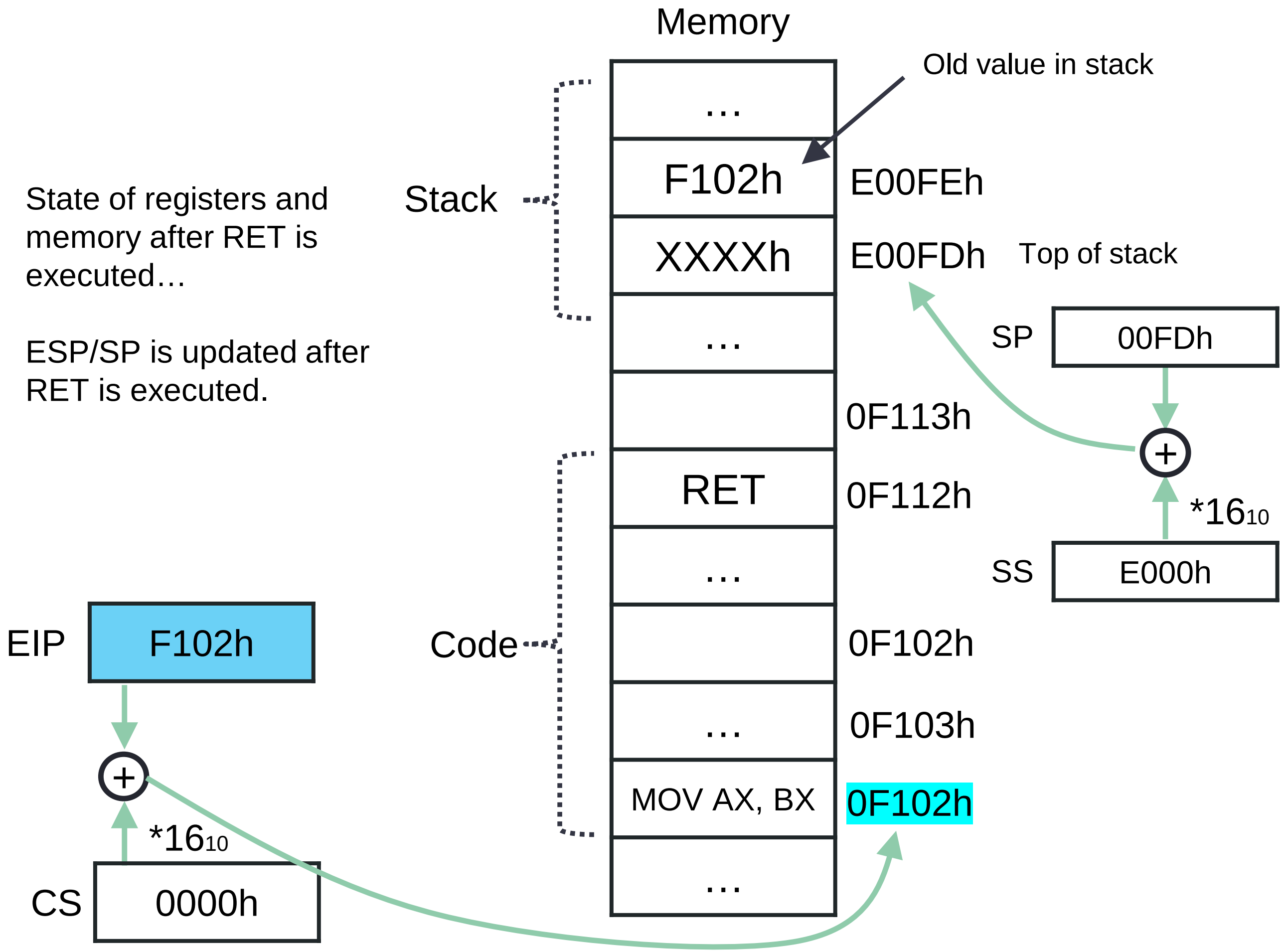
Code being executed

```
CALL    L1
MOV    AX, BX


L1:  MOV CX, AX
     RET
```

State of registers and memory after RET is executed…

ESP/SP is updated after RET is executed.



Adapted from lecture notes by Dr Chintan Patel and Avani Dave

Instruction Set

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

8086 unconditional transfers

Operands	**Comments**
CALL	Call a subroutine
JMP	unconditional and conditional jump

JMP Simply updates EIP to the location specified by its one and only operand.

To calculate the address, the offset specified by the label L1 is added or subtracted from the current address of the JMP instruction.

If it’s a forward jump, then the offset is added. EIP will hold this value.
If it’s a backward jump, the offset is subtracted. EIP will hold this value.

e.g.
```
JMP L1

L1: MOV AX, BX
```

Adapted from lecture notes by Dr Chintan Patel and Avani Dave

Instruction Set

8086 conditional transfers

Operands	Comments
JNC	Jump not carry
JE, JNE, JZ, JNZ	Jump equal/zero,not equal/zero
JA, JB, JAE, JBE	Jump above/below/or equal
JG, JL, JGE, JLE	Jump greater/less/equal to
JO, JNO	Jump overflow/not overflow
JS, JNS	Jump sign/no sign
JPO, JPE	Jump parity odd/even
JCXZ	Jump if CX == 0

Instruction Set

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

Instruction Mnemonic	Condition (Flag States)	Description
Unsigned Conditional Jumps		
JA/JNBE	(CF and ZF) = 0	Above/not below or equal
JAE/JNB	CF = 0	Above or equal/not below
JB/JNAE	CF= 1	Below/not above or equal
JBE/JNA	(CF or ZF) = 1	Below or equal/not above
JC	CF = 1	Carry
JE/JZ	ZF = 1	Equal/zero
JNC	CF = 0	Not Carry
JNE/JNZ	ZF = 0	Not equal/not zero
JNP/JPO	PF = 0	Not parity/parity odd
JP/JPE	PF = 1	Parity/parity even
JCXZ	CX = 0	Register CX is zero
JECXZ	ECX = 0	Register ECX is zero
JRCXZ	RCX = 0	Register RCX is zero
Signed Conditional Jumps		
JG/JNLE	SF = OF, and ZF = 0	Greater/not less or equal
JGE/JNL	SF = OF	Greater or equal/not less
JL/JNGE	SF ≠ OF	Less/not greater or equal
JLE/JNG	SF ≠ OF, or ZF = 1	Less or equal/not greater
JNO	OF = 0	Not overflow
JNS	SF = 0	Not sign (non-negative)
JO	OF = 1	Overflow
JS	SF = 1	Sign (negative)