28.02.2024

M-P for final

Register Indirect Addressing

register memory

MOV AX,[BX] transfers a byte on word between a register and a memory location addressed by an Index on base register. (BX/BP)

MOV CX,[BX] → copies the word contents of the data segment memory location addressed by BX into CX.

Index
- DI
- SI

Base
- BX
- BP

MOV AL,[BP] → copies the byte size cotents of data segment memory location addressed by Bp int #AL

[SP বা BP থাকলে stack segment হবে]

MOV AL, [EDX] → copies the byte size contents of data segment memory location addressed by EDx, into AL

MOV [DI], BH → copies BH into the byte sized data segment memory location addressed by DI.

table 9.3 - chapter 3

[Mov instruction দিয়ে memory to memory copy করা যায় না] we need to use MOVS [Ax], [Bx]

MOV Ax, Bx ✓
MOV Ax, [Bx] ✓
MOV [Ax], Bx ✓
MOV [Ax] [Bx] ✗
MOVS [Ax], [Ax] ✓

Base - plus - Index addressing

MOV AX, [BX + SI]
[BP + SI]
[BX + DI]
[BP + DI]

copies the word contents of the data segment memory location addressed by BX plus SI into AX.

MOV [BP + DI], CL → copies CL into the byte sized stack segment memory location addressed by BP plus DI

MOV [EBX + EDI], ECX → copies ECX into the doubl word size data segment memory location addressed by EBX plus EDI

MOV [BX], 10 H - Illegal

MOV Byte PTR [BX], 10 H - Legal

page: 102
Ex: 22, 23, 24, 25

03.03.21

Register Relative addressing

reg ↓ mem Ad [Base or Index R + Displacement]

MOV AX, [BX + 04H] → copies the word content of the data segment memory location by BX plus 04H into AX.

[BP + 02H]
[DI + 1234H]
[SI + 01H]

Note

Set [BX]
[Set + BX]

MOV Array [EBX], EAX

MOV [Array + EBX], EAX → Copies EAX into the double word size data segment memory location addressed by Array plus EBX.

Type	Instruction	Source	Address Generation	Destination
Register	MOV AX, BX	BX		AX
Immediate	MOV CH, 3AH	Data (3AH)		CH
Direct	MOV [1234H], AX	Reg AX	DS x 10H + 1234H	11234H mem Addr
Register Indirect	MOV [BX], CL	Reg CL	DS x 10H + BX	10300H mem Addr
Base plus Index	MOV [BX+SI], BP	Reg BP	DS x 10H + BX + SI	10500H
Register Relative	MOV CL, [BX+4]	10304H mem addr	DS x 10H + BX + 4	Reg CL
Base Relative plus index	MOV Array [BX+SI], DX	Reg DX	DS x 10H + Array + BX + SI	11300H mem

chapter:04

MC

Data Movement Instructions

Code	w=0 (Byte)	w=1 (word)	w=1 (doubleword)
000	AL	AX	EAX
001	CL	CX	ECX
010	DL	DX	EDX
011	BL	BX	EBY
100	AH	SP	ESP
101	CH	BP	EBP
110	DH	SI	ESI
111	BH	DI	EDI

MOD

MOD	Function
00	No displacement
01	8 bit dn
10	16 bit n
11	is R/M is a register

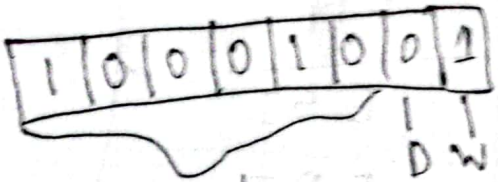
Mov AX, BX - Register

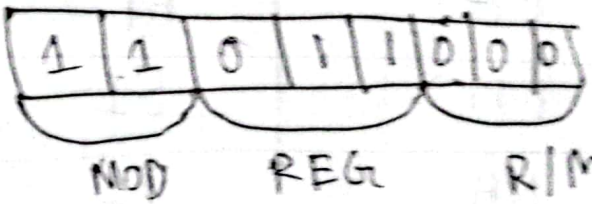
Mov AX, [BX] - No Dis

Mov AX, [BX + 04H]
01

Mov AX, [BX + 1234H]
10

Mov AX, BX - priority
to from





opcode

Machine equivalent code → 89D8

opcode = Mov = 100010

D = from (source) 0 // to (destination) 1

W = 1

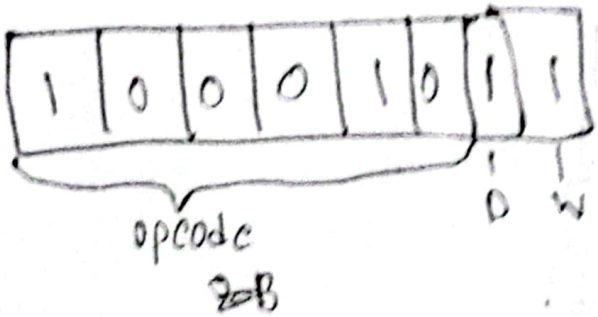
MOD = 11

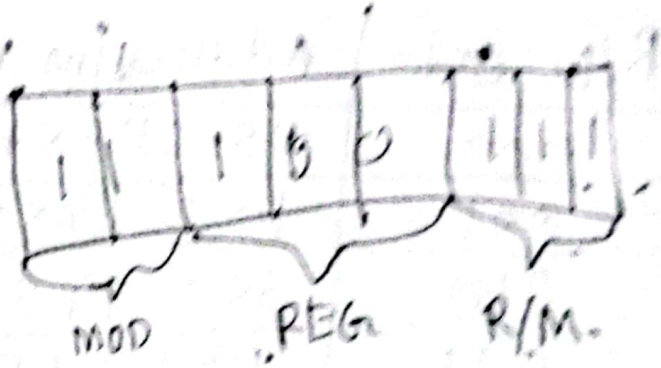
REG = BX (011)

R/M = AX (000)

priority

Q// MOV SP, DI





Equivalent machine code → 8BE7

Opcode = MOV = 100010

D = 1

W = 1

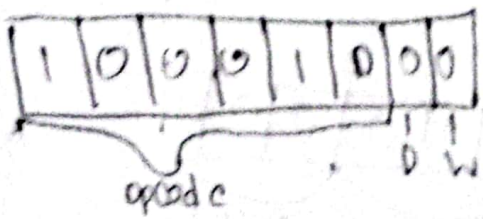
MOD = 11

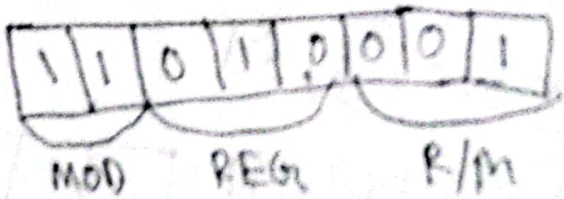
REG = SP = 100

R/M = DI = 111

Q// 88D1

MOV ~~DL, CL~~ CL, DL





Opcode = 100010

D = 0

W = 0

MOD = 11

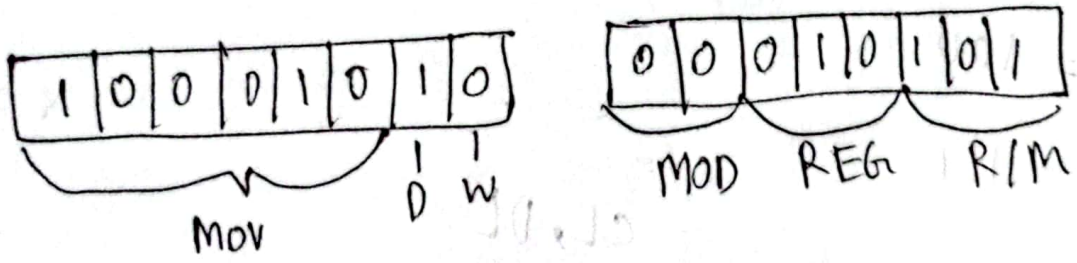
REG = 010 = DL

R/M = 001 = CL

16-bit memory Addressing mode

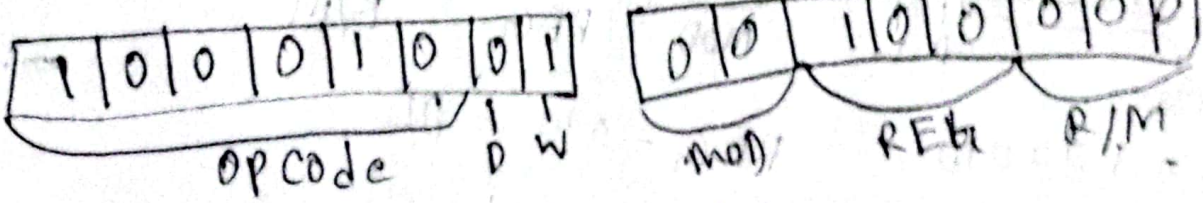
R/M code	Addressing mode
000	DS : [BX + SI]
001	DS : [BX + DI]
010	SS : [BP + SI]
011	DS : [BP + DI]
100	DS : [SI]
101	DS : [DI]
110	SS : [BP]
111	DS : [BX]

Q/ MOV DL, [DI]



machine code - 8A15

Q/ MOV [BX + SI], SP



Machine C → 8920

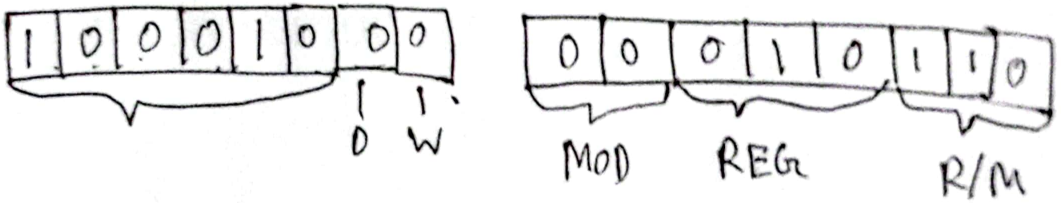
24.03.2024

Microprocessor

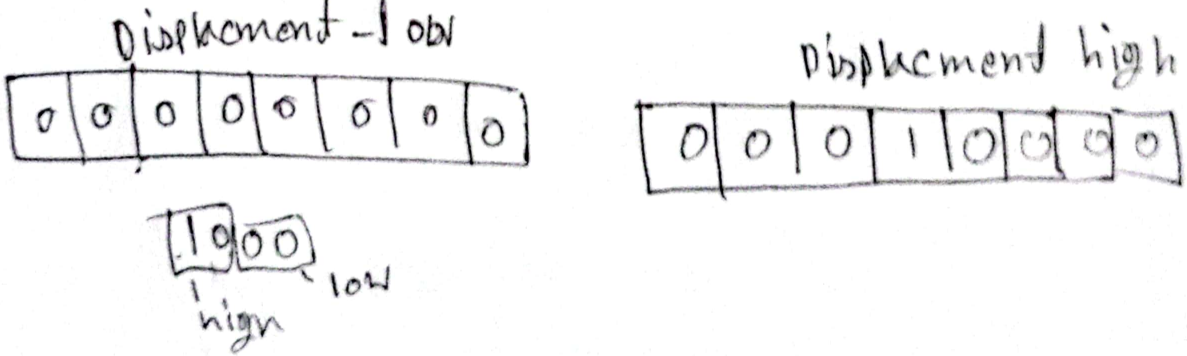
if memory is only a displacement in that case MOD = 00 and R/M = 110

Special case

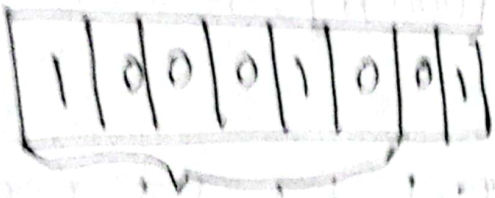
Mov [1000H], DL

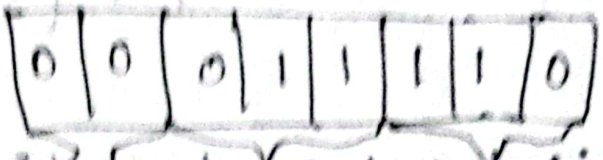


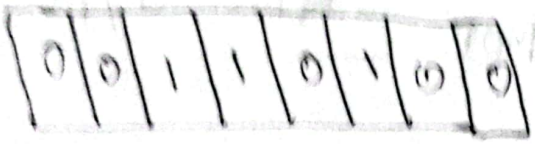
Whenever an instruction

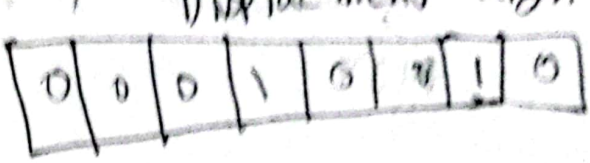


Q. Mov [1234H], Bx









Machine code → 8B1E 3412
(low high)

Important

Mov [12H], Bx — কোনো high low নেই শুধু displacment

special

Move Bx [BP+00] ; শুধু BP এর ক্ষেত্রে,
এর নিচে 2x
8bit displacment mod (01)
Displacment এর 00 হবে

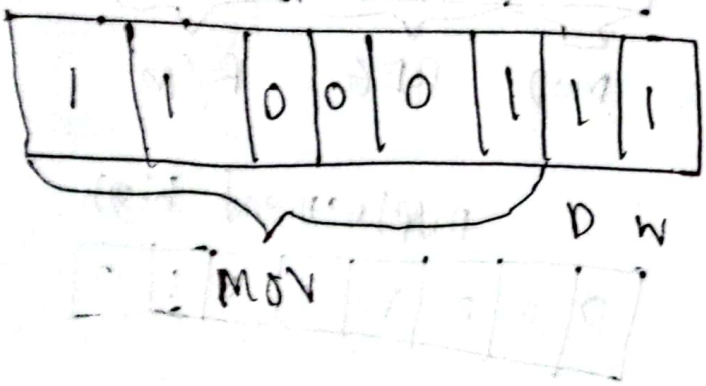
27.03.24

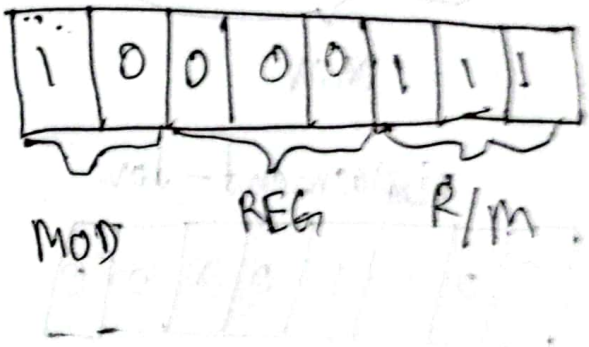
Immediate addressing

MPMC

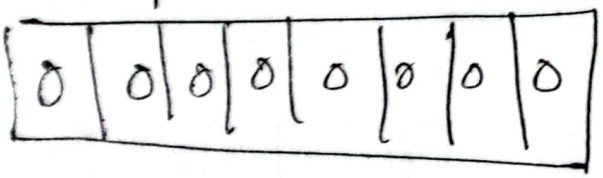
when Immidat (mov) → [1000]

MOV word PTR [BX + 1000H], 1234H

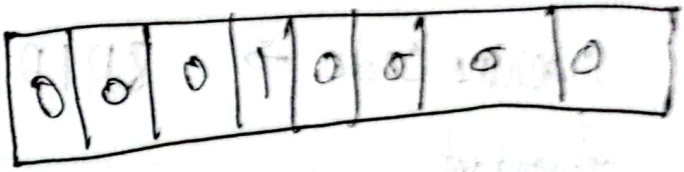




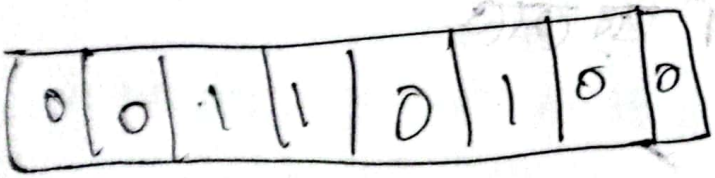
Displacement - low



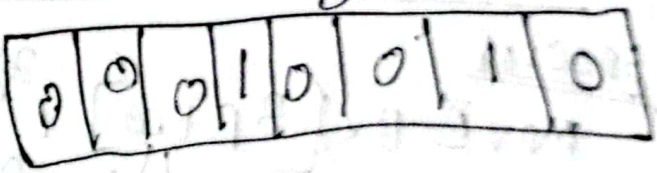
Displacment - high



Data - low



Data high



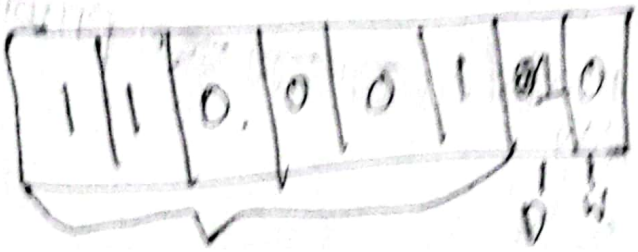
machine code :-

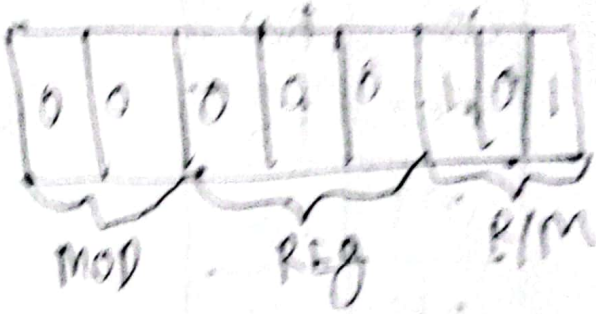
12 C600

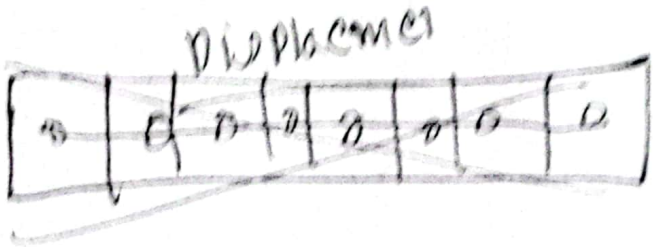
9 4 2 1
0 1 1 0

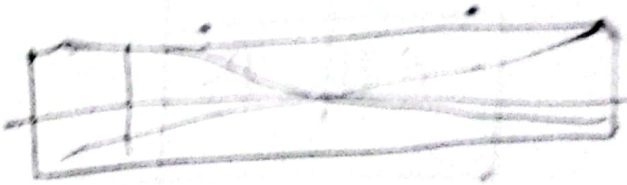
Q/ mov 67H, Byte PTR [DI] - Illegal

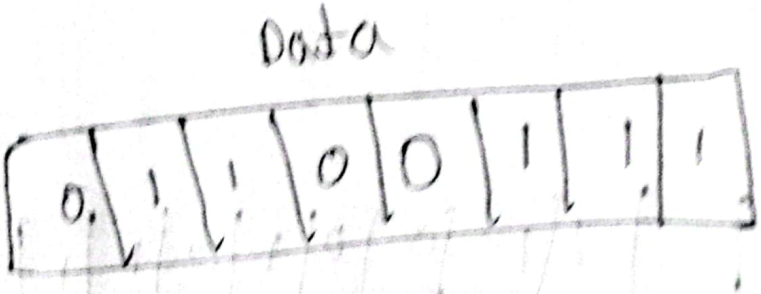
mov Byte PTR [DI], 67H











machine code: C6 05 67

Segment register

Mov - 100011

segment Reg এর size w = 0

ECS = 1

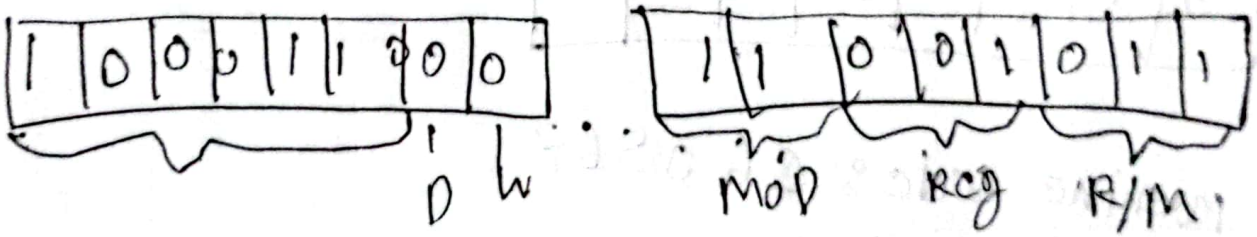
Cod	Reg
000	ES
001	CS
010	SS
011	DS

segment register এর তার মধ্যে priority

fixed priority

mov ES, SP

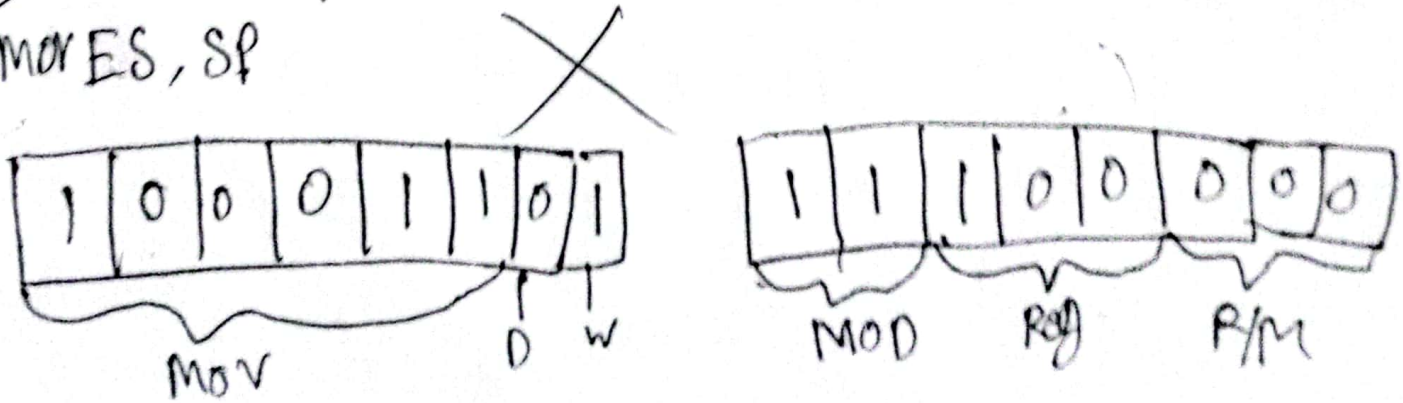
Q// mov BX, CS



machine code CCB

Ans = 8EC4

Q// mov ES, SP



M.C → 8DE0

31.04.2024

MP-MC

Load Effective Address

LEA

Mov Ax, NUM

LEA Ax, Num — Address এর copie করবে

Mov Ax, offset Num — same কাজ করে

page: 119

Paragraph below table

⇒ Diffrence bitween mov Ax, offset NUM, LEA, Ax, NUM or which one is suitable

31.03.2021

Mp-Mc

Push

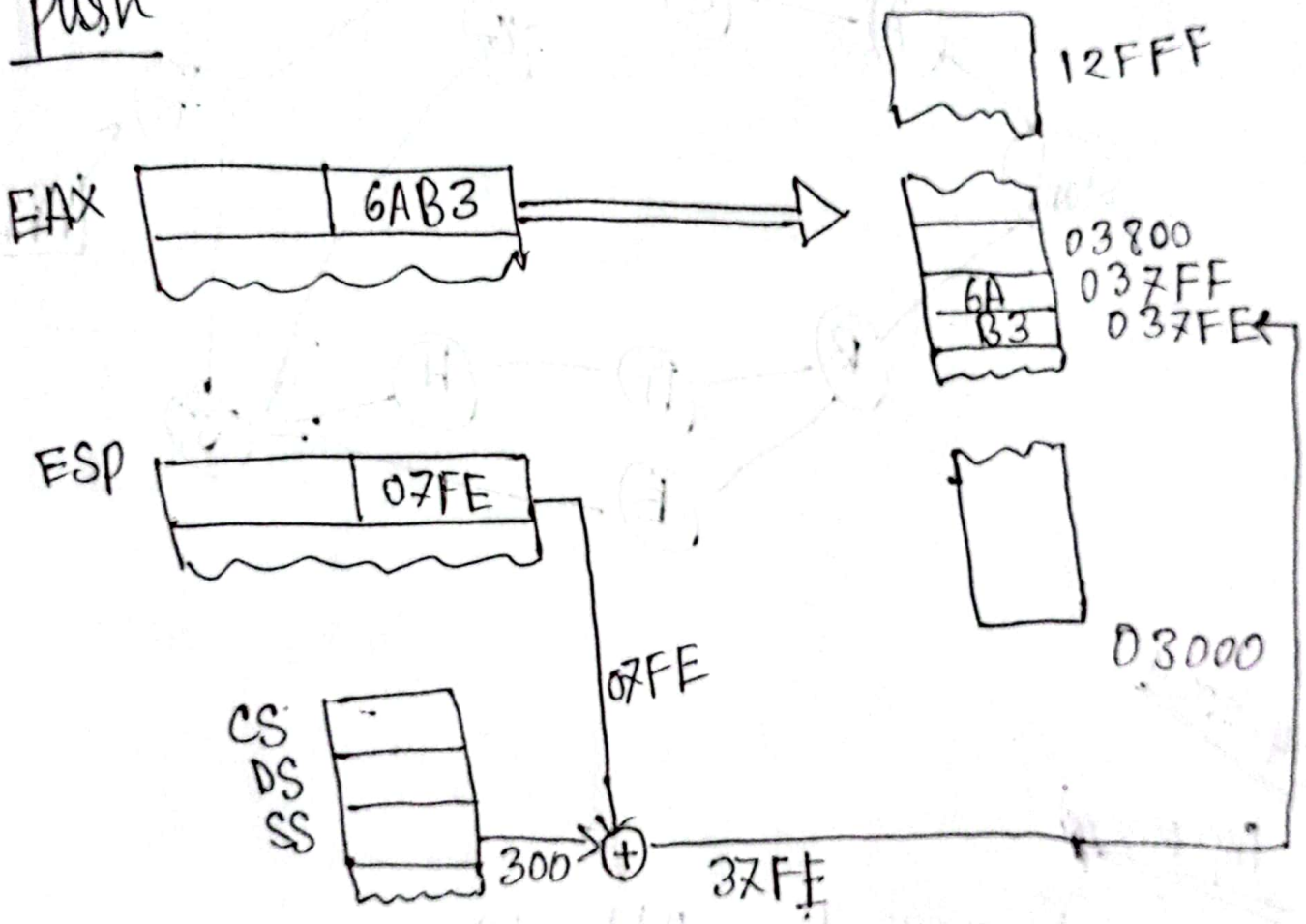


fig: The effect of the push Ax. Instruction on. Esp and stack memory locations 37FFH and 37FEH

LEA AX, NUMB ⟶ Loads Ax with the offset Address of Numb

LEA EAX, NUMB ⟶ Loads EAx with the offset Address of Numb

LDS DI, List ⟶ Loads DS and DI with the 32-bit contents of the data segment memory location List

LDS EDI List - Loads EDI and DS with 48 bit...

LES BX, cat

LFS DI, DATA1

LGS SI, DATA3

LSS SP, MEM

String Data transfer instruction

1. LODS
2. STOS
3. INS
4. OUTS
5. MOV

Q. Types of string data transfer instruction.

03.04.2024
MPMC

(1) LODS instruction loads AL, Ax. or EAx with data stored at the data segment offset address indexed by the SI Register

Instruction	
LODSB	AL = DS : [SI]; SI = SI ± 1
LODSW	Ax = DS : [SI]; SI = SI ± 2
LODSD	EAX = DS : [SI]; SI = SI ± 4

LODS List : { AL = DS : [SI]; SI = SI ± 1 (If List is byte
Ax = DS : [SI]; SI = SI ± 2 (if a word)
EAx = SS : [SI]; SI ± 4 (if a DW)

(3) STOS instruction stores the data at AL, Ax or EAx in the Extra segment offset address indexed by the DI register.

Instruction	
STOS B	ES : [DI] = AL; DI = DI ± 1
STOS W	ES : [DI] = AX; DI = DI ± 2
STOS D	ES : [DI] = EAX; DI = DI ± 4
STOS List	{ ES : [DI] = AL; DI = DI ± 1 (If it is byte ES : " " ± 2 (if it is word " " " ± 4 (if it is DW)

③ MOVS instruction transfer a byte, word, or double word from the data segment location addressed by SI to the Extra segment location addressed by DI

MOVSB. ES:[DI] = DS:[SI]; DI=DI ±1; SI=SI ±1 ↓ (Byte transfered)

MOVSW ES: ..... ±2; ... ±2 (word transfered

MOVS Num1, Num2

4d9 Table

exchange (swap

XCHG AL, BL

conditional move instruction

C MOVB ~~if above or equal~~ 10

C MOV AE if above or equal
C MOV E if equal
C MOV Z

C MOV O

C MOV P (parity ffn

C MOV PO (parity odd 22

Chapter 06

Program control Instruction

Unconditional Jump

Intrasegment Jump

1) Short Jump → ① bytes
   ② opcode | Disp
   ③ Jump to memory location within +127 bytes to -128 bytes from the address following the jump

2) Near Jump → ① 3 bytes
   ② opcode | Displ low | Displ high
   ③ Jump within ± 32 kB/or anywhere in the current code segment from the instruction in the diffrent code segment

Inter Segment Jump

3) FAR Jump → ① 5 bytes
   ② opcode | IP Low | IP High | CS Low | CS High
   ③ Allow a jump to any memory location within the real memory system

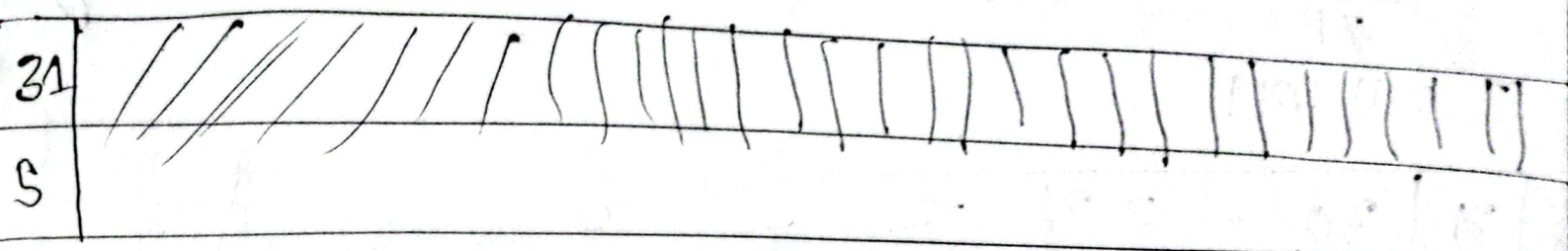
online
24.04.2024

31	30	23	22	0
S	Biased exponent		magnitude	

short with a bias of 7FH

16|8
16-2

2F
3
8.2H



63	62	52	51	0
S	Biased exponent		magnitude	

Long with a bias of 3FFH

2|83
41

1 01 0010

② // convert 9.75_{10} to single-precision (short real) floating point

1001.0111 0101

	30	23	51	0
0	1 0 0 0 0 0 1 0			

Q2. .078125 single precision.

.078125 Decimal = 0.00010 1 binary

7F
-4
7B

= 1.01 E-4 scientific binary

0111 1011

31	30 – 23	22 – 0
0	0 1 1 1 1 0 1 1	0 1 0 0 0 0 . . . . .

28.04.2024

Q// convert $(152.1875)_{10}$ to double precision floating point

$152.1875 = 0\,10011000\,0011000)_2$

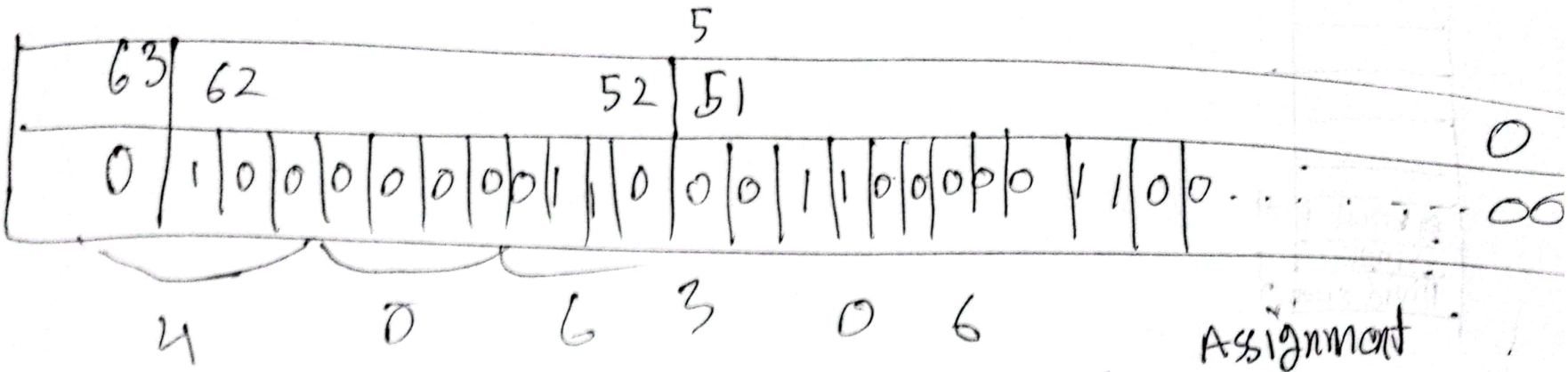
~~.1875 = 1100~~

Scientific Binary = 1.0011000 1100 E7

S = 0

Biased exponent $(7+3FF) = (406)_{16} = 1000\,0001\,1$

Magnitude = 00 1100 0011 000



Assignment

8.5.23

Indirect add segment register with [BP]

Interrupt

3 type of interrupt

Hardware interrupt

Software interrupt

Error condition

Interrupt pointer table:

256 ta interrupt