EA = (DS × 10H) + offset
Data Segment

UNIT - 2

JC → when carry is 1
jump to label

JNZ → jump on non zero

INSTRUCTION FORMAT.

opcode + operand

1. One Byte instruction HLT 1 byte

2. Register to Register MOV AX, BX → 2 byte

3. Register to/from memory without displacement. 2 byte

4. Register to/from memory with displacement 2 or 3 byte (1 byte)

4. Transfer the immediate operand to register 3 or 4 byte (if it is 1 → 3 byte)

5. Transfer operand with displacement 5 byte or 6 byte

Addressing Mode. — Sequential control flow / Control transfer.

the way to get operand

IP → points the address of next instruction

Sequential Control flow.

MOV AX, 1010H — 1. Immediate addressing mode [when operand is given in the instruction] (data is a part of the instruction)

MOV AX, [1200H] — 2. Direct addressing mode [when address is given in the instruction] EA = DS×10H + Offset

MOV AX, BX — 3. Register addressing mode [when data is in register) (no effective address)

MOV AX, [BX] — 4. Register indirect mode [when address is in register) (other than SI, DI) EA = DS × 10H + [BX]

MOV AX, [BX] 1234H — 5. Register relative addressing mode {displacement either 8 bit or 16 bit is given) EA = DS ×10H + BX + Displacement

MOV AX, [SI] — 6. Index addressing mode → (SI, DI) → EA = DS × 10H + [SI] (addr 1200 data in 1200 is not data)

MOV AX, [BP] [SI] — 7. Based Index addressing mode [BP, SI], both. EA = DS×10H + [BP] + [SI]

MOV AX, [BX] [DI] 5000H — 8. Relative base index addressing mode. EA = DS×10H + [BX] + [DI] + 5000H

Control transfer.

unconditional jump - JMP
Conditional jump - JC or JNZ. } based on flag.

Intra Segment — Direct / Indirect

8 bit no Short Jump.

16 bit no. Long Jump.

Inter Segment — Direct / Indirect.

JMP-(relative number) → no of byte has to jump

without giving.

JMP [BX]

In jump no need of data.

JMP (1234 H) → address.

Direct → JMP CS, IP.

i. Find the effective address

Offset displacement = 5000H.

[AX] - 1000H, [BX] - 2000H

[SI] - 3000H, [DI] - 4000H

[BP] - 5000H, [SP] - 6000H,

[CS] - 0000H, [DS] = 1000H,

[SS] - 2000H, [IP] - 7000H.

1. Direct addressing mode.

MOV AX, [5000H].

EA = DS × 10H + 5000H

= 10000H + 5000H

= 15000H

2. Register indirect

MOV AX, [BX].

EA = DS × 10H + [BX]

= 10000H + 2000H

= 12000H

3. Register relative

MOV. AX, [BX] 5000 H

EA = 10000 H + 2000 H + 5000 H

= 17000H

4. Index addressing mode.

MOV. AX, [SI].

EA = DS × 0H + 3000 H

= 10000H + 3000 H

= 13000H.

5. Based Index

MOV AX, [B~~P~~X] [SI]

EA = DS × 10H + [B$\overset{X}{\cancel{P}}$] + [SI]

= 10000H + $\overset{2000}{\cancel{5}}$000H + 3000H

= 1$\overset{5}{\cancel{8}}$000H

6. Relative based index.

MOV AX, [BX] [DI] 5000H

EA = DS × 10H + [BX] + [SI] + 5000H

= 10000H + 2000H + 3000H + 5000H

= 1A000H.

10+2+5

```
 10000
  2000
  3000
  5000
 -----
 1A000
```

1. CALL [BX]

IP

CALL 2000H^{CS} : 0050H

MP

Intra - direct.

01

02

OFFFF

INC SI

01

02

03

04

XCHG [5000H], AX
XCHG BX, AX

8 bit → op code instruction 2
16 bit → " 2^16

INSTRUCTION SET OF 8086.

one operand
INC, DEC, JMP
No operand
HLT

1. Data tranfer instruction.

No flags modified

Transfering data from one register to another register or from/to memory to address.

POP → retrieve data from stack

PUSH – Push to stack

PUSH · AX

des · source
MOV [1200H], AX

we can't load data to the segment
MOV DS, 1000H X

SS X10H + SP = EA
SS = 2000H

MOV AX, [1200H]

MOV AX, 1000H
MOV DS, AX

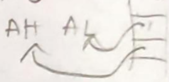
→ data	2FFFD	FFFD	→ SP
	2FFFE	FFFE	
	2FFFF	FFFF	

MOV AX, BX

MOV AL, 1234H x
AL is one byte 1234 is 2 byte

'LIFO' last in first

2. ARITHMETIC AND LOGICAL INSTRUCTION.



PUSH → PUSH · AX.

POP → retrieve data from stack.

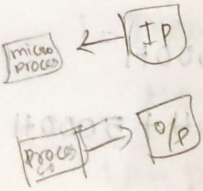
XCHG [5000H], AX

XCHG BX, AX.

IN : Input → port address.

2^{16} → devices can be connected

IN AL, 03H



IN AX, DX

0 – ASCI – 30

MOV DX, 0800H

IN AX, DX.

OUT port address, AX.

XLAT – translate.

MOV AX, SEG TABLE

MOV DS, AX

MOV AL, CODE

MOV BX, OFFSET TABLE

XLAT.

LEA - load effective Address.

LEA BX, ADR → address

LEA SI, ADR [BX].

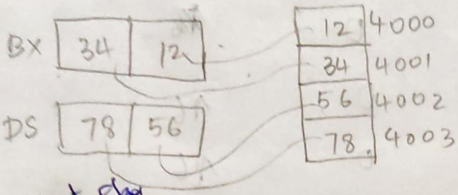
no flags are modified

LDS / LES - Load pointer to DS / ES.

LDS BX, 4000H

MOV BX, [4000H]

MOV DS, AX



affect flag

LAHF - load AH ~~with~~ to flg register AH ← flag register

~~affect the flag~~

SAHF - AH → flg ~~to~~ AH

affect flag

PUSHF → from stack to flag

POPF → flag to stack.

2. ARITHMETIC INSTRUCTION. 6 flags will be modified

ADD: all conditional flags are affected.

ADD AX, 1234H

ADD AX, BX

ADD AX, [SI]

ADD AX, [5000H].

CMP → Compare. (Subtraction)

(zero & carry affected)

CMP AX, BX

AX > BX C=0 Z=0

AX = BX C=0 Z=1

AX < BX C=1 Z=0

ADC → Add with carry

ADC AX, 1234 H

ADD AX, BX

ADD AX, [SI]

ADD AX, [5000H]

AAA → ASCII Adjust After Addition

unpacked BCD

in 1 byte one digit

SUB → Subtract

SBB → Subtract with borrow (carry flag is used ~~as~~ borrow)

Write a program to perform addition of 2-8 bit with carry

MOV AX, [1100H].

if lower nibble of AL	AC 0	higher nibble of AL	AL
0 - 9	0	0	~~[illegible]~~
0 - 9	1	0	16
>9		0	16

16 | 36 2-4 24

Write a program to perform addition with carry and subtraction with borrow

```
MOV AL, 04H
MOV BL, 09H
MUL BL
AAM
```

AH	AL
03	06

DAA & DAS → Decimal instruction.

NEG : 2's. Complement

CBW → Convert Byte into Word.
Sign bit is used to fill upper byte

7 8 → 0000 0000 0111 1000

8F → 1111 1111 1000 1111

CWB

DIV → Division → Quotient → AL AX
Reminder → AH DX

IDIV → Sign

Logical.

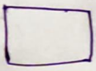
AND dst, src

AND [5000H], DX → direct.

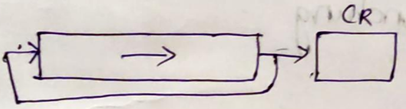
AND DX, [5000 H] → indirect.

NOT → no immediate addressing mode.

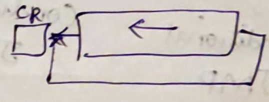
SHL / SAL : Shift logical / Arithmetic Left



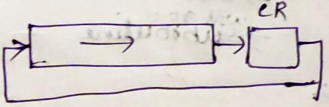
ROR. Rotate Right without carry.



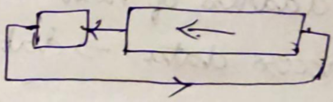
ROL



RCR Rotate with carry.



RCL.



8 String Manipulator Instruction

REP : Repeat instruction prefix.

REPE / REPZ. (→ equal.)

RENE / REPNZ. (→ not equal.)

MOVSB / MOVSW

```
MOV AX, 5000H
MOV DS, AX
MOV AX, 6000H
MOV ES, AX.
MOV CX, 0FFFH
MOV SI, 1000H
MOV DI, 2000H
CLD
REP MOVSB.
```

CLD - Clear the Direction flag

if DF 0 → increment

DF 1 → decrement

CX → Counter (or) length

CMPS – Compare String

SCAS → Scan string (equal na stop) unequal no iterate

```
MOV AX, SEG1
MOV DS, AX
MOV AX, SEG2
MOV ES, AX
MOV SI, Offset string1
MOV DI, Offset string2
MOV CX, 0010H
CLD
REPNE CMPSW
```

SI, DI will be decremented or incremented

for chking the particular word is in the string

```
MOV AX, SEG
MOV ES, AX
MOV DI, Offset
MOV CX, 010H
MOV AX, WORD
CLD
REPNE SCASW
```

DS	ES
SI	DI

LODS → load String AX ← from string

STOS → store string AX → string

3. Control Transfer OR Branching

- UnConditional
  - JMP
- Conditional
  - JC, JNC.

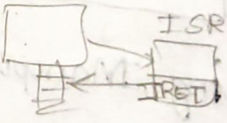
CALL

→ last instruction of subroutine – return

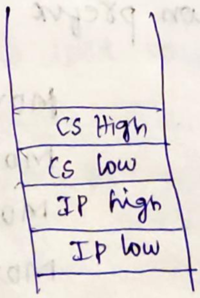
→ will go to subroutine

to find address of next → CS, IP

access data → data Segment.



INT N (Interupt Type N)



INTO

when it is over it interrupt the processor.

```
      MOV  CX,0005                    REP => string only.
      MOV  BX, EFFFH                  edge → one level to
Label MOV  AX, Code 1                 another level at that
      OR   BX, AX                     instant it is active
      AND  DX,AX      location where ISP available
      loop Label.     INT 20x4
                      0010
```

Single step- trap flag.

Addition with carry.

```
MOV CX, 0000H
MOV AX, Data 1          Sum ≤ FFFF H                15
MOV BX, Data 2                                      4
ADD AX, BX          → AX ← AX + B               16 [19
JNC Carry                                           1  -3
JNC cx                  FFFF
MOV [1200H],AX          1234
MOV [1202H]           1 1233 → AX
HLT.                  Stored in
```

Machine Control Instruction.

Subtraction with borrow.

Write a Code to find no of even & odd numbers in a given array