Lecture-1

8086/Microprocessor
8088

Chapter-1 (Charles Manut এর বই)

Computer Organization and Assembly Language

Microcomputer Systems :

I) I/O devices are called peripherals.

Input devices - keyboard, mouse

Output devices - moniton.

I/O device এর সাথে microcomputer এর relation কে interfacing বলে।

II) Integrated Circuits, also known as chips. digital circuits. discrete voltage এ কাজ করে — high/low.

IC circuits are known as digital circuits because it operates on discrete voltage signal levels typically a high voltage and a low voltage represent by 0 and 1's

III) These symbols are called binary digits or bits.

IV) All information processed by the computer is represented by strings of 0's and 1's that is by bit string.

Motherboard:

I) The Motherboard is a printed circuit board that is the foundation of a computer, located at the bottom of the computer case.

II) It allocates power to the CPU, RAM, and all other computer hardware components.

III) Most importantly, the motherboard allows hardware components to communicate with one another.

Bytes and Words:

I) Information processed by the computer is stored in its memory.

II) The memory circuits are usually organized into groups that can store eight bits of data.

III) A string of eight bits is called a byte.
A string of sixteen bits is called a word.

IV) Each memory byte is identified by a number that is called address.
Address দিয়ে memory location কে access করা যায়।

V) The data stored in a memory byte are called its content.

VI) The address of a memory byte is fixed and is different from the address of any other memory byte in the computer.

VII) The contents of a memory byte are not unique and are subject to change, because they denote the data currently being stored.

Bit position:

Memory address 16 bit এর হয়,

I) The positions are numbered from right to left, starting with 0.

II) In a word, the bits 0 to 7 form the low byte and the bits 8 to 15 form the high byte.

III) For a word stored in memory, its low byte comes from the memory byte with the lower address and its high byte is from the memory byte with the higher address.

CamScanner

Buses

I) A processor communicates with memory and I/O devices by using signals that travel along a set of wires or connections called buses.

II) There are three kinds of signals : address, data and control.

III) And there are three buses : address bus, data bus and control bus.

IV) To read the contents of a memory location, the CPU places the address of the memory location on the address bus, and it receives the data, sent by memory circuits, on the data bus.

V) A control signal is required to inform the memory to perform a read operation.

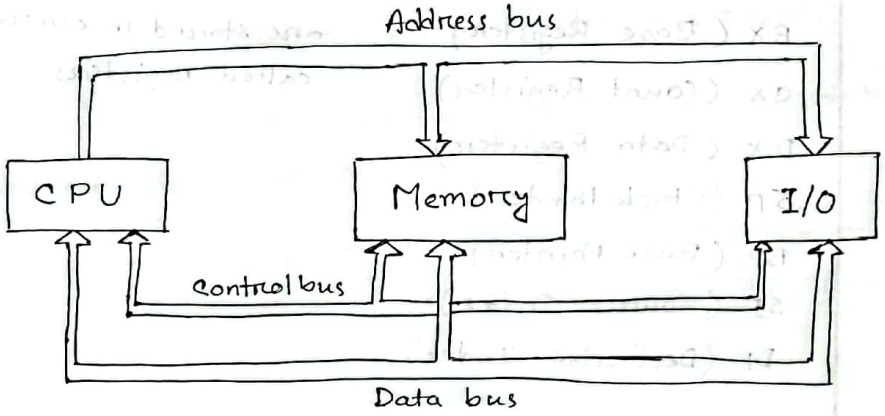
VI) The CPU sends the control signal on the control bus.

Lecture-2

Address bus - যে location থেকে data read বা write হয়, সেই location টার CPU address CPU থেকে Bus এ দেয়।

Data bus - কোনো specific location এ যে data read বা write হয় সেটা data bus এর মধ্য দিয়ে যাবে।

Control bus - পুরো সিস্টেম control করে। সেজন্য control signal এ ধরনের হয়, read operation এর জন্য read signal. এসব signal control bus এর through CPU send করে।



Intel 8086 Microprocessor Organization

Two units:- i. Execution Unit : Executes any instruction

ii. Bus Interface Unit : Deals with the buses

Execution unit কোনো instruction execute করার পর, bus unit next কোন instruction execute করতে হবে, সেটার info store করে রাখে,

* General Purpose Register : 8 registers

AX (Accumulator Register)
BX (Base Register)
CX (Count Register)
DX (Data Register)
SP (Stack Pointer)
BP (Base Pointer)
SI (Source Index)
DI (Destination Index)

data for the operations are stored in circuits called registers.

* Internal Bus :

Execution unit and Bus Interface unit connect করে,

* Temporary register : operand hold করায়,
* ALU : Arithmetic and Logic Unit
* Flag register : operation result reflect করে,

* Registers

CS (Code Segment)
DS (Data Segment)
SS (Stack Segment)
ES (Extra Segment)
IP (Instruction Pointer)

* Bus Control logic

* Instruction queue : 6 byte বড়,

next কোন instruction execute হবে সেটা bus interface unit ও instruction queue ও store করে রাখবে।

BIU is responcible for transmitting addresses, data and control signals on the buses.

The instruction pointer (IP) contains the address of the next instruction to be executed by the EU.

Instruction prefetch:

While the EU is executing an instruction, the BIU fetches up to six bytes of the next instruction and places them in the instruction queue.

Machine Instruction:

Opcode - The opcode specifies the type of operation

Operand - The operands are often given as memory addresses to the data to be operated on.

Fetch - Execute cycle:

* Fetch -
  - fetch an instruction from memory
  - Decode the instruction to determine the operation.
  - Fetch data from memory if necessary
* Execute -
  - Perform the operation on the data.
  - Store the result in memory if needed.

C = a + b

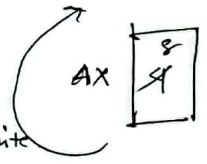
↓ ↓ ↓

AX BX + CX

Programming Language.

* Machine language:
  - CPU can only execute machine language instructions
  - They are bit strings.
  - Instruction - 1010000100000000 0000000
    Fetch the contents of memory word 0 and put it in register AX.

word 0 থেকে read করে AX এ রাখতে হবে.



  - Add 4 to AX.
  - AX in memory word 0.

AX = AX + 4.

W0 = AX.

Assembly Language :

Assembly instruction -

```
MOV AX, A       ; fetch the contents of location A and
                  put it in register AX.
Add AX, 4       ; Add 4 to AX.
MOV A, AX       ; move the contents of AX into
                  location A.
```

Chapter-2 → Self Study

Lecture-3

Chapter-3

Register →

Data Register → instruction execute করতে data আনার জন্য store

Address Register → data address store

Status Register → operation এর পরে data status/information store করা.

Address Register

- Segment
- Pointer
- Index

Status Register

- FLAG Register

CamScanner

Data Register :

AX (Accumulator Register) : Arithmetic operation, Logical operation (MUL, DIV, ADD, SUB, AND, OR ইত্যাদি, Input Output)

~~BX~~
~~CX~~

BX (Base Register) : XLAT → এক table থেকে data আনতে table এ index করতে

CX (Counter Register) : instruction এ loop use করলে.

MOV CX 10 (CX এর value 10)

REP (Repeat operation)

DX (Data Register) : MUL, DIV, I/O operation এর output show করার জন্য

Address Register :

1. Segment Register :

CS (Code Segment) → Code রাখার জন্য

DS (Data Segment) → data store

SS (Stack Segment) → stack operation use এর জন্য

ES (Extra Segment) → more than 16 bits হলে

Physical address 20 bit ↘

2^{20}, 1 MB .. → 8086 এর memory.

Register গুলো 16 bit এর,

Memory কে ছোট ছোট block এ ভাগ করে নিলে access করতে সুবিধা হয়.

00000000 . . . 000
00000000 . . . 001
⋮
11111111 . . . 111

Address range → 00000H to FFFFFH.

Memory কে কতগুলো block এ ভাগ করা হয় সেটাই size $2^{16} \rightarrow 64kb$

16 bit এর memory কে segment করা হয়।

একেকটা segment এর ভেতরে specific location এর address কে offset বলা হয়।

~~Segment 0 → 00000~~

0000f
00002
00001
Segment 0 → 00000

0001f
00012
00011
Segment 1 → 00010 → offset

last → fffff

Logical Address
Segment : Offset

675760
18546

A4FB : 4872h

~~4bit করে left shift~~
Hex multiplication
Hex 10

Physical address = segment × 10 + offset.

∴ Physical address = A4FB0 + 4872
= A9822 h

Ex. 3:1 Physical address : 1256A h.

Segment X : 1256h

~~offset~~ y : 1240h
segment

$1256A = 1256 \times 10 + \text{offset } X$

$X = A\,h.$

$1256A = 1240 \times 10 + Y$

$Y = 16A\,h.$

logical address → 1256 : 000A.

logical address → 1240 : 016A.

Pointer Register:

Pointer
- SP (Stack Pointer) → SS এর সাথে use হয় segment & pointer মিলে location
- BP (Base Pointer) → stack এর data access করার জন্য

Index
- SI (Source Index) → DS
- DI (Destination Index) → ES

SI
↓
RUET
TEUR .

LAB-1

- MODEL SMALL
- STACK 100H
- DATA (not mandatory : variable initialize/ করতে চাইলে define)

  DB → Double byte
  DW → word.

  A DW 2
  B DW 5
  SUM DW ? → value unknown থাকবে

- CODE

```
MAIN PROC   [int main()]
    MOV AX, @DATA
    MOV DS, AX
    MOV AX, A   ; AX = A
    ADD AX, B   ; AX = A+B
    MOV SUM, AX
    MOV AX, 4CH
    INT 21H
    MAIN ENDP
END MAIN
```

Source
Destination

BIOS – Basic I/O system

DOS – Disc Operating system

EXP = A+B-C

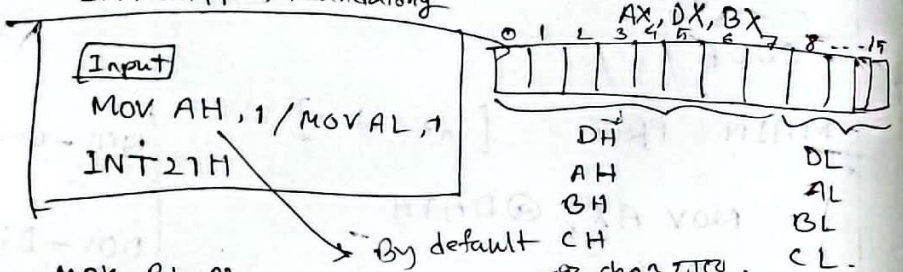
Input, Output এর
* keyboard থেকে char নিয়ে print. • @, _, %

variable name হিসেবে এগুলো ব্যবহার করা যায়।

```
• MODEL SMALL
• STACK 100H.
• CODE
  MAIN PROC
    MOV AH,2 → output
    MOV DL,'?'
    INT 21H → mandatory
```

output show করাতে হলে DL দিতে হয়

DX → 16 bit
DH (high byte) DL → Low byte



MOV BL,AL.

Newline

```
MOV AH,2
MOV DL,0DH
INT 21H.
MOV DL,0AH
INT 21H
MOV DL,BL
INT 21H
MOV AH,4CH
INT 21H
MAIN ENDP
END MAIN
```

Lecture-4

IP (Instruction Pointer)

program এ next যে instruction run হবে তার address কে point করে রাখে।

FLAG Register

Parity → PF = 1
Zero → ZF = 1 → result=0

3.3.4 Startup Operation → চাকরির ক্ষেত্রে লাগবে
self-study

Chapter-4

```
a DB 'A'
a DB ?
```

Naming field
length 1-31 ?, ., @, $, _, %
. → সবসময় ব্যবহার

AB.CDX
.ABCDX

CamScanner

Operation field

NOP → No operand

INC (AX) → Increment
↳ Operand

ADD AX, BX
Operand

Binary	Decimal	Hexadecimal
10b/10B	10	10h/10H

DB → Define byte → 1 byte
DW → Define word → 2 byte
DD → Double word → 2W
DQ → Define Quard word → 4W
DT → Define Ten byte → 10 B

Array

Symbol — Address

B_ARRAY DB 10H, 20H, 30H

Symbol	Address	content
B_ARRAY	200h	10h
B_ARRAY+1	201h	20h
B_ARRAY+2	202h	30h

Characterstring

LETTER DB 'ABC' → ASCII values
||
LETTER DB 41H, 42H, 43H

EQU → equal

name EQU → constant ও value 0AH

assign করার সময় carriage return ↳ newline ও first space

new line (line feed) → LF EQU 0AH

source এর content unchanged থাকবে, কিন্তু dest. এ move করবে

MOV → Move → destination, source

XCHG → Exchange → destination, source

&

both change হবে

MOV AX, Mem → memory location এর content AX এ move

MOV Mem, AX → AX এর মান mem loc. এ move

Mov Mem, Mem ✗.

সবসময় register এর through move করাতে হবে.

9.2 table

GR → GR

GR → ML

ML → GR

ML → ML ✗

GR= General Register

ML = Memory Location

ADD

SUB

INC } only destination, no source

DEC }

9.3

ML to ML তে ADD বা SUB করা যাবে না

NEG → Negation → 2's complement

** A = B - 2A tny

MOV AX, B

SUB AX, A

SUB AX, A

MOV A, AX

SMALL → 1 Code segment, 1 Data segment.

MEDIUM

COMPACT

LARGE

Memory models

Not mandatory

DATA → global variable declaration এর জন্য

WORDT DW 2

MSG DW 'Hello world'

• STACK size.
100 H.

• CODE

```
MAIN PROC
    ;
    ;
    ;
    ;

  MAIN ENDP
    ;
  Subfunction/ user defined function
    ;
    ;
END MAIN.
```

Input

MOV AH, 1 → single char input
INT 21H (char → AL → ভেতর যাবে)

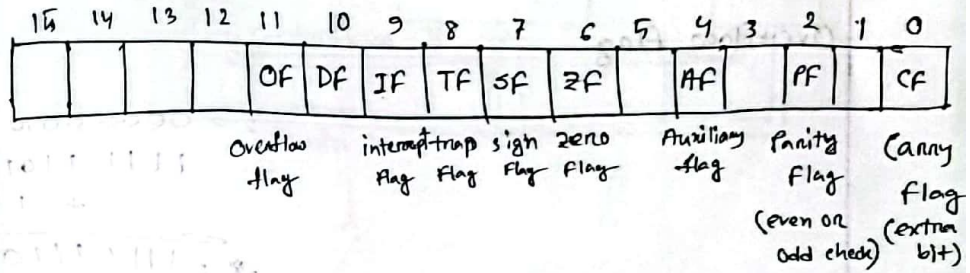
Output

MOV AH, 2 q·g
output → DL .
INT 21H

Lecture-5

Chapter-5

The Processor Status and FLAG Registers



operation result = 0 হলে zero flag = 1

carry in
1 1
1 1
100

carry out
1
1
10

Addition ও

Parity flag → FFFE
1111 1111 | 1111 1110
HB | LB
PF = 0

Auxiliary carry flag (Af):

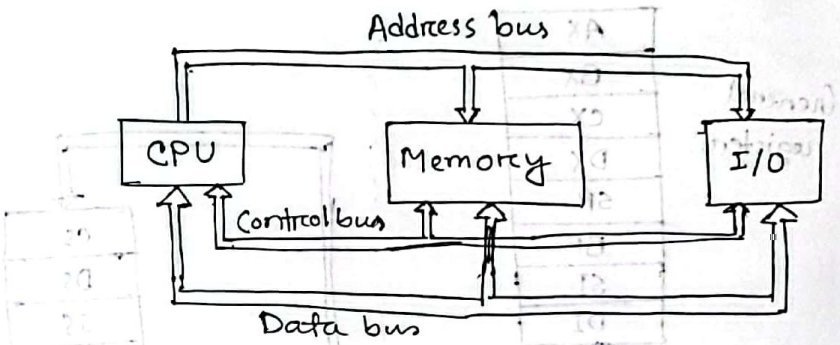
Sign flag

Overflow flag

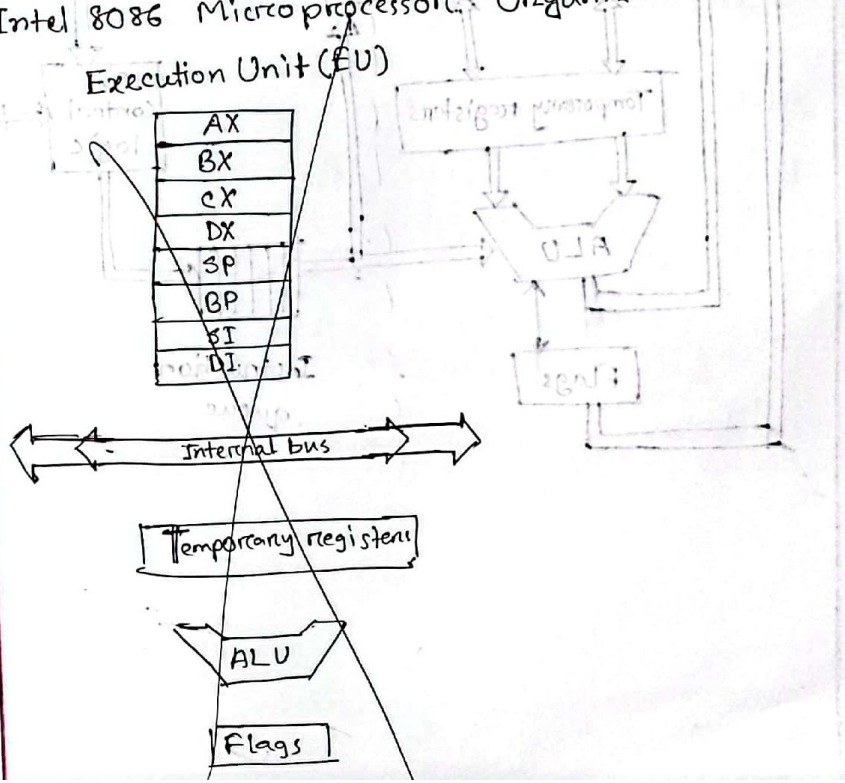
2 = 0000 0010
1111 1101
+ 1
-2 = 1111 1110

CT-1

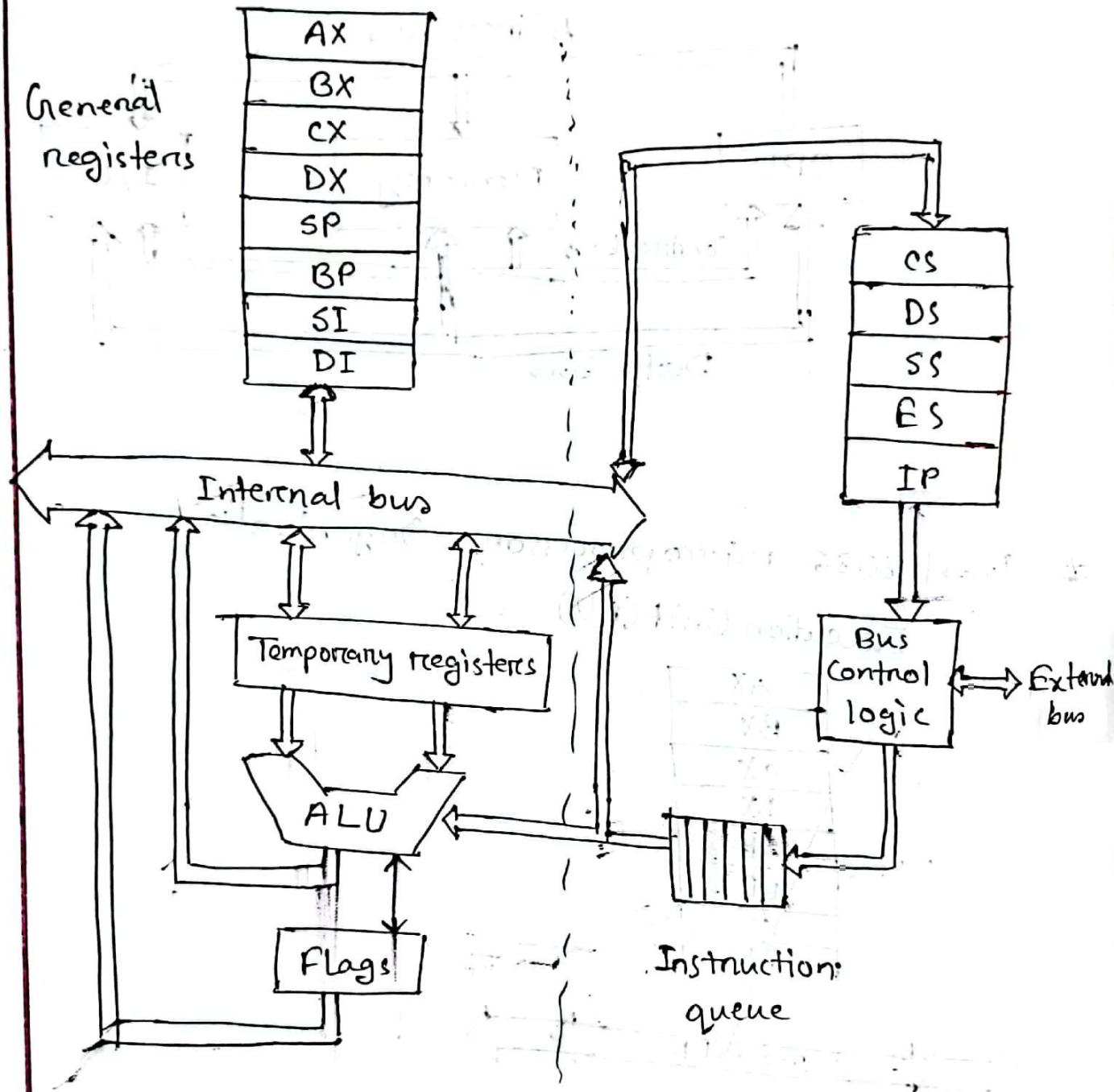
Bus Connections of a Microcomputer



Intel 8086 Microprocessor Organization



Execution Unit (EU)



35h = 00110101

0Fh = 00001111

00000101

XOR - specific bit complement

NOT - 16 bit complement.

AND - destination change

Test - no change.

Assembly CT-2

Tuesday - 4টা

Chapter-7

SHL AL,1

AL 1 0 1 1 1 0 1

CF=1 0 1 1 1 0 1 0 , Leftside থেকে CF এ যাবে

আর Right side এ যেটা হবে 0 add হবে.

Multiply by 2 = Leftshift -

AL = 5 = 00000101 b

Left shift = 00001010 b = 10

k times shift = 2^k multiply.

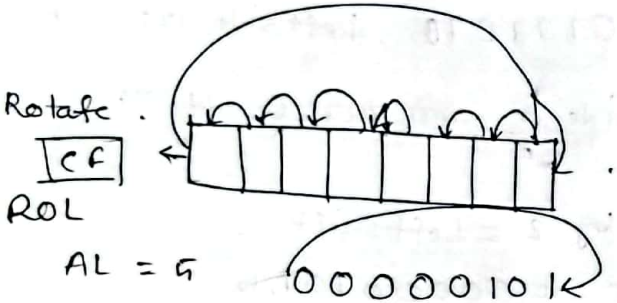
BL = 80h CL = 2

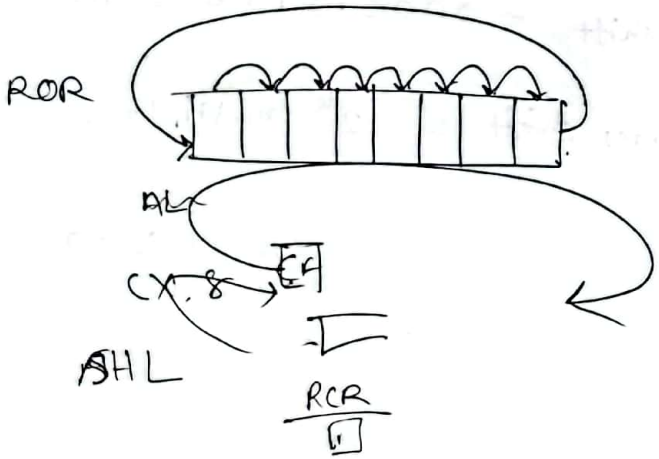
SHR

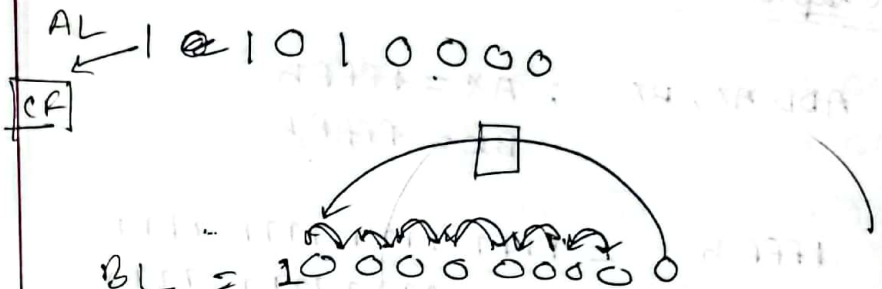
AL = 5 = 00000101

2 = 00000010 CF=1

floor value







00000001

11000000

01100000

10110000

01011000

00101100

00010110

00001011

CT-2

Chapter-5

5.1 ADD AX, BX ; AX = ffffh
BX = ffffh.

ffffh = 1111 1111 1111 1111
ffffh = 1111 1111 1111 1111
---
1 1111 1111 1111 1110

SF = 1
ZF = 0
PF = 0
CF = 1.
OF = 0

AL = 80h, BL = 80h

80h = 1000 0000
80h = + 1000 0000
---
1 0000 0000

SF = 0
PF = 1
ZF = 1.
CF = 1.
OF = 1.

5.3 SUB AX, BX .

AX = 8000h = 1000 0000 0000 0000
BX = 0001h = 0000 0000 0000 0001

1111 1111 1111 1110
+ 1

1111 1111 1111 1111

7fffh = 1 0111 1111 1111 1111

SF = 0 -A-B = -(-ve)
PF = 1
ZF = 0
CF = 0
OF = 1

5.4 INC AL , AL = ffh = 1111 1111
+ 0000 0001
---
1 0000 0000

SF = 0 -A+B = +ve
PF = 1.
ZF = 1.
CF = 0 [unchanged]
OF = 0

5.6 NEG AX .

AX = 8000h = 1000 0000 0000 0000

1's com. 0111 1111 1111 1111

+ 0000 0000 0000 0001

-8000h = 1000 0000 0000 0000

SF = 1 .

PF = 1

ZF = 0

CF = 1 (always unless result=0) .

OF = 1 .

Exercise

1. a) ADD AX, BX ; AX = 7FFFh , BX = 0001h .

7FFFh = 0111 1111 1111 1111

0001h = 0000 0000 0000 0001

1000 0000 0000 0000

SF = 1

ZF = 0

PF = 1

CF = 0

OF = 1 .

b) SUB AL, BL

01 h = 0000 0001

FF h = 1111 1111

= 1 00 0000 10

SF = 0

PF = 0

ZF = 0

CF = 1 .

OF = 0 .

c) DEC AL

AL = 00h = 0000 0000

- 0000 0001

FFh = 1 1111 1111

SF = 1 A-B =

ZF = 0

PF = 1

CF = 0 [unchanged]

OF = 0 [same sign overflow হচ্ছে না]

d) NEG AL.

7Fh = 0111 1111

1's comp. 1000 0000

+ 0000 0001

81h = 1000 0001

SF = 1. -A+B =

ZF = 0

PF = 1

CF = 1. [always 1 unless result = 0]

OF = 0

e. XCHG AX, BX [does not affect]

f. ADD AL, BL

AL = 80h = 1000 0000

BL = FFh = 1111 1111

7Fh = 1 0111 1111

SF = 0

ZF = 0

PF = 0

CF = 1

OF = 1.

g. SUB AX, BX

AX = 0000h = 0000 0000 0000 0000

BX = 8000h = 1000 0000 0000 0000

8000h = (1) 1000 0000 0000 0000

SF = 1 A - -B = A+B

PF = 1 = (-ve)

ZF = 0

CF = 1

OF = 1.

h) NEG AX

AX = 0001h = 0000 0000 0000 0001

1's comp = 1111 1111 1111 1110

+ 0000 0000 0000 0001

FFFFh = 1111 1111 1111 1111

SF = 1

ZF = 0

PF = 1.

CF = 1

OF = 0

2. a)

carry out = 0

carry in = 1.

AX = 01** **** **** ****

BX = 01** **** **** ****

10** **** **** ****

Ex:

AX = 7FA0h = 0111 1111 1010 0000

BX = 600Dh = 0110 0000 0000 1101

DFADh = 1101 1111 1010 1101

Cin XOR Cout = 1 XOR 0 = 1 = OF

b)

Cout = 1

Cin = 0

AX = 10xx xxxx xxxx xxxx

BX = 10xx xxxx xxxx xxxx

100

AX = 8DE4h = 1000 1101 1110 0100

BX = B216h = 1011 0010 0001 0110

3FFAh = 1 0011 1111 1111 1010

Cout XOR Cin = 1 XOR 0 = 1 = OF

3. a)

512Ch = 0101 0001 0010 1100

+4185h = 0100 0001 1000 0101

92B1h = 1001 0010 1011 0001

SF = 1.

ZF = 0

PF = 1

CF = 0

OF = 1.

signed OF = 1 unsigned CF = 0

b) FE12h = 1111 1110 0001 0010
+1ACBh = 0001 1010 1100 1011
18DDh = 1 0001 1000 1101 1101
Sf = 0, Pf = 1, Zf = 0
unsigned Cf = 1 signed Of = 0

c) E1E4h = 1110 0001 1110 0100
+DAB3h = 1101 1010 1011 0011
BC97h = 1 1011 1100 1001 0111
Sf = 1 Pf = 0 Zf = 0
unsigned Cf = 1 signed Of = 0

d) 7132h = 0111 0001 0011 0010
+7000h = 0111 0000 0000 0000
E132h = 1110 0001 0011 0010
Sf = 1, Pf = 0, Zf = 0
unsigned Cf = 0, signed Of = 1.

e) 6389h = 0110 0011 1000 1001
1176h = 0001 0001 0111 0110
74FFh = 0111 0100 1111 1111
Sf = 0, Pf = 1, Zf = 0
unsigned Cf = 0, signed Of = 0.

4. a) 2143h = 0010 0001 0100 0011
- 1986h = 0001 1001 1000 0110
07BDh = 0000 0111 1011 1101
Sf = 0, Zf = 0, Pf = 1, Cf = 0, Of = 0

b) 81FEh = 1000 0001 1111 1110
-1986h = 0001 1001 1000 0110
6878h = 0110 1000 0111 1000
Sf = 0, Zf = 0, Pf = 1
unsigned Cf = 0, signed Of = 1 [-A - B = +ve]

c) 19BCh = 0001 1001 1011 1100
-81FEh = 1000 0001 1111 1110
97BE = 1001 0111 1011 1110

SF = 1 , ZF = 0 , PF = 1

unsigned CF = 1 signed OF = 1.

[A-(-B) = A+B ve(-ve)]

d) 0002h = 0000 0000 0000 0010
-FE0Fh = 1111 1110 0000 1111
01F3h = 0000 0001 1111 0011

SF = 0 , ZF = 0 , PF = 1.

unsigned CF = 1 , signed OF = 0

A-(-B) = A+B = (+ve)

e) 8BCDh = 1000 1011 1100 1101
-71ABh = 0111 0001 1010 1011
1A22h = 0001 1010 0010 0010

SF = 0 , ZF = 0 , PF = 1

unsigned CF = 0 , signed OF = 1 [-A-B ±ve]

Lecture

chapter-8

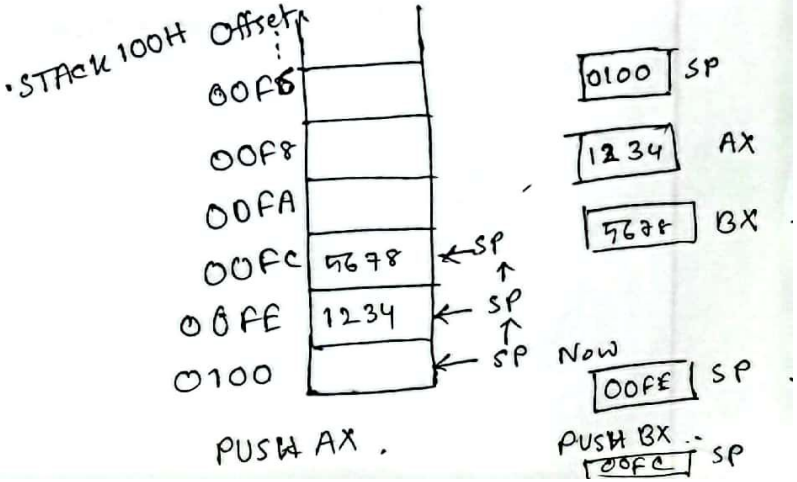
Stack

PUSH source → source এর value stack a push হবে।

POP destination →

Push →
1. SP → decrease হবে
2. Push

POP →
1. POP
2. SP → increase হবে।



High byte FD

Low byte FF.

00FC	SP
FFFF	CX
0001	DX

POP CX

5678	CX
00FE	SP

POP DX

0134	DX
0100	SP

Procedure declaration

```
name  PROC type
-  -  -  -
-  -  -
```

RET (mandatory)

```
name ENDP.
```

```
A * B
product = 0
REPEAT.
    If lsb of B = 1
then product = product + A.
END-If.
```

Lecture

Chapter-9

Multiplication & Division Instructions.

10000000 * 11111111 = 0111111100000000

Unsigned

128 * 255 = 32640

signed

-128 * -1 = 128.

= 0000000010000000

MUL – unsigned

IMUL – signed

Integer multiplication

8 bit * 8 bit = 16 bit

16 bit * 16 bit = 32 bit

→ 2 bit register লাগবে.

MUL source

IMUL source

Byte form → AL , Result → AX — AH upper
. AL Lower.

Word form → AX Result → DX – upper half

AX – Lower half

carry & overflow flag = 0 ; Result এর upper half 0 হবে.

otherwise , CF/OF = 1.

IMUL এর ক্ষেত্রে

CF/OF → 0 → এরও upper half, lower half sign extension হবে, lower half এর sign bit upper half এর সবগুলো bit হবে

→ 1 [otherwise] Ex: Lower half এর sign bit 1 হলে, upper half এর সবগুলো bit 1 হবে,

AX = 1

BX = ffff

MUL BX 65535 0000ffff

	DX	AX
CF/OF = 0	0000	ffff

IMUL BX -1 ffffffff

	DX	AX
CF/OF = 0	ffff	ffff

Ex: [illegible].

A = 5 X A - 12 X B

```
MOV  AX, 5    ; AX = 5
IMUL A        ; AX = 5 X A
MOV  AX, A    ; A = 5XA
MOV  AX, 12   ; AX = 12
IMUL B        ; AX = 12XB
```

```
SUB  A, AX ;  A = A - AX
              = 5XA - 12XB
```

Factorial.

9.7

DIV → unsigned
IDIV → signed.

DIV divisor.

Byte form
dividened → 16 bit → AX
divisor → 8 bit

quotient → AL
remainder → AH

```
divisor) AX ( AL
         ----
          AH.
```

Word form
dividened → 32 bit → DX, AX.
source/divisor → 16 bit → [illegible].

quotient → AX.
remainder → DX.

```
source) DX;AX ( AX
                --
                DX.
```

9.3

DX = 0000h

AX = 0005h

BX = 0002h.

	decimal quotient	decimal remainder	AX	DX
DIV	2	1	0002	0001
IDIV	2	1	0002	0001

DX = 0000h

AX = 0005h

BX = FFFEh = 65534 = −2

	decimal quotient	decimal remainder	AX	DX
DIV	0	5	0000	0005
IDIV	−2	1	FFFE	0001

Lecture

9.4

Sign Extension → self study.

Decimal I/O.

Decimal Output.

123

−123.

AX > 0, direct print

AX < 0, '−' print then NEG AX.

AX = −1.

NEG AX ; AX = 1.

If AX < 0, Then print '−'

Replace A ← 2's compliment (Neg)

End if.

Get digit in AX's decimal representation

Convert digit → character & print.

Decimal representation

24618

10 দিয়ে ভাগ করে কাছ.

24618 /10 , Q = 2461 , R = 8

2461/10 , Q = 246 , R = 1

246/10 , Q = 24 , R = 6

24/10 , Q = 2 , R = 4

2/10 , Q = 0 , R = 2

Reverse

count count=0

Repeat

divide Q by 10.

push reminder into stack

count = count +1 ; Pop কতবার লাগবে তার জন্য

until Q = 0.

for count times do

Pop digit from stack.

digit → ch

output → ch

End for.

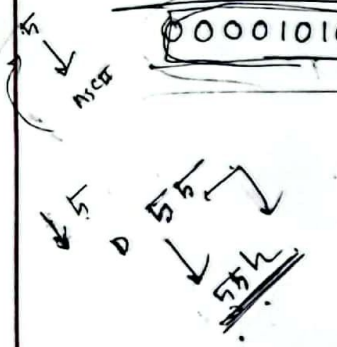
9.1

Decimal Input.

01000001 = 41h → A

00110111 = 37h

00001010 = A



00001001
× 0000 1111
1001

9

A — 10 — 1010
B — 11 — 1011
C — 12 — 1100
D — 13 — 1101
E — 14 — 1110
F — 15 = 1111

42h = 1000010

Chapter-10

Arrays & Addressing Modes.

↳ Data structure to store data.

MSG DB 'ABCD'

W DW 20, 25, 35, 40

Offset Address	Symbol	Decimal
0200h	W	20
0202h	W+2h	25
0204h	W+4h	35
0206h	W+6h	40

repeat count duplicate operator → DUP(value)

G DW 100
(array name) (word size (2 byte)) (elements)

DUP(0) (initially 0)
DUP(?) (no initialization)

Line DB 5, 4, 3 DUP (2, 3 DUP(0), 1)
(byte size (1 byte)) (1st element) (2nd) (কতবার duplicate) 3X (2, 0,0,0 1)

Line DB 5, 4, 2,0,0,0,1, 2,0,0,0,1, 2,0,0,0,1

position	location
	A
1	A = 1 x S
2	A = 2 x S
3	
.	
.	
.	
N	A = (N-1) x S .

Byte

10th position element ⟷ 24th position element

B+9 B+24

Word

W+9X2 W+24X2

W+18 W+48 .

MOV AX, W+18

XCHG W+48, AX

MOV W+18, AX .

Addressing Mode

- Register
  - operand → (Register)
- Immediate
  - eg. (value)
  - operand → constant
- Direct .
  - (memory location) .
  - operand → variable / memory location

MOV AX, 0

↓ ↓

Register Immediate .

MOV VAR, AX .

↓ ↓

Direct register .

Addressing mode :

I) Register Indirect → Register কে pointer হিসেবে use করা হয়

[register] → Address এর content

↓

BX, SI, DI, BP .

DS SS .

II) Based

III) Indexed

IV) Based Indexed .

SI = 0100h

0100h → 1234

MOV AX, [SI] → 1234

MOV AX, SI → 0100h

W DW 10, 20, 30, ..., 100

XOR AX, AX
LEA SI, W.
MOV CX, 10
REPEAT:
ADD AX [SI]
ADD SI, 2.
Loop REPEAT.

Ex: 10 → Reverse.

Lecture

Based & Indexed.

displacement
- variable ; A
- constant ; -2
- offset address ± constant; A+4.

* Representation

[reg + dis]

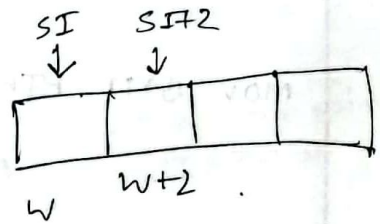
[dis + reg]

[reg] + displacement

displacement + [reg]

dis [reg]

BX, BP (based) . SI, DI (Indexed) → Addressing mode এর addressing করার জন্য



W, BX = 4.

MOV AX, W[BX]

W + BX
W + 4

W+2
W+4 → 3rd element AX এর মধ্যে move হবে

mov AX, [SI+2] → end element AX এর মধ্যে move হবে.
W+2.

```
W DW  10,20,30,40,50,60,70,80,90,100
   XOR AX,AX
   XOR BX,BX
   MOV CX,10
ADD_ADD AX, W[BX]
   ADD BX,2
   LOOP ADD_
```

PTR operator

Word instruction: MOV AX,5 ; 5 → 16 bit হয়

Byte instruction: MOV AL,5 ; 5 → 8 bit হয়

MOV [BX], 5 ; Byte/Word both হওয়াতে so illegal.

MOV BYTE PTR [BX],5 ; source, destination byte হয়ে, Address indicate করছে.

MOV WORD PTR [BX],5 ; word.

```
MSG . DB 'This is a msg'
```
T 5

MOV [BX], BYTE MSG; Byte হয়ে এসে came MSG আগে যেভাবে declare করা, source type বলে দেয়া আছে,

Byte instruction ↑

one dimensional array

MOV BYTE PTR [BX], MSG

LEA SI, MSG ; first letter point করে আছে.

MOV MSG [SI], 'T'.

MOV MSG [SI+3], '5'

SP use করলে অবশ্যই BP use করতে হবে.

AX
BX
CX

MOV BP, SP ← Stack top point করার জন্য BP use

MOV AX, [BP]

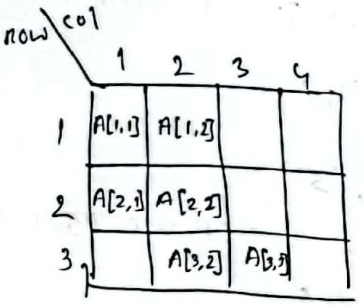
MOV BX, [BP+2]

MOV CX, [BP+4]

10.3 ex.

Array element sort

Two dimensional array তে access করার জন্য Based Indexed addressing mode.



Row major order → A[row, col]

Col major order → A[col, row].

Row major order

B DW 10, 20, 30, 40
DW 50, 60, 70, 80 [3x4] array
DW 90, 100, 110, 120

Column major order

B DW 10, 50, 90
DW 20, 60, 100
DW 30, 70, 110
DW 40, 80, 120

Array indexing

A DW ; M X N

Row i A[i, 1] → A + (i-1)*N*2

col j, A[1, j] → A + (j-1)*2

(word size ↓ (Byte size হলে 1))

Based- Indexed Addressing representation

V [b_rr] [i_rr]

[b_rr + i_rr + V + C]

C [b_rr + i_rr + V]

V [b_rr + i_rr + C]

107 A DW 5X7.

1) clean row 3.

2) clean col 4.

1) A+(3-1) * 7 * 2 = A+28

2) MOV CX, 7.

MOV BX, 28

XOR SI, SI

CLEAR_:

MOV A[BX][SI], 0

ADD SI, 2

loop CLEAR_

2) A + (4-1) * 2 = A+6

MOV A[BX][SI], 0

ADD BX, 2

loop

*** article 10 c Avenage in test score

CT-3

chapter-6

1. a)
```
CMP AX,0
JNL / JGE END_IF
MOV BX,-1
END_IF:
```

b)
```
CMP AL<0
JNL/JGE ELSE_
    MOV AH,FFh
    JMP END_IF
ELSE_:
    MOV AH,0
END_IF:
```

c)
```
CMP DL, "A"
JL END_IF
CMP DL, "Z"
JG END_IF
MOV AH,2
INT 21H
END_IF:
```

d)
```
CMP AX,BX
JNL/JGE END_IF
    CMP BX,CX
JNL/JGE ELSE_
    MOV AX,0
    JMP END_IF
ELSE_:
    MOV BX,0
END_IF:
```

e)
```
CMP AX,BX
JL THEN_
CMP BX,CX
JL THEN_
MOV DX,1
JMP END_IF
THEN_:
    MOV DX,0
END_IF:
```

f)
```
CMP AX,BX
JNL ELSE_
MOV AX,0
JMP END_IF
ELSE_:
    CMP BX,CX
    JNL ELSE2_
    MOV BX,0
    JMP END_IF
ELSE2_:
    MOV CX,0
```

2.
```
MOV AH,1
INT 21H

CMP AL,'A'
JE CAR_RET
CMP AL,'B'
JE LINE_FEED
MOV AH,4CH
INT 21H
JMP END_CASE
CAR_RET:
    MOV AH,2
    MOV DL,0DH
    INT 21H
    JMP END_CASE
LINE_FEED:
    MOV AH,2
    MOV DL,0AH
    INT 21H
END_CASE:
```

CamScanner

3. a) (148-1)/3 = 49

```
MOV CX,49
MOV AX,1
MOV BX,1

LOOP1:
    ADD BX,3
    ADD AX,BX
    LOOP LOOP1
```

b) (100-5)/5 = 19

```
MOV CX,19
MOV AX,100
MOV BX,100
LOOP1:
    SUB BX,5
    ADD AX,BX
    LOOP LOOP1.
```

4. a)

```
MOV CX,50
MOV BX,1
MOV AX,1

L1:
    ADD AX,4
    ADD DX,AX
    LOOP L1
```

b)

```
MOV AH,1
INT 21H
MOV AH,2
MOV DL,0AH
INT 21H
MOV DL,0DH
INT 21H
MOV DL,AL
MOV CX,80
DISPLAY:
    INT 21H
    LOOP DISPLAY
```

c)

```
MOV CX,5
MOV AH,7
L1: INT 21H
    LOOP L1

MOV DL,'X'
MOV CX,5
MOV AH,2
L2: INT 21H
    LOOP L2
```

5.

```
MOV AX,0
WHILE_:
    CMP AX,BX ;AX=dividened
    JL END-     Bx=divisor
    INC CX      Cx=quotient
    SUB AX,BX
    JMP WHILE-
END_:
```

6. MOV CX, 0

L1: ADD CX, AX ; CX=product, AX=M, BX=N.
DEC BX
JNZ L1.

7. a)
```
MOV AH, 1
MOV CX, 80
L1: INT 21H
CMP AL, 20H
LOOPE L1.
```

b)
```
MOV AH, 1
MOV CX, 80
L2: INT 21H.
CMP AL, 0DH
LOOPNE L2.
```

chapter-7

1. a) 10001011b, b) 11111001b, c) 10100110b
d) 10100001b

2. a) AAAAh = 1010 1010 1010 1010 b.

Clearing even number of bits of AX.

AX = 01_10_10_11_01_01_10_00

AND AAAAh = 10_10_10_10_10_10_10_10

0010 1010 0000 1000

MOV AX, AAAAh.

b) 81h = 10000001b. set msb and lsb of BL.

OR BL, 81h

c) Complement msb of DX

8000h = 1000 0000 0000 0000

XOR DX, 8000h

d) NOT WORD1 → complement word variable.

XOR WORD1, ffffh.

3. a) TEST AX, FFFFh.

b) TEST BX, 0001h

c) TEST DX, 8000h

d) TEST DX, 8000h

e) TEST BL, FFh

AL = 11001011b = CBh

4. a) 10010110b = 96h

b) 01100101b = 65h

c) 10010111b

00101111b = 2Fh

d) 11100101b

11110010b

01111001b = 79h

e) 11100101b

11110010b = F2h

f) 10010111b = 97h

g) 11100101b, CF=1

11110010b, CF=1

11111001b, CF=0

= F9h

5. a) Double the value of byte variable B5

SHL B5, 1.

b) Multiply AL by 8.

MOV CL, 3
SHL AL, CL

SHL AL, 3

c) Divide 32142 by 4 ; AX = quotient

MOV AX, 32142
SHR AX, 2

d) Divide -2145 by 16

MOV BX, -2145
SAR BX, 4.

6. a) ADD AL, 30H // convert value to decimal
OR AL, 30H.

b) OR DL, 20H // Upper case to lower case

7. a) MOV DL, BL
MOV CL, 3
SHL BL, CL.
SHL DL, 1
ADD BL, DL.

b) MOV CL, 3
MOV AH, 0
ROR AX, CL.

BL = 0011 1011 ⇒ 1011 1000
10d = 0000 1010
0000 1010
0001 0100
1101 1100
0111 0110
0110

AX = 0101 1010 0111 0110
5 A 7 6
AH → 0000 0000 0111 0110
1 → 0000 0000 0011 1011
2 → 1000 0000 0001 1101
3 → 1100 0000 0000 1110
C 0
12

BL = 0101 0111 = 87d x 10d
DL = 0101 0111

1 → 10101110
2 → 01011100
3 → 10111000
OR 10101110
10111110

BL = 0000 0111
DL = 0000 0111
1 → 00001110
2 → 00011100
3 → 00111000
00111110

AL = 8CH

= 1000 1700 b.

ROR AL, 20

1 → 01000110
2 → 00100011
3 → 10010001
4 → 11001000
5 → 01100100
6 → 00110010
7 → 00011001
8 → 10001100

8 X 2 + 4 = 20

AL = 11001000 = C8H.

ii) SAR AL, CL , CL = 3

8CH = 1000 1100

1 → 1100 0110
2 → 1110 0011
3 → 1111 0001

AL = F1H.

Lecture

*** XLAT – chapter 10 → self study

ex:

Chapter-11

String operation

DF → Flag register.

string operation তৈরিতে more করবে DF অনুযায়ী inc বা dec হবে।

DF = 0 , → left to right , SI+1 , DI+1

DF = 1 , ← right to left SI-1 , DI-1

SI, DI

Offset address 0200H 0201H 0202H

STRING1 DB 'ABCDE' ← 0204H, 0203H

CLD → DF = 0 clear DF

STD → DF = 1 Set DF.

MOV SB → MOV String Byte.

↓ 1 byte more

Non operand instructions → CLD, STD

SI, DI → offset (moving a string)

Segment: DS: SI, ES: DI

STR1 DB 'HELLO'

STR2 DB 5 DUP (?) . ← 5টা letter

MOV AX, @DATA

MOV DS, AX ← initialize.

MOV ES, AX

LEA SI, STR1 point করে দেওয়া

LEA DI, STR2

~~CLD~~ ← left to right

MOV CX, 5

REP: ~~MOV SB~~

CLD

MOVSB

LOOP REP

MOVSW → word size string

SI+2 , SI+4 , SI+6 .

SI-2 , SI-4 , SI-4 .

CT-4 → chapter 8, chapter-9

decimal

Lecture

String store – character need.

Dest. — string → ES:DI.

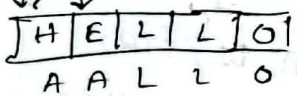
Source — AL , AX

Byte Word

```
STR1  'HELLO'
LEA DI, STR1
CLD
MOV AL, 'A'
STO SB
STOSB .
```

store

DI DI



A directly DI দিয়ে store হয় না, Register AL দিয়ে store করতে হয়

Load String

Source → string

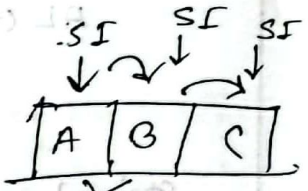
Dest. → Register.

DS:SI → AL (LODSB)

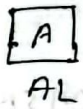
→ AX (LODSW)

DF

```
STR1 DB 'ABC'
CLD
```



Register AL এ Load করতে হবে.



Read-এ ডাটা store এ করতে হয়

Display-এ ডাটা load এ করতে হবে.

Display a character string.

```
FOR count_time DO
    LOAD string ch into AL
    MOV DL, AL.
    output ch.
```

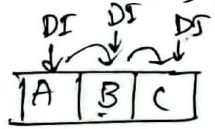
Scan String

SCA SB → non operand
SCA SW.

target byte → AL - register → character রাখা

AL (–) ES:DI → string যেটা scan করবে.

→ ZF affected হয়.



target string এ scan করে যেখানে যেয়ে B আছে.

→ B পাইলে AL-ES:DI করে 0 হয়ে যাবে।

SCASB → A - B → ZF = 0

SCASB → B - B → ZF = 1

string বা target change হবে না, শুধু flag affected হবে.

REPNZ → Repeat until not zero

REP NZ SCASB → যতক্ষণ পর্যন্ত target এর string এর control 0 না আসে ততক্ষণ scan হবে.

REP NE →

*** # for counting vowels and consonants.

CT-4

Chapter-8

Initially SP → 0100h.

PUSH source → must be 16 bit register or memory word.

1. SP decreased by 2.
2. A copy of source content moved to SS:SP
   source unchanged.

PUSHF → no operands
→ pushes the contents of the FLAGS registers onto the stacke

POP destination → must be 16 bit reg. or mem word
(except IP)

1. The content of SS:SP moved to destination
2. SP increased by 2 (destination changed)

Exercises

1. a) 0100h.

b) 100h byte for stack contains

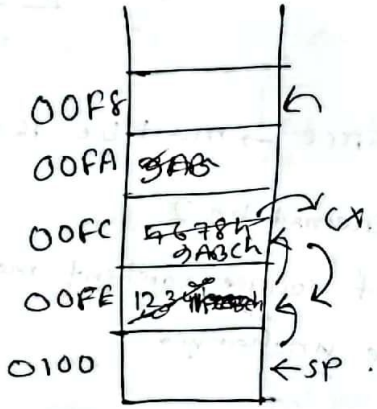
$100h/2 = 80h$ word.

2. AX = 1234h

BX = 5678h

CX = 9ABCh.

SP = 100h.



XCHG AX, CX.

POP CX = 5678h. AX = 9ABCh.

PUSH AX = 9ABCh

BX = 9ABCh.

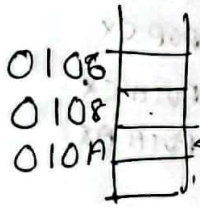
SP = 00FE

4. CALL PROC1

~~08FD:0003h~~ MOV AX, BX

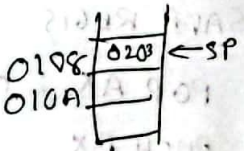
08FD: 0203h.

08FD:0300h PROC1

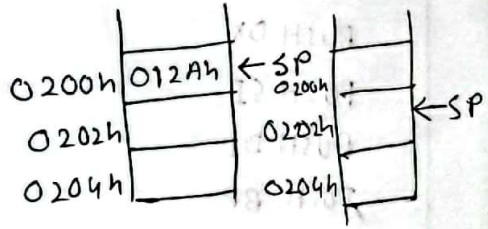


SP = 0108h.

IP = 0300h.



5. SP = 0200h,



a) After RET.

IP = 012Ah

SP = 0202h

b) RET 4

RET POP_VALUE

IP = 012Ah

SP = 0206h

SP = 0202h + 4 = 0206h

6. a)
```
POP AX
PUSH AX
```

b)
```
POP AX
POP CX
PUSH CX
PUSH AX
```

c)
```
POP AX
POP BX
PUSH AX
PUSH BX
```

7.
```
SAVE_REGS PROC
  POP AX ; return address
  PUSH BX
  PUSH CX
  PUSH DX
  PUSH SI
  PUSH DI
  PUSH BP
  PUSH DS
  PUSH ES
  PUSH AX ; return address at top.
  RET
SAVE_REGS ENDP.
```

b)
```
RESTORE_REGS PROC
  POP AX ; save return address
  POP ES
  POP DS
  POP BP
  POP DI
  POP SI
  POP DX
  POP CX
  POP BX
  PUSH AX ; top of stack is return address
  RET
RESTORE_REGS ENDP.
```

CamScanner

Chapter-9

MUL source

IMUL source

Byte form

one number → source → byte register or memory byte.

other number → AL.

product → AX

Word form

one number → source

other number → AX

product → DX:AX

MSB : LSB.

for positive numbers, MUL & IMUL are same.

After MUL,

00000000 11010110 → CF/OF = 0

10101010 11011010 → CF/OF = 1

After IMUL,

11111111 11010110 → CF/OF = 0

9.1 AX = 1, BX = FFFFh = 65535 = -1

Inst.	Dec. product	Hex product	DX	AX	CF/OF
MUL BX	65535	0000FFFF	0000	FFFF	0
IMUL BX	-1	FFFFFFFF	FFFF	FFFF	0

9.2 AX = FFFFh, BX = FFFFh

Inst.	Dec Prod	Hex Prod	DX	AX	CF/OF
MUL BX	4294836225	FFFE0001	FFFE	0001	1
IMUL BX	1	00000001	0000	0001	0

16 | 4294836225
16 | 268427264 — 1
16 | 16776704 — 0
16 | 1048544 — 0
16 | 65534 — 0
16 | 4095 — 14 → E
16 | 255 — 15 → F
16 | 15 — 15 → F
0 — 15 → F

9.3 AX = 0FFFh

Inst.	Decimal Prod.	Hex Prod	DX	AX.	CF/OF
MUL AX	16769025	00FFE001	00FF	E001	1
IMUL AX	16769025	00FFE001	00FF	E001	1

9.4 AX = 0100h , CX = FFFFh.
= 256 = 65535
= -1.

Inst	Dec Prod	Hex Prod	DX	AX	CF/O
MUL CX	16776960	00FFFF00	00FF	FF00	1
IMUL CX	-256	FFFFFF00	FFFF	FF00	0

9.5 AL = 80h , BL = FFh
= 128 = 255
10000000 = -128 = -1.

Inst	Dec Prod	Hex Prod	AH	AL	CF/OF
MULBL	32640	7F80	7F	80	1
IMULBL	128	0080	00	80	1

CamScanner

9.6 $A = 5 \times A - 12 \times B$

```
MOV AX, 5     ; AX = 5
IMUL A        ; AX = 5 x A
MOV A, AX     ; A = 5 x A
MOV AX, 12    ; AX = 12
IMUL B        ; AX = 12 x B
SUB A, AX     ; A = 5 x A - 12 x B
```

9.7

```
FACTORIAL PROC
    MOV AX, 1
TOP:
    MUL CX
    LOOP TOP
    RET
FACTORIAL ENDP
```

#

DIV divisor

IDIV divisor.

quotient & remainder same size as divisor

Byte form

divisor → 8 bit reg or mem

dividend → 16 bit → AX.

quotient → 8 bit → AL

remainder → 8 bit → AH

Word form

divisor → 16 bit reg or mem

dividend → 32 bit → DX:AX

quotient → 16 bit → AX

remainder → 16 bit → DX

for signed → remainder and dividend same sign

dividend & divisor → positive then DIV & IDIV same

Flags undefined.

CamScanner

9.8 DX = 0000h , AX = 0005h.

BX = 0002 h.

Inst	Dec. quo.	Dec. rem.	AX	DX
DIV BX	~~0002~~ 2	1	0002	0001
IDIV BX	2	1	0002	0001

9.9 DX = 0000h , AX = 0005h
= 5

BX = FFFEh
= -2

Inst	Dec. quo.	Dec. rem.	AX	DX
DIV BX	0	5	0000	0005
IDIV BX	-2	1	FFFE	0001

9.10 DX = FFFFh = -1, AX = FFFBh = -5
= 65535 = 65531
BX = 0002h

Inst	Dec. quo.	Dec. rem.	AX	DX
DIV BX	-2	-1	FFFE	FFFF
IDIV BX	DIVIDE OVERFLOW			

9.11 AX = 00FBh = 251

BL = FFh = 255

Inst	Dec. quo.	Dec. rem.	AL	AH
DIV BL	0	251	~~0000~~ 00	FB.
IDIV BL	DIVIDE OVERFLOW.			

9.12 -1250 by 7

MOV AX, 1250
CWD
MOV BX, 7
IDIV BX.

9.13 XBYTE by -7.

MOV AL, XBYTE
CBW
MOV BL, -7
IDIV BL

Exercise

1. a) AX = 0008h , BX = 0003h

b) AX = 00FFh , BX = 1000h
= 255. = 4096

c) AX = 0005h CX = FFFFh = 65535.
= -1

d) AX = 8000h = -32768 , WORD1 = FFFFh. = -1

Inst	Dec prod	Hex prod	DX	AX	CF/OF
a) MUL BX	24	00000018	0000	0018	0
b) MUL BX	1044480	000FF000	000F	F000	1
c) IMUL CX	-5	FFFF FFFB	FFFF	FFFB	0
d) IMUL WORD1	32768	0000 8000	0000	8000	1
e) Illegal					

2.

Inst			AX	CF/OF
a) MUL BL	2736	0AB0	0AB0	1
b) IMUL BL				
c) MUL AH	(01 X AB) 171	00AB	00AB	0
d)				

3. a)

DX = 0003h , AX = 0007h , BX = 0002h

Dec. quo.	Dec. rem.	AX	DX
~~0003~~ 3	~~00~~ 1	0003	0001

b) DX = 0000h , AX = FFFEh = 65534 BX = 0010h = 16

Dec quo	Dec rem	AX	DX
4095	14	0FFF	000E

c) DX = FFFFh , AX = FFFCh , BX = 0003h.
DX:AX = -4

Dec quo	Dec rem.	AX	DX
-1	-1	~~0007~~ FFFF	FFFF

3) -4 | -1
-3
-1

d) DIVIDE OVERFLOW

4. a) AX = 0000H , BL = 03H

Dec quo	Dec rem	AL	AH	AX
4	1	04	01	0104

AX = FFFBH , BL = FEH

= -65531 = -254

Dec quo	Dec rem	AL	AH	AX
257	-253			

AX = 00FE , BL = 10

AL	AH
15 → 0F	14 → 0E

5. a) 0000

b) FFFF

c) 0000

6. a) FFF0

b) 005F

c) FF80 .

7. a)
```
MOV AX, 5
IMUL AX
MOV A
SUB AX, 7
MOV A, AX
```

b)
```
MOV AX, A
SUB AX, B
ADD B, 10
IMUL B
MOV B, AX
```

c)
```
MOV AX, 9
IMUL A
MOV BX, 6
SUB BX, AX
MOV A, BX
```

d)
```
MOV AX, A
IMUL AX
MOV A, AX
MOV AX, B
IMUL AX
ADD
MOV BX, AX
ADD BX, A
MOV AX, C
IMUL AX
CMP BX, AX
JE SET_CF
CLC
JMP END_
SET_CF: STC
END_
```

1.(c) AX = 0005h , CX = ffffh.

= -1.

Dec product	DX	AX
-5	ffff	fffB

2. b) AL = ABh BL = 10h

= 171 = 16

= 10101011

01010100

+ 1

-85 = 01010101

-85 x 16 = ffff FAB0

d) AL = 02h , B1 = FBh.

= 2 = 251

= 11111011

00000100

+ 1

-5 = 00000101

prod = -10

FF : F6

3-c) DX = ffffh , AX = fffch

DX : AX = ffff &fffc

= -4.

BX = 0003h

= 3.

3) -4 (-1

-3

-1

DX	AX
ffff	ffff

4. b) AX = fffB

&

1111 1111 1111 1011

0000 0000 0000 0100

+ 1

-5 = 0000 0000 0000 0101

BL = FEh

= 1111 1110

0000 0001

+ 1

-2 = 0000 0010

-2) -5 (2

-4

-1

AH	AL
-1	2
FF	02

Lecture

Compare string

CMPSB

CMPSW.

DS:SI

ES:DI

Same হলে 0

SI SI
↓ ↓
STR1 DB 'ACD'

STR2 DB 'ABC'
↑↺ DI
DI

CLD ;left to right

CMPSB

A - A = 0 , ZF = 1.

CLD;

CMPSB

C - B , ZF = 0.

- 43h - 42h = 1

REPZ
condition REPE

Compare string → বড় string এর substring কিনা বের করা.

SI SS
↓ ↓
SUB1 DB 'ABC'

SUB2 DB 'CAB'

MAINST DB 'ABABCA'.
↑↺ DI
DI
SI
↓
ABC

ABABCA
↑
DI

SI
↓
ABC

ABABCA
↑
DI

DI SI SS
ABC

ABABCA
↑↺ ↑↺
DI DI DI

STOP = MAINST + length_MAINST - length_SUB1
(offset address)
= MAINST + 6 - 3

= MAINST + 3.

ABABCA
M M+1 M+2 M+3 M+4 M+5

Hardware / Architecture related

Chapter-11

SI DI

ARR DW 10, 20, 40, 50, 60, ?

STD
LEA SI, ARR+8h
LEA DI, ARR+Ah.

If DF=0, SI, DI increments
DF=1, SI, DI decrements.

MOVSB → Source → DS:SI
Destination → ES:DI

STOSB → Source → AL/AX
Destination → ES:DI

LODSB → Source → DS:SI
Destination → AL/AX

SCASB → Source → AL/AX → AL - DI
Destination → ES:DI

LODSB → loads byte in DS:SI

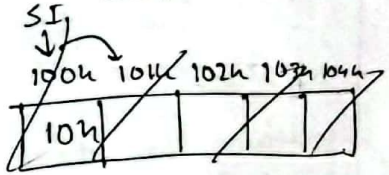
SCASB → scans byte in ES:DI

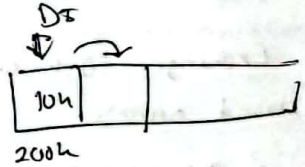
CMPSB → Source DS:SI — Destination ES:DI

Exercise

1. SI contains 100h
DI contains 200h
AX contains 4142h
DF = 0
CLD

100h → 10h
101h → 14h
200h → 20h
201h → 25h.





a) MOVSB → copy
SI → 101h → 15h
DI → 201h → 10h

b) MOV SW

c) STOSB
Source = AL
DI = 42h → 200h

d) STOSW
Source = AX
DI = 4142h

e) LODSB

Chapter-10

LINE DB 5,4, 3DUP (2, 3 DUP(0), 1)

LINE DB 5,4, 2,0,0,0,1, 2,0,0,0,1, 2,0,0,0,1

location of array elements

Position	Location
1	A
2	A = 1XS
3	A = 2XS
.	.
.	.
N	A = (N-1) XS

Ex10.1 Exchange 10th and 25th elements in a word array.

A of W + 9X2 = W + 18

A of W + 24X2 = W + 48

MOV AX, W+18

XCHG W+48, AX

MOV W+18, AX.

Addressing

1) register mode → operand register

2) immediate mode → " constant

3) direct mode → " variable.

MOV AX, 0 ← source. immediate mode

↳ dest. register mode

MOV ALPHA, AX

↳ source register mode

↳ dest. direct mode.

Register Indirect Mode → acts as pointer to the mem loc.

BX, SI, DI → operand's segment number in DS

BP → " " " in SS

SI contains 0100h → 1234h

MOV AX, [SI] → 1234h

MOV AX, SI → 0100h. [register]

10.2 Ⓐ

	Source offset	result
a)	1000h	1BACh
b)	2000h	20FEh
c)	illegal	must be BX, SI, DI
d)	illegal	mem-mem add
e)	3000h	031Eh

Based & Indexed Mode

Offset address is obtained by adding a number called displacement

MOV AX, W[BX] → register
↳ displacement

MOV AX, [W+BX]

MOV AX, [BX+W]

MOV AX, W + [BX]

MOV AX, [BX] + W

MOV AX, 2[SI]

MOV AX, [SI+2]

BX or BP → Based

SI or DI → Indexed

10.6 Ⓐ

	Source offset	result
i.	[ALPHA+2]	0456h
ii.	BX+2 2+2 = 4	2BACh
iii.	ALPHA+4	~~0ABCDh~~ 0789h
iv.	-2[SI] [SI-2] = 4-2 = 2	1084h
v.	[ALPHA+4]	0789h
vi.	illegal	
vii.	illegal	

PTR operator

MOV AH,1 ✓

MOV AX,1 ✓

MOV [BX],1 ✗

MOV BYTE PTR [BX],1

MOV WORD PTR [BX],1

Using PTR to override a Type

type PTR address_expression

LABEL Pseudo-op.

10.9

a) illegal

b) 7

c) 0BC9Ah

d) 5678h

e) 9Ah

f) 0BC9Ah

Segment override.

segment_register : [pointer_register]

MOV AX, ES:[SI]

MOV AX, DS:[DI].

Accessing stack.

BP → SS.

MOV BP, SP → BP points to stack top

MOV AX, [BP]

MOV BX, [BP+2]

MOV CX, [BP+4]

Two dimensional Array

memory → one dimensional

so, two dim. array elements must be stored sequentially.

i. Row major order — row1 — row2 — row3

ii. Col major order. — col1 — col2 — col3

Locating an element in an array

i. Row major order — M×N array

M = row
N = col

S = 1 → byte
S = 2 → word.

(1) Row 1 → A

Row 2 → A + N×S

Row 3 → A + 2×N×S

Row 4 → A + 3×N×S

⋮

Row i → A + (i-1)×N×S

10.12

(2)

i. Row major →

$A[i,j] \rightarrow A + ((i-1) \times N + (j-1)) \times S$

2. Col major →

$A[i,j] \rightarrow A + ((i-1) + (j-1) \times M) \times S$

1. S = 2 → word

Row i begins, $A[i,j] = A[i,1]$

$A + (i-1) \times N \times 2$

2. col j begins, $A[1,j]$

$A + (j-1) \times 2$.

3. N columns, 2×N bytes between elements.

CamScanner

Based Indexed Addressing Mode

variable [base_reg] [index_reg]

[base_reg + index_reg + variable + constant]

variable [base_reg + index_reg + constant]

constant [base_reg + index_reg + variable].

MOV AX, W[BX][SI]

MOV AX, [W+BX+SI] → MOV AX [W+6]

MOV AX, W[BX+SI]

10.13

5 X 7 → word

M X N.

Row major order.

1) clear row 3.

A + (3-1) X 7 X 2

→ A + 28.

MOV BX, 28.

MOV CX, 7

MOV A[BX][SI], 0

then SI + 2

2) clear col 4

A + (4-1) X 2

= A + 6. → col 4, Row 1.

= A + 6 + 2X7 → col 4, Row 2

= A + 20

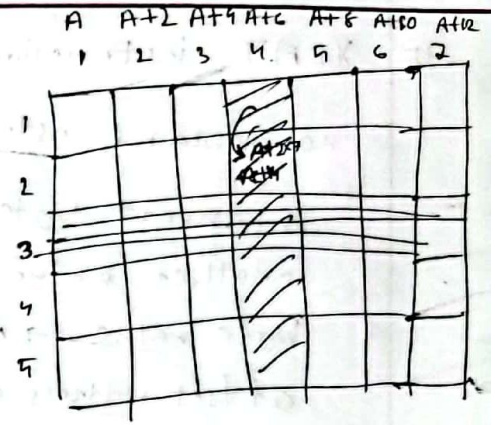
= A + 20 + 2X7 → col 4, Row 3

= A + 34

= A + 34 + 2X7 → col 4, Row 4

= A + 48 + 2X7 → col 4, Row 4

= A + 62



XLAT instructions

no operand instruction.

convert byte value into another value.

byte value → AL

offset address of conversion table → BX

Convert 0Ch to 'C'

```
MOV AL, 0Ch
LEA BX, TABLE
XLAT
```

↓

TABLE + Ch

TABLE + 12.

043h = C.

Exercise

1. a) legal ; DI = 1500h

b) legal ; DI = 0200h

c) legal ; AX = 0500h + 0150h

= 0650h

d) legal; BX = 1000h - 0200h

~~= -1000h~~

= 0E00h

e) legal; BX = BX + BETA + BX

= 1000h + 1000h

= 2000h

f) illegal; mem-mem

g) illegal

h) illegal

i) AX = [BX + DI + BETA]

= [1000 + 2000 + 1000]

= [4000h]

= 0300h

2. a) legal , AH = 01h

b) AX = 0504h.

c) AX = 'BA'

d) illegal

e) legal; AH = 'A'

3. a)
```
MOV BP, SP
MOV [BP], 0
MOV [BP+2]; 0.
```

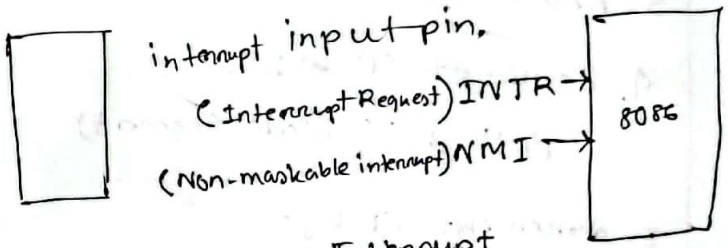
b)

Lecture

8086 Pin Diagram

Hall — Chapter 8

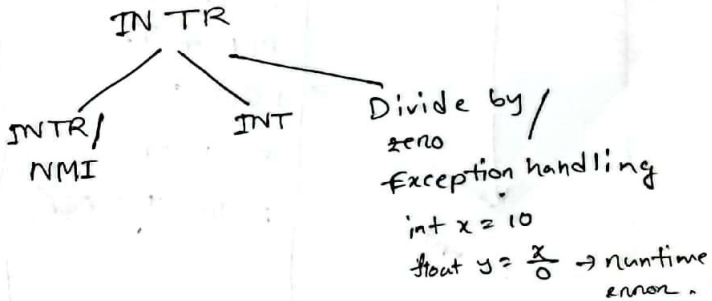
8086 Interrupts & Interrupt Applications



IRET → Interrupt Return

INTR/NMI ... External interrupt → Hardware interrupt

INT ... program ... interrupt ... → software interrupt



1. decrement SP by 2, PUSH → flag register

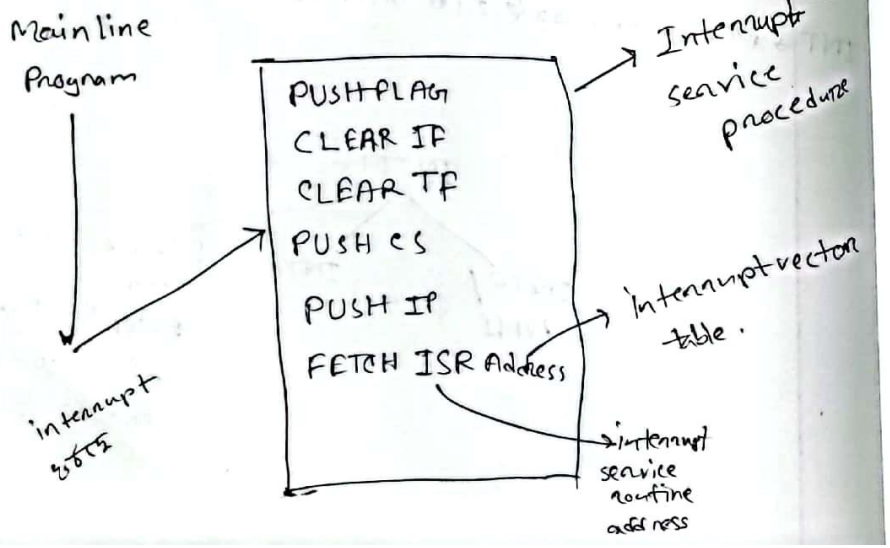
2. disable INTR input, IF clear
   করে দিবে, IF = 0 / Clear.

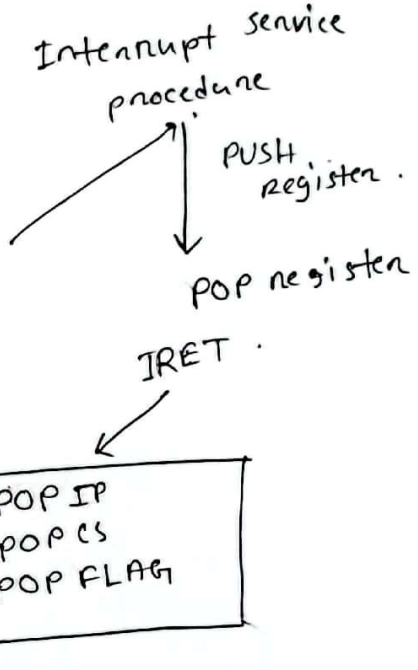
3. Reset → TF (Trap flag)

4. decrement SP by 2.
   PUSH → CS (code segment)

5. decrement SP by 2.
   PUSH → IP

6. Jump → Start of Procedure





interrupt কয় প্রকার?