27-02-23

Week-1/3

Microprocessors & Assembly Language

Brain of PC

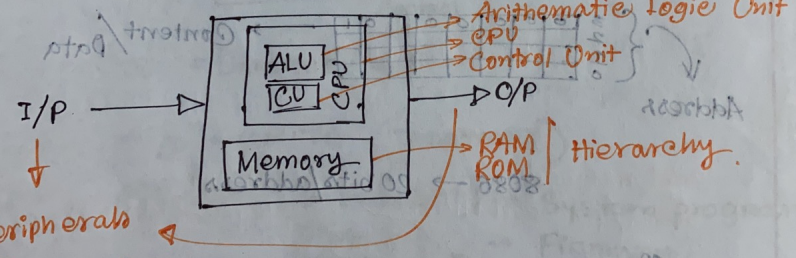
8086 ME

Mid Level Language

Why?

(Human) High Level → Compiler → Low level (Machine)

Von Neumann Architecture Simplified



CPU → Microprocessors

Mother board....

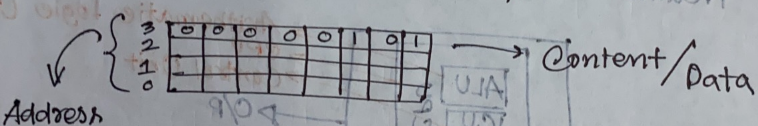
RAM → READ + WRITE → Harddics/Drive → RAM

ROM → Only READ. (Non Volatile) Can't update.

Harddics/Drive → Run korte chayle RAM এ Nite hobe

RAM-e data thake kivabe?

8bits = 1 byte.



8086 → 20 bits/address

2^{20} byte = 1 MB

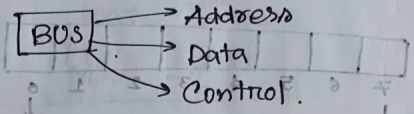
2^{30} byte = 1 GB

Address-এ Bit সংখ্যা

2^{32}

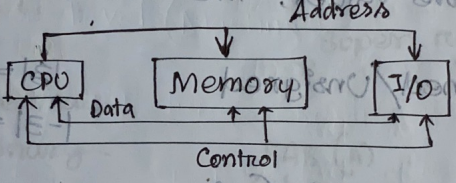
64 bit pe***

Word size depends on byte size.



Control Signal : READ & WRITE

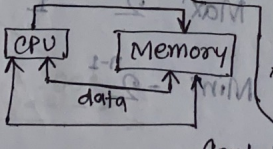
egular kaz



CPU → Unidirectional

RAM - READ & WRITE. (Temporary)

ROM - READ ONLY (Permanent) → System program.



Content of Memory Location → Firmware

READ korar jonno.

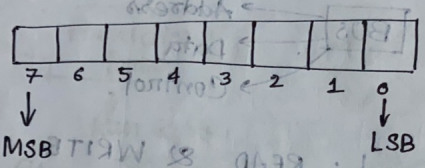
Address of Memory toe place kore Address bus -La.

Control → READ Operation.

01-03-23

Week-1/5

8 bit = 1 byte



For -ve values, the MSB has 1

Signed / Unsigned

$|3| = 3$

$|-3| = 3$ } Magnitude.

1s compliment (Flip)

2s compliment (+1)

Unsigned Representation

$2^n - 1$

Signed Representation

Max: $2^{n-1} - 1$

Min: -2^{n-1}

All Chapter-2 things

Unsigned	Signed
Max $2^n - 1$	Max $2^{n-1} - 1$
Min 0	Min -2^{n-1}

1000

Signed

Keno Hexa use kori?

= Binary. er shathe super related.

1010d → decimal

1010b → binary

1010h → hexa

FA8 (h)

1111 1010 1000 (b)

FCB9

MSB > 8

hole

negative

F9

1111 1001

Signed bit

1111 1001

00000111 → 7 (-7)

7F
FF
} hexa

1000 0000
8 0
} hexa

Code in Assembly Languge

Hello World
Substruction
Addition

Chap-4

- model small
- Stack 100h

```
A  DW  2
B  DW  3
C  DW  ?
```

- Code

```
MAIN PROC
 MOV  AX, @Data
 MOV  DS, A
 MOV  AX, A
 ADD  AX, B
 MOV  C, AX
 MOV  AX, 4C00H
 INT  21H
MAIN END
 END MAIN
```

Temp. Variable

→ Stack Segment
→ Data → Variable use kori
→ Code → Instruction thake

Initialization

Opcode (ki operation)

Opcode / Operand.

04-03-23
Lab - 1
Week - 2

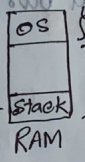
.asm extension. ta File save korte hobe.

.model small → use kora hoi.

Load korte → RAM

Software
System
Application

.stack 100h → 256 B



Paging
Segmentation

Load Effective Addr.
LEA → Pointer jonno

HW
1. How to show 5+5=10
2. User input A, B.
3. Substruction e. Sum e. Move hole

Stack Segment [Local Variable]
256 B.

Memory to Memory (Illegal)
MOV A, B
MOV C, A

.data → 16 bit

```
A dw 2
B dw 3
C dw ?
```

dx
cx

.code

```
main proc
MOV AX, a Data
MOV DS, AX
```

Accumulator
Register
Temp. Value

Base Address dhore Rakha

.cde

```
main proc
MOV AX, a Data
MOV DS, AX
MOV AX, A
ADD AX, B
```

LR: Code Output Discussion

a keno direct a Data → DS newa jay na?

```
MOV C, AX
MOV DX, C
```

```
MOV AH, 2   → Output
INT 21H     → Interupt.
```

06-03-23

Week-2/3

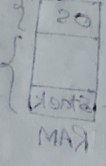
Microprocessors

Evaluation of Microprocessor: (Intel)

4 bit - microprocessors.

Address bus/Data bus - ekta upor depend kore

MP - ekta bit.



INTEL 4004:

Clock speed 740 KHz.

4 bit MP

INTEL 4040:

Dif. betn:

8085 vs. 8086

8086 vs. 8088

8 bit - microprocessor:

INTEL 8008

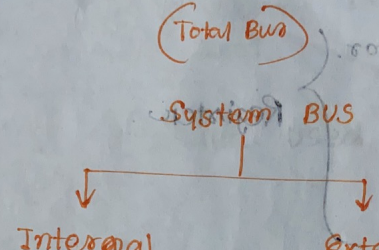
INTEL 8080

INTEL 8085

16 bit - MP:

INTEL 8086.

Internal Architecture of INTEL 8086

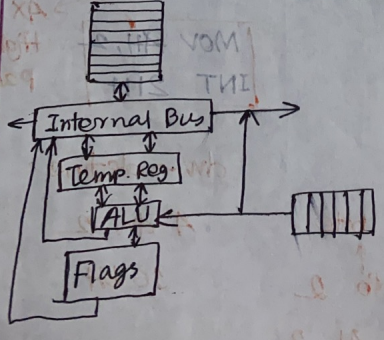


6 byte extra instruction is stored in "Instruction Queue".

↓

Compared to standby

Bus Interface Unit



Week-2/4

Types of Registers of INTEL 8086.

- **Data Registers:**

AX - Accumulator.
BX - Base
CX - Count
DX - Data
} Register.

16 bit registers

15 ——— 7 ——— 0
Higher Lower

AX - Data movement

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

ADD A,B
Memory to Memory add kora jay na.

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

AX er Higher part

dw → define
A dw 2
16 bit

bite.

Memory বাঁচাতে পারি

```
A db 2
MOV AL 2
```

BX — Address Register :: Address store korar jonno.

CX — Loop counter, Counter variable store kori. Left/Right shift.

DX - data register. used in MULTIPLICATION/ Division.

- Segment Register:
  - Code Segment:
    - ↓
    - অংশ/ভাগ

- model small
- stack 100h (256 B)

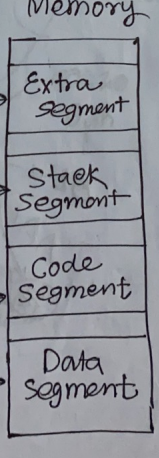
Temp/Variable/value kichu store korar jonno shurute jayga allocate kora hoi.

stack declare na korle 1KB jayga nijey nibe.

প্রত্যেক Segment-এর base address store kore.

Four Segment in BIU Register

ES	7000 H
CS	5000 H
SS	3000 H
DS	2000 H

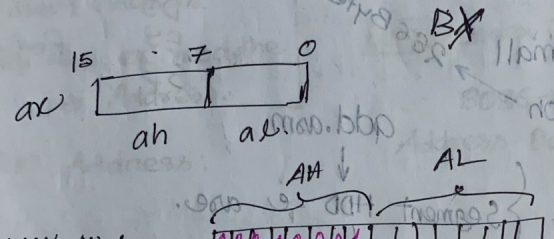


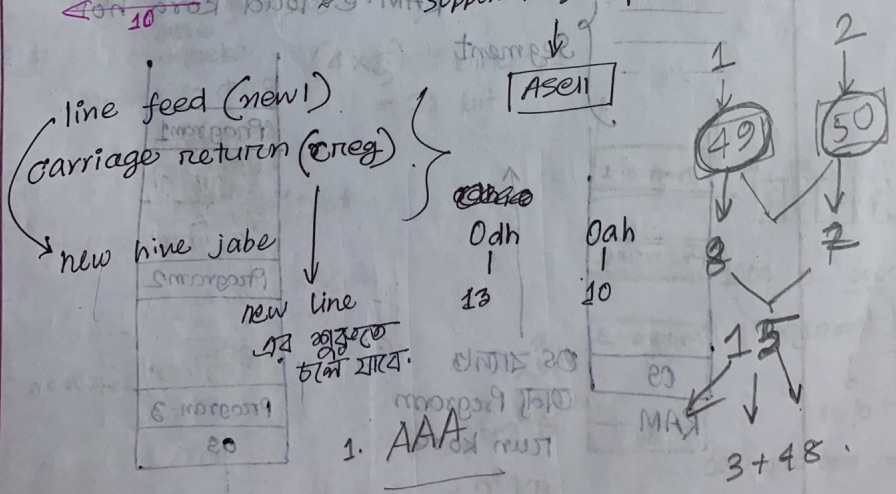
Address BUS 20 bit.

11-03-23
Lab-2
Week-3

AAA – Adjust Assii after Addition

LEA – Load Effective Address.





14-3-23

Week-3/4

AX
BX
CX
DX
} 16bit registers.

CS
DS
ES
SS
} Segment Register.

- model small
- stack 100h → 256 Byte
- data
- code

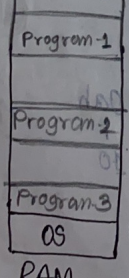
add.asm

Segment HDD-e ase.

Run korle

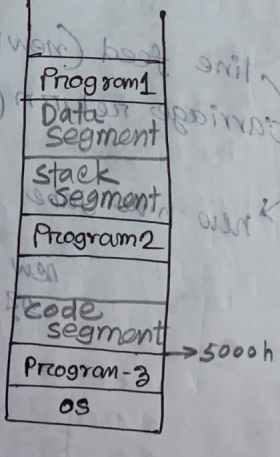
RAM-e load kora hoi.

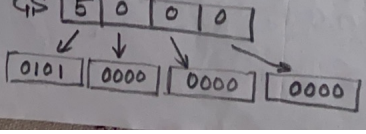
Segment



RAM

OS বাদেও অন্য Program run kore





Pointers & Index Registers:

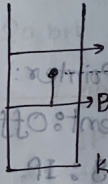
Segment Address:
Seg Reg. er moddhe jei Address.

IP = Index Pointer

8086 →

Address Bus 20 bit (er Address gen. korte pore)

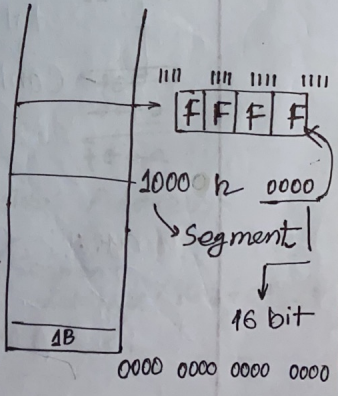
Offset Address:



5bit hex. (Physical Address)

(4×5) binary = 20 bit.

Address আছে তাই Offset.



$2^6 \times 2^{10} B$

$= 2^6$ KB

location in → = 64 KB. segment range is Offset Address.

Segment Address : 16-bits.

↳ CS, DS, SS, ES

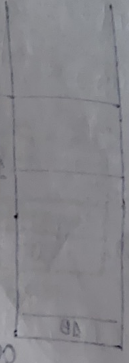
Offset Address : 16-bits

↳ IP, BP, SI, DI, SP.

Instruction Pointer:

offset of next instruction in code segment.

Stack Pointer:



Logical Pointer:

Segment : Offset.

CS : IP

↙ Base Add.

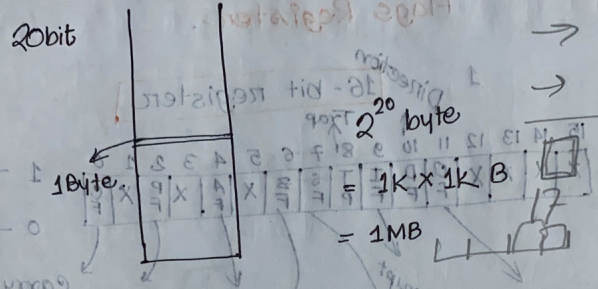
↓ Offset Add.

Stack Segment { SS : SP → stack Pointer
SS : BP → Base Pointer

DS : SI } Source Index.
ES : DI

↓ Destination

BP er niye jekono Segment er data access korte pari.



Q: Segment + Offset. = 16 bit + 16 bit theke kivabe 20 bit 8086 pabo?

CS = 1234H.
IP = 0128H.

5bit × 4

= 20 bit

or physical Add.

(20+16) bit

= 20 bit.

1100
1000

Discard na korle -

11000!

~~1000~~
~~0010~~
~~1000~~

Physical Address

= (Segment Address X10) + Offset

CS 3000
0000 Offset.
30000 - 20 bit.

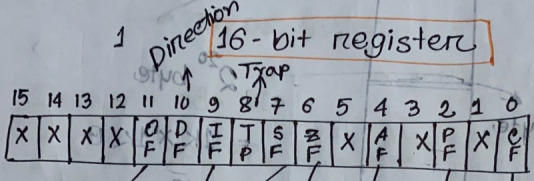
Changing.

Flaux Address.

15-3-23

Week - 3/5

Flags Register.



Flags

- Status (6) → Result er vitti te status determine kore
- Control (3)

DF ✓
TF ✓
IF ✓

চুকবে যাচ্ছে

C P A Z S O
0 2 4 6 7 11

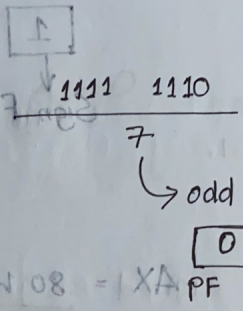
Carry:

baki - Dont Care

AX = FFFF h

BX = FFFF h

ADD AX BX → 1FFFE h



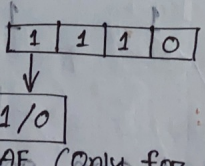
Parity:

Lower 8 bit এ Even amount of 1. → 1

Axilary Carry: (ONLY FOR BCD operation)

Example:

Same as carry but considered for lower nibble.



Zero: Result 0000 হলে, 1

না হলে, 0.

Sign: MSB ex value.

[1] → Sign Flag

[0]

AX = 80 h = 1000 0000

BX = 80 h = 1000 0000.

1 0000 0000

↳ Expected different : 0

Sign	zero	Parity	carry	
0 ✓	1 ✗	1 ✓	1 ✓	OK

Overflow:

① Expected sign same - no overflow.

② " " different - overflow

0~7 (+)ve

8~F (-)ve

Substruction এ Overflow Impossible.
/different sign.

$$\begin{array}{r} -7 \\ +8 \\ \hline 1 \end{array} \qquad \begin{array}{r} +7 \\ -8 \\ \hline -1 \end{array}$$

OV=0.

Chapter-1 }
Chapter-2 } Slide.
Chapter-3 }
→ Chapter 4.

Chapter-5

+7
+8

18-03-23
Lab-3
Week-4

AAA - Adjustment After Addition.

MUL, DIV } Unsigned Multiplication & division.

Signed er jonno: Chapter-9 cover hoye jabe.

IMUL
IDIV.

cz signed number has different representat

kivabe use korbo:

Opcode Multiplier

IMUL Source
MUL Source
DIV Source
IDIV Source

MUL BL দিয়ে সুরু করবো
↓
Multiplier

Multiplicand

ba variabe

MOV AL, 2
MOV BL, 3
MUL BL

MULTIPLICATION :

AL → Multiplicand.

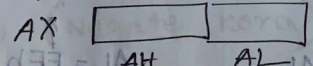
After Multiplication:

AL X BL

= AL → Product (X)

AAM
→ Adjust After Multiplication

MOV AX, 20h
MOV BX, 4h
DIV BL



AL = 12

(8X8) bit = 16 bit

Signed

9h

4

Remaind

Divisor

22-03-23

Week-04/05

Chapter-5

SUB AX, BX

AX = 8000h
BX = 0001h

```
 8000
-0001
------
 7FFF
```

Carry Flag = 0
PF = 1
SF = 0
ZF = 0
OF = 1

i = 5
++ } Ineremement.
--

EMU:

INC AX
DEC

INC AL

AL = FFh

-ve

AL = 100h

Carry Flag INC-er
যায়না Affected hobe না.
Also, DEC.

PF = 1
SF = 0
ZF = 0
OF = 1 0

```
  0100
+ 0001
------
  0101
```

carry আসে ও বের হয়ে যায় OF = 0
borrow নিয়ে আবার বের হয়ে যায় OF = 0 } For MSB.

দুইটা (+)ve
sign (-)ve

দুইটা (-)ve
sign (+)ve

Overflow = 1.

MOV AX, 5h: MOV Operation - এ Flag unaffecte

Example 5.5: Flag unchanged.

NEG (Negate kora)

AX = 5h 0005h SF = 1

NEG AX ..... 0101 binary ZF = 0

..... 1011 PF = 0

0005 h FFFBh OF = 0

NEG kore always CF = 1

Zero thakle NEG kore CF = 0

Trap Flag:

TF = 1 থাকলে Single Step Run korte থাকে.

- Interpreter line by line Before & after run of programe.

Interrupt Flag:

IF = 0 (Interrupt ignore)

IF = 1 (Enable, kora)

↓

SW/HW. Interrupt

Direction Flag:

String processing higher → lower. DF = 1

lower → higher DF = 0

29-4-23

Lab-4

Week-5

WRITE PROGRAM:

Enter two number: 5 6

Sum = 11

Enter two number: 5 3

Sub = 2

Enter two number: 5 2

Product = 10

Enter two number: 8 3

Quotient = 2

Remainder = 2

Discussion: Flag er value. [Single Step].

4 - 4004, 40

8 - 8008, 8080

16 - 8086, 8088

2-4-2023

Week-5/4

Chapter-4

- model small
- stack 100h → Pseudo → Model-er size define korte, 256 byte jayga allocate kori.
- data → Directive

msg db "Hello $"
a db ?

→ capital/small same na

- code → Pseudo-op

main proc → Directives

MOV AX, @data
MOV DS, AX → Instruction

.....

main endp

end main

Statement-er format:

Name - operation - operand/s - comments

[name operation operand/s comments]

L1: MOV AS, AX ;.....

CT

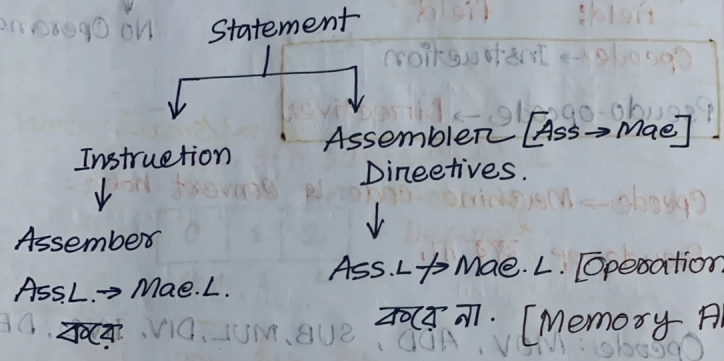
Chap-5
Flags

Chap-3
- Gi Register
- S. "
- Physical Address calc.
- Konta koto bit.
- 8086-8088
- 16→20 bit addres
- Logical Addres

Chap-1
- Figure
- EU - IU
- Bus Architech

Statement:

Code এর প্রত্যেকটা Line.



L1: — Name

msg — Name — db "Hello $"

main — Name — proc — Operation

Convension: Character [1-31]

digit/letter/specific special Char

শুরুতে থাকতে পারবে না

? : @ $ _ %

Majh khane thakte parbe

শুরুতে OK na

A a → Same jinish

Hexa Num: OFFh

Name with no digit as initial.

MOV | DS, AX

operation field: | operand(s) field

nop → No Operand

Opcode → Instruction

Pseudo-opcode → Directives

Opcode → Machine code ↳ Convert hobe

Pseudo-op হবে না।

Opcode: MOV, ADD, SUB, MUL, DIV, INC, DEC, NEG

INC AX → 1 bit

MOV AX, DX → 2 bit

Comments: → ;

To increase readability.

db – 1 byte

dw – 2 byte

dq – 8 byte

dd – [Double define] 4 word → 4 X 16 bit quardword

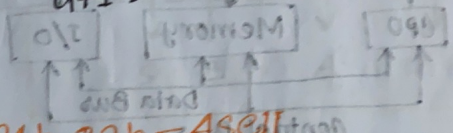
dt – 10 byte 4 word 2 X 16 bit

Array:

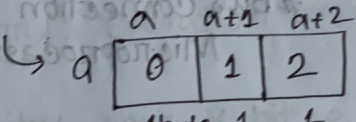
int a[5]

a → base address

a+1 →

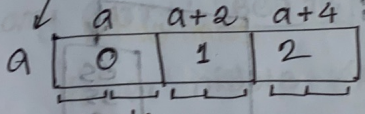


a bd 30h, 31h, 32h – Ascii



– Decimal.

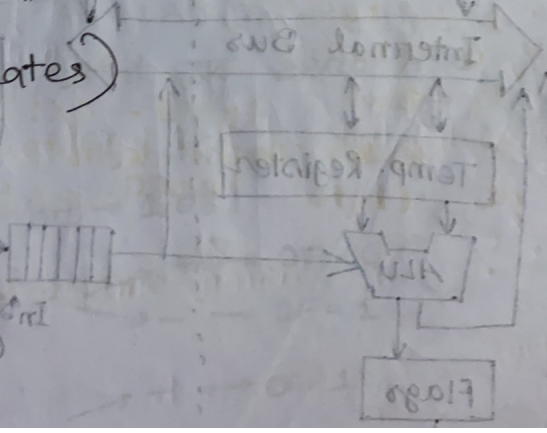
a bw 30h, 31h, 32h.



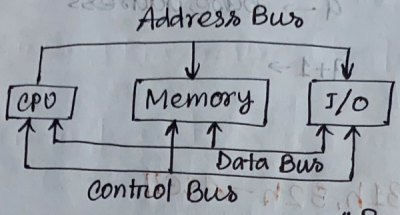
#define N 100

EQU (Equates)

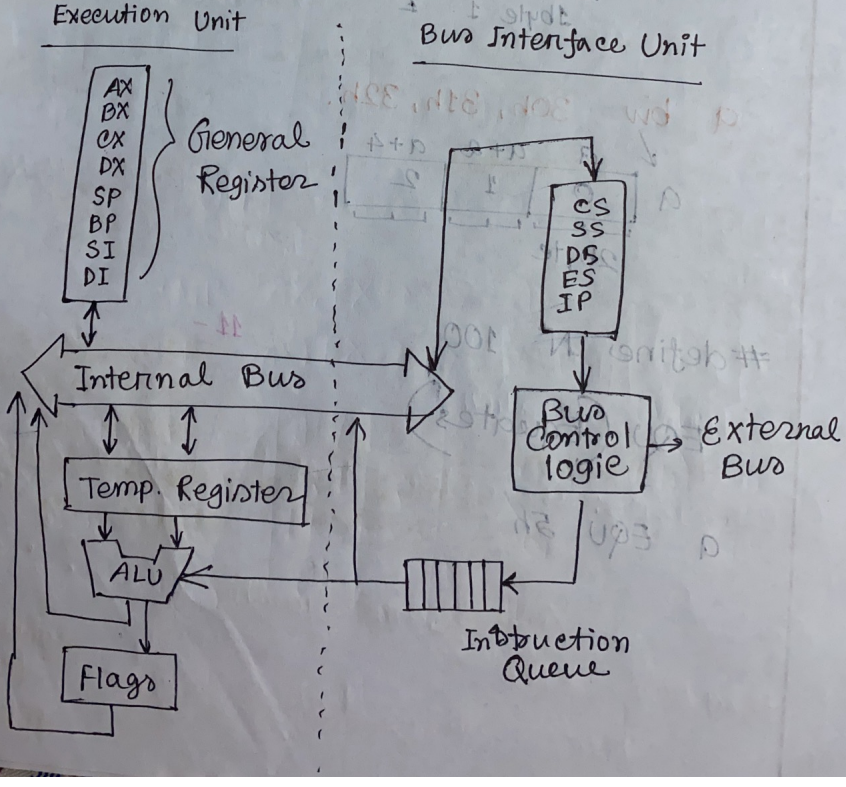
a EQU 5h



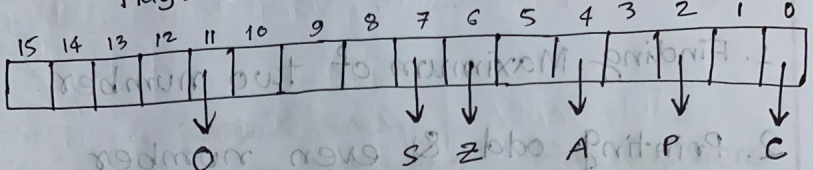
Extra:



Bus Connection of a Microprocessor.



Flags 16 bit Register



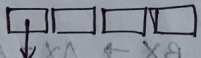
Carry: Operation er carry.

Parity: Last 8 bit e

Amount of 1 even hole → 1.

Auxilary Carry: BCD te

lower nibble - 4 bit



A: 1/0

Zero: 8 bit / 16 bit ~~अ~~ zero hole Yes → 1
No → 0.

Sign: Sign bit → 1/0

Overflow: + + → + → OF = 0
→ − → OF = 1

− − → + → OF = 1
→ − → OF = 0

6-5-23
Lab-5
Week-5/1

LR:

1. Finding Maximum of two number.
2. Printing odd & even number.

Chap-6.

JMP

Conditional Unconditional.

JUMP when zero

CMP AX, BX → AX = AX
SUB AX, BX → AX = AX - BX
} But AX এর value compare hobe

JUMP IF Greater JG

JUMP IF LESS JL

CMP 6h 5h

JG বড়

JL বড়

20-5-25

Lab-6

Week-7/1

CMP
SUP

JZ
JNZ } Jump if zero or Not zero. Flag Register FR=1.

BL K 16 bit la extend kora.

22-05-23

Week-7/3

Different Opcode: ADD, SUB, INC, DEC, NEG

ADD AX, BX → AX = AX + BX → Addition

SUB AX, BX → AX = AX - BX → Subtraction

INC AX → AX++ → Increement

DEC AX → AX-- → Decrement

NEG AX → 2's complement.

AL = 3
BL = -5

-2 → AL (in 2's complement form)

General Segment Memory Location

SOP \ POP	GR	SR	ML	Constant
GR	✓	✓	✓	X
SR	✓	X	✓	X
ML	✓	✓	X	X
Constant	✓	X	✓	X

Destination can't be constant

ek segment theke arek segment register nite parbo na.

MOV DS,SS X

Address fixed *

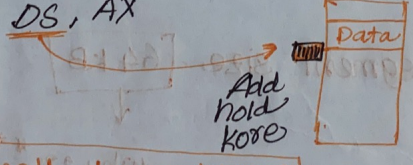
MOV A,B X

MOV/ XCHG AX,BX

N.B. Look over charts from Textbook Chapter-4

MOV AX,@data → constant

MOV DS, AX

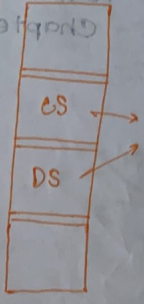


• model small keno likhi ?

- small *
- medium
- compact
- large
- huge

When where to use?

	Model	Description
code < 1 data < 1	Small	Code & data 64 KB - cross na korle (In one segment)
code > 1 data < 1	Medium	Code ~~যদি~~ segment er beshi Data in one segement
code < 1 data > 1	Compact	Code in one segment Data in more than one segment
code > 1 data > 1	Large	Code & Data more than one segment (No array larger than 64 KB)
code > 1 data > 1	Huge	Code & Data more than one segment (Array larger than 64 KB)



→ Case convession programe

Segment size - 64 KB

Why ??

INT 21H
- AH - 01 - Char input
- AH - 02 - Char output
- AH - 09 - String Output

ASCII

Assembly Run korar steps

Next: Chap-6.

24-May-23

Week-7/5

Chapter-6

JUMP

- Conditional
  - Signed
    - ADD AX, BX
    - JG/JNLE
    - Signed flag
    - Zero flag
    - Overflow flag
    - JG → ZF = 0, JNLE → SF = OF
    - JGE/JNL
    - JL/JNGE → SF ≠ OF
    - JLE/JNG → ZF = 0
  - Unsigned
    - Carry Flag
    - JA/JNBE → JUMP ABOVE
    - JUMP if NOT BELOW OR EQUAL
    - JAE/JNB → CF = 0
    - JB/JNAE
    - JBE/JNA → ZF = 0/1
  - Signed flag
    - jz/jnz
    - jp/jpe
    - js/jns
    - jc/jnc
    - jo/jno
- Unconditional

jmp destination_label

CODE:

L1:

126B

JZ L1 | JNZ L2

JNZ L2

127 B

L2:

JMP L1

int a;

Switch (a) {

case 1 : P ——— P

case 0 . . . . . .

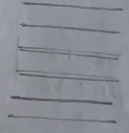
case -1 : . . . .

```
CMP AX, 0
JG  L1
JL  L2
JE  L3
```

LOOP:

Syntax : LOOP L1

L1:



LOOP L1

CX er value zero holeo program ekbar ghurbe.

0000

↓

FFFF

65535

Counter Register

CX → ৳ iteration value Nibo.

[Exam ৳ code-ex implementation আসতে পারে.]

29-05-23

Week-8/3

Chapter-7

Rotate Instruction, Logic, Shift

AND
OR
NOT
XOR

b AND 1 = b b OR 0 = b b XOR 0 = b

b AND 0 = 0 b OR 1 = 1 b XOR 1 = ~b

```
  10101101
AND 01111111  -> MASK
```

jeta clear korte chay oita zero kore baki shob 1 (one).

```
MASK <- 11111110  AND  10101101
```

jeta set korte chay oita one kore baki shob 0 (zero).

OR

Flip korar jonno XOR

Shob bit zero kore felar jonno:

```
AND AL, 0000 0000
SUB AL, AL
XOR AL, AL
```

Example:: Check

a = 97 → 01100001

A = 65 → 01000001

RESET → AND
SET → OR

b = 01100010 (D F)

B = 01000010 (2 0)

AL 11110000
BL 00001111

AND AL, BL

TEST AL, BL [AND er value ~~ekhn~~ kaz kore like CMP, SUB but AL er value unchanged & Flag er value change]

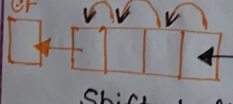
~~Shift~~

- ~~Left Shift → SH~~
- ~~Right Shift~~
- ~~Left Arithematic~~

Shift

- Left Shift → SHL
- Shift Arithematic Left → SAL
- Right Shift → SHR
- Shift Arithematic Right → SAR

Multiplication



Shift Left : MSB → CBit (carry)

LSB = 0.

ekta shift SHL AL, 1

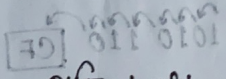
>1 shift SHL AL, CL

Shift Arithematic Left

0000 0101

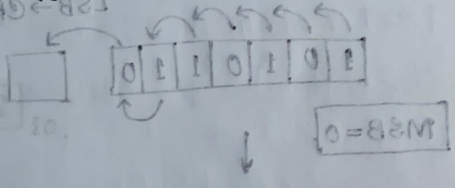
0000 1010 → 10

0001 0100 → 20



এটি Left shift 2x

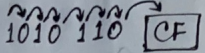
Multiply kore.



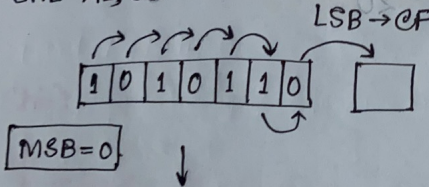
30-05-23

Week-8/4

Shift Right : SHR (Unsigned)



SHL AL, CL



3 times Right Shift.

01010 110

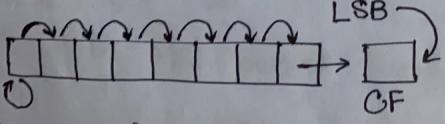
0000 1010

SHR AL, CL → 2+

For 2>

SHR AL, 1.

Shift Arithematic Right:: Division kori. (Signed)



MSB: MSB তেই থেকে যাবে।

[Divided by 2 in each shift]

$(15)_{10} = [00001111]_2$

2's complement,

11110001

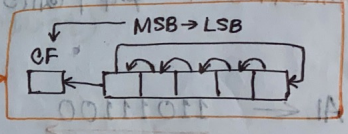
-8 11111000 → 1 (CF)

↓

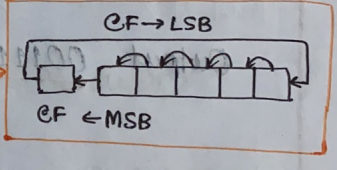
$00001000_2 = 8_{10}$.

Rotate Instruction:

- ROL : Rotate Left

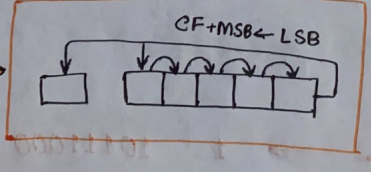


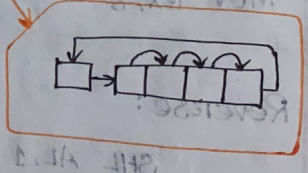
- ROR: Rotate Right
- RCL: Rotate Carry Left



- RCR: Rotate Carry Right

ROR→





ROR AL, CL

DL
CL

Example 7.14

CF = 1 CL = 3

RCR DH, CL

↓

8Ah

1

Last Rotation-e sign change hoye gele Overflow = 1, otherwise OF = 0.

Application: 7.4 ↑ আগের [Binary & Hexa input/output

AL ← 11011100

CX/CL ← 8

Output 00111011

Example ***

MOV CX, 8

Reverse:

SHL AL, 1 → 1 10111000

RCR BL, 1 → 0 10000000 } Repeat.

LOOP Reverse

MOV AL, BL

End Chap-7

Next Chap-8

31-05-23

Chapter-9

Multiplication & Division Instruction.

Unsigned & Signed Multiplication:

MUL BX (For 16 bit)

IMUL BX

AX ← Multiplicand

BX ← Multiplier

DX:AX → 32 bit (Product)

For 8 bit:

MUL BL
IMUL BL → AX → 16 bit (Product)

S Z AC P → Undefined for unsigned MUL

O C → Defined.

Unsigned:

Carry Flag: Upper Half zero hole CF=0.
OF=0

Signed Multiplication:

CF/OF = 0 hobe when MSB = 1 hole DX = 111.....

MSB = 0 DX = 000.....

Example:

AX ← 1

BX ← FFFFh

		Product	DX	AX
MUL BX	→	FFFF	0000h	FFFFh
IMUL BX	→	FFFF FFFF	FFFFh	FFFFh

CF/OF = 0

Example: 9.5

AL ← 80h (-128) 128

BL ← FF (-1) 255

MUL BL → 7F80

IMUL BL → 128 7F 80

10-6-23

Week-9/1

DIV BL AH AL

IDIV

String Reverse

Binary Conversion

8bit

BX ← 16 bits

BX clear করতে হবে (XOR)

AL 0000 0000

0000 1111

12-06-23

Week-9/3

DX:AX → DX AX

0000

Status Flag ↑ Undefined

Factorial

Division

CX ← 9

AX → Divisor of

Loop

DX:AX

Remainder Quotient

AH AL

Dividend = Divisor × Quotient + Remainder

3
2
1
0
-1
-2
-3

Dividend { DX ← 0000h
AX ← 0005h
BX ← 0002h

DIV BX

IDIV BX

Q	R	AX	DX
2	1	0002	0001

→ DIV

→ IDIV

DX:AX → 0000 : 0005h
DX AX

BX ← FFFE h (Unsigned 65534)
(Signed -2)

1110
0010 → 2

Q	R	AX	DX	
0	5	0000	0005	(DIV)
-2	1	FFFE	0001	(IDIV)

Divide Overflow:

AX ← 00FB h → 0000 0000 1111 1011

BL ← FFh → 1111 1111 255

Q	R	AL	AH	
0	251	00	FF	(DIV)
			0001	(IDIV)

Divide Overflow

$\frac{-5}{2}$

CBW - Automatically convert word to double wo
CBD

CT-3

DIV
IDIV

DX:AX
↙ Quotient
Remainder

word

9:12

MOV AX, -1250
CWD (extend sign to DX)
MOV BX, 7
IDIV BX

Byte

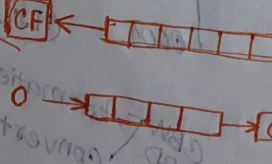
9.13

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

XD	XA	R	Q
	5A	12	0

MOV AX, 5
IMUL A
B
MOV 0A, AX
MOV AX, 12
IMUL B
SUB A, AX

SHL
SAL
SHR
SAR



19-06-23

Week-10/3

String Instructions

Direction flag (control flag)

DS: Segment Address

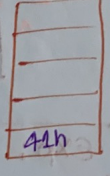
ES: Extra Segment Address.

Str1 DB 'ABCDE'

DS:SI → Source String

ES:DI → Destination String

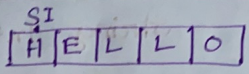
Offset Address	Content	Ascii Char
0200	41h	A
0201	42h	B
0202	43h	C
0203	44h	D
0204	45h	E



CLD → Clear Direction Flag

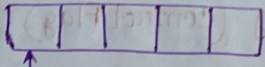
SLD → Set " "

Source String



Str 1

Destination String



Str 2

SI & DI depends on DF

Code:

•DATA

str1 DB 'HELLO'

str2 DB 5 DUP (?)

↳ Duplicate

↳ 5ta

exp: SP DW 212 DUP (0)

•CODE

main proc

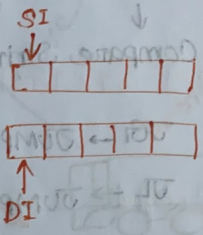
MOV AX, @data

MOV DS, AX

MOV ES, AX → Extra Segment এ কাজ

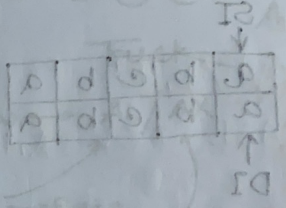
করানোর Segment Register এর Value

```
LEA SI, str1
LEA DI, str2
CLD
MOVSB
```

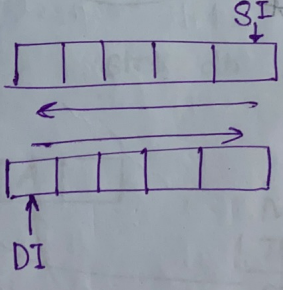



```
MOVSB
MOVSB
  .
  .
  .
MOVSB
```

```
MOV CX, 5
REP MOVSB
```




```
ADD SI, 4
STD
MOVSB
ADD DI, 2
```



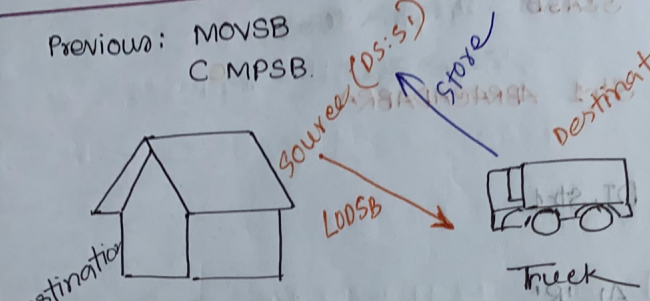

```
MOV CX, 5
L1:
  MOVSB
  ADD DI, 2
  LOOP L1
```

10th July

Week-10/3

Previous: MOVSB

CMPSB.



ES:DI ← Address → AL/AX

STOSB → Byte

STOSW → word

Function Design

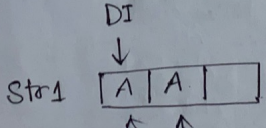
Read String

Out

SCANNext.

LEA DI, Str1

Str1 db 3 dup(?)



MOV AH, 1

INT 21H AL←A

STOSB

STOSB

L1:

STOSB

INT 21H

LOOP L1

MOV AX, @data

MOV ES, AX

LEA DI, Str1

CLD (left to Right)

Code from Book

Character input, BX ++, count o/p.

17th July

Week-11/3.

Chapter-10

Arrays & Addressing Mode.

Array

Declaration

```
A   db   10,11,12,13,14
         8  8bit

B   db   5 dup (0)  ----> (?) uninitialized
```

Index	Element	Symbolic Address
1	A[1]	A
2	A[2]	A + S×1
3	A[3]	A + S×2
4	A[4]	A + S×3
5	A[5]	A + S×4

S = 1 → Byte

= 2 → Word

A[N] → A+SX(N-1).

W[10] ⟷ W[25] word

W+18 W+48

XCNG W+18, W+48 → Not Possible.

MOV AX, W+18.
XCNG W+48, AX
MOV W+18, AX

DUP operator ব্যবহৃত Nested আকারে থাকে.

A db 5,4,3 dup (2,3 dup(0), 1).

A db 5,4,3 dup (2,0,0,1).

5,4,3,2,3,0,0,1.

Addressing Mode:

The way ~~of~~ in which the operands are specified.

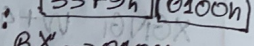
MOV AX, BX ⟶ Register Mode.

MOV AX, 3 ⟶ Immediate Mode (constant)

MOV AX, NUM ⟶ Direct Mode.

More four types of Addressing Mode.

4. Register Indirect Mode:



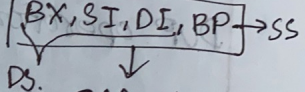
BX → 0100h

[BX] → 35F9h.

5. Based Mode.

6. Indexed

7. Based-Indexed



10.2

W DW 10, 11, 12, 13, 14

• Code

XOR AX, AX

LEA SI, W.

Loop chalaye content gulo newar try korbo.

25th July '23

Week-12-

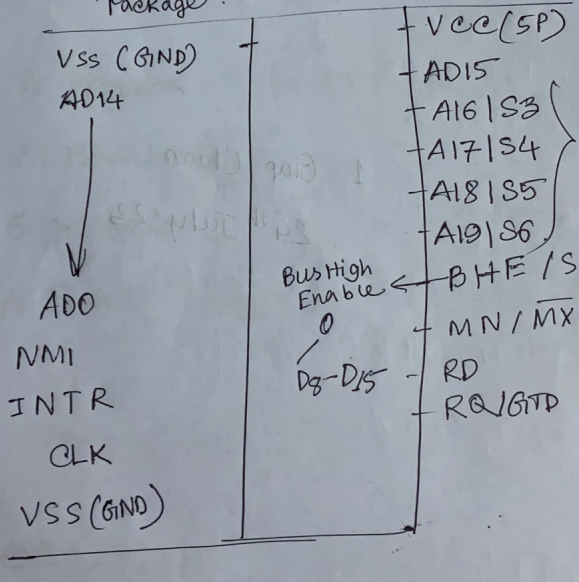
Pin Diagram of 8086.

Intel 8086. – 40 Pin

DIP

↓

Dual Inline Package.



প্রচুর Signal আদান প্রদান হয় → 2টা Ground.