2/8/17 Wednesday

MODULE I

1st generation (1971-1973)

- Low cost, speed, power
- pmos
- not compatible with high speed (TTL)

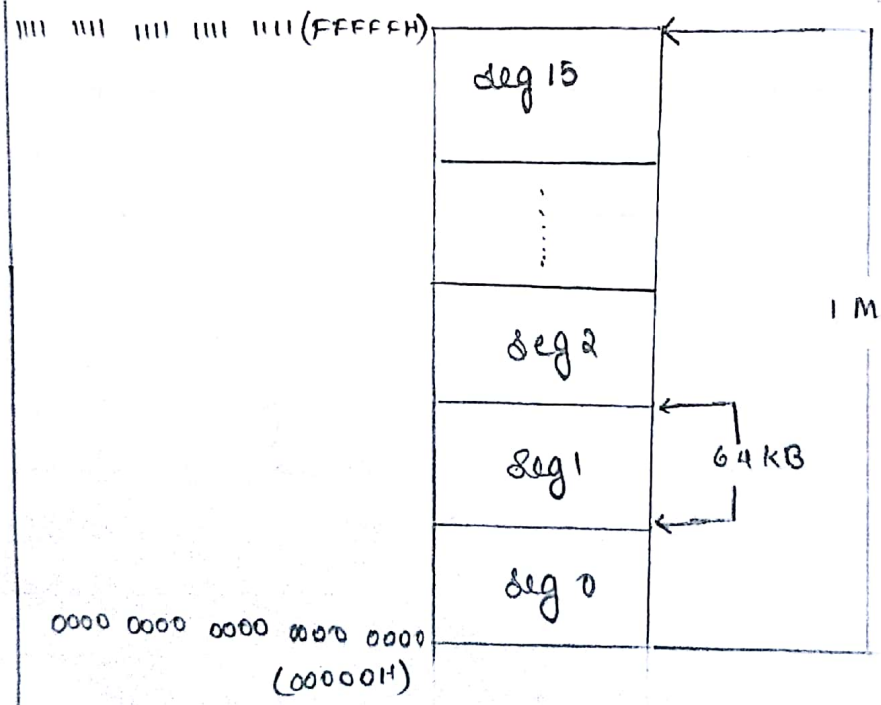
7/8/17 Monday

8086 Architecture

* 16 bit data bus, ALU, registers
* 20 bit address bus = 2^{20} (1 MB) memory
* 40 pins in 8086 chip

1) Bus Interface Unit (BIU) — Interface with peripherals
   - memory address calculation
   - pipelining

2) Execution Unit (EU)
   - Execute Instructions



$\#\text{segments} = \frac{1\,MB}{64\,kB} = \frac{2^{20}}{2^6 \times 2^{10}} = 2^4 = \underline{16}$

⇒ Data → segment address
↘ offset

⇒ Actual Physical Address (20 bit)
= segment address (16 bit) + offset (16 bit)

* 20 bit physical address ⇒ formation

1) shift segment address 4bit, left
2) Add offset

eg:- segment Add = 2001H

offset = A320H

2 0 0 1
0010 0000 0000 0001

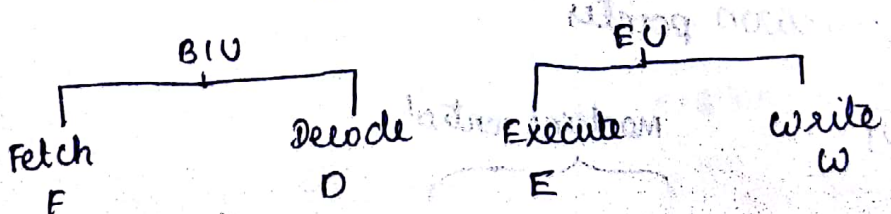
1 shift 00100 0000 0000 0010

2 shift
3 shift
4 shift 0010 0000 0000 0001 0000 + (offset)
0000 1010 0011 0010 0000
0010 1010 0011 0011 0000

20 bit Physical address : 2A330

Physical Address = (segment address * 10) + offset

* Parallel processing with pipeline implemented



BIU contains precoded queue for pipelining

8/8/17

8086 Registers

1> General data Registers (16) [EU]

	8	8	
AX	AH	AL	Accumulator
BX	BH	BL	
CX	CH	CL	Counter
DX	DH	DL	

2> Segment Registers (16) [BIU]

CS: Code Segment Reg
DS: Data "
SS: Stack "
ES: Extra "

3> Index & Pointer Registers (16)

EU
- SP: stack pointer
- BP: Base pointer
- SI: Source Index
- DI: Destination Index

BIU - IP : Instruction pointer

4> Flags (16) [EU]

Machine control

X	X	X	X	O	D	I	T	S	Z	X	Ac	X	P	X	C
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

O: overflow flag

D: Direction flag
- 0: Autoincrement
- 1: Autodecrement

1> status flag / Conditional code
2> Machine control

I: Interrupt

0: Ignore interrupt

1: Enable interrupt

T: Trap

S: Sign

0: +ve

1: -ve

Z: zero

1: result is zero

AC: Auxiliary Carry

P: Parity

C: Carry

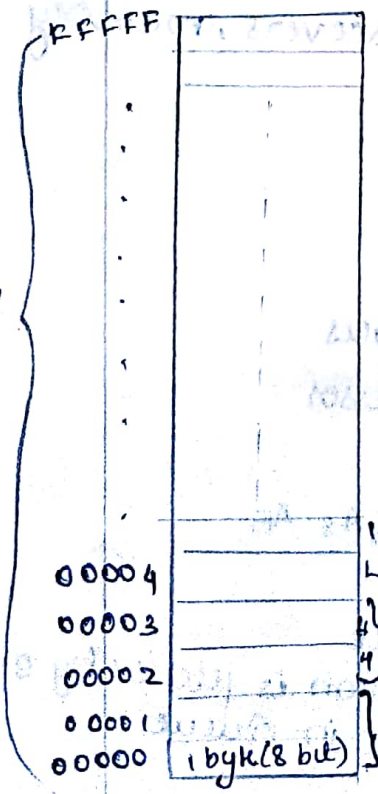
9/8/17 Wednesday

Physical Memory Organization

Data bus = $D_0 - D_{15}$

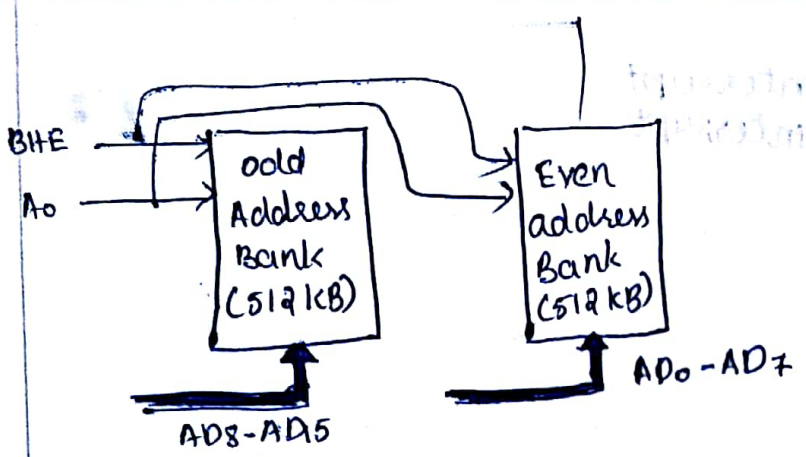
$D_0 - D_7$ Lower byte

$D_8 - D_{15}$ Higher byte



* (512 kB) Lower byte (0-7) in even add
= even address bank

* (512 kB) Higher byte (8-15) in odd add
= odd address bank



Modes of Operation

minimum mode	maximum mode
→ Only one processor (8086)	→ more than one processor
→ All signals (RD, WR, HALT - - -) generated by processor	→ 8086 generate only status signal S0, S1, S2. then bus controller generate all other signals
→ MN/$\overline{MX}$ = 1 MN = 1 MX = 0	→ MN/$\overline{MX}$ = 0 MN = 0 MX = 1
→ Latches, transcevers, memory	→ Latches, transcevers, memory

8088/8086

8086	8088
→ 20 bit add bus 1MB memory	→ same
→ 16 bit data bus	→ 8 bit databus
→ 16 bit processor	→ 8 bit processor
→ Even or odd address bank	→ No bank
→ AD0 - AD15	→ AD0 - AD17 & A8 - A15
→ $\overline{BHE}$ signal used	→ no
→ 6 byte Queue	→ 4 byte queue
→ BIU fetch next instruction when 2 byte is free in Queue	→ next instruction is fetched by BIU when 1 byte is free in Queue.
→ IO/$\overline{M}$	→ IO/$\overline{M}$

→ Maximum current: 360 mA

→ Max current: 340 mA

Pin diagram of Intel 8086

1) Common Pins
2) Maximum mode Pins } (24-31)
3) Minimum mode Pins }

* Pins $AD_0 - A_{15}$ are bi-directional
* $AD_0 - AD_7$: carries lower order byte of data
* $AD_8 - AD_{15}$: carries higher order byte of data

⇒ A_{19}/S_6 , A_{18}/S_5 , A_{17}/S_4 , A_{16}/S_3

- These lines are multiplexed unidirectional address and status bus.
- During T_1, they carry higher-order 4-bit address.
- In the remaining cycles, they carry status signal.

* $\overline{BHE}/S_7$

BHE stands for Bus High Enable

BHE is used to indicate the transfer of data over higher order data bus ($AD_8 - AD_{15}$)

⇒ $\overline{RD}$ (Read)

It is a read signal

active low

o/p signal

⇒ READY

This is an acknowledgement signal from low speed IO devices or memory

★ RESET

- when high, microprocessor enters into reset state and terminates the current activity.
- It must be active for atleast four clock cycles to reset the microprocessor.

⇒ INTR -

↳ NMI (non-maskable interrupt signal)
edge trigerred interrupt

⇒ $\overline{TEST}$

used to test the status of math co-processor 8087.

If low execution continues, or else in wait state.

⇒ MN / $\overline{MX}$

minimum mode
maximum mode.

⇒ $\overline{INTA}$

- Interrupt acknowledgement signal

⇒ ALE

Address Latch Enable

- indicates that valid address is available on bus $AD_0 - AD_{15}$.

* $\overline{DEN}$
→ Data enable signal
→ Used to enable the transciever

* DT/$\bar{R}$
→ Data Transmit / Receive signal
→ Decides the direction of data flow through transcievers

* $\overline{WR}$
write signal - to enable write operations

* HOLD
Hold Acknowledge signal

21/8/17

MODULE - II

ADDRESSING MODES

Sequence

1. Implied ← HALT, RESET, CLEAR

2. Immediate

MOV AL, 02H AL ← 02

MOV AX, 0102H, AH ← 01, AL ← 02

3. Register

MOV AX, BX ; AX ← BX

4. Direct

MOV BX, [5000H]

EA = 10 × DS + [5000]

5. Register indirect

MOV AX, [BX]

BX = 5000H

EA = 10 * DS + [BX]

Bx | SI | DI

6. Indexed

MOV AX, [SI]

SI | DI

EA = 10 * DS + SI

10 * ES + DI

7. Relative Based

MOV AX, 50H [BX]

EA = 10 * DS + [BX | BP] + 50

8) Based Indexed

MOV AX, [BX][SI]

BP DI

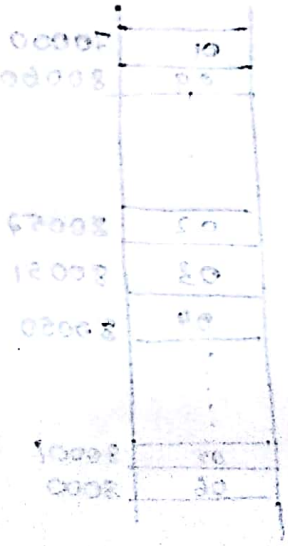
EA = 10 * DS + [BX | BP] + [SI | DI]

9) Relative Based Indexed

MOV AX, 20H [BX][SI]

EA = 10 * DS + 20H + [BX | BP] + [SI | DI]

non-sequence

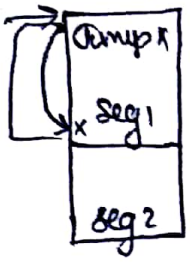


b) Relative Jump

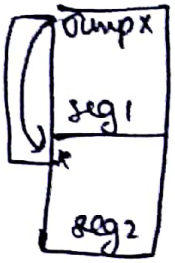
Jump 500H

EA = 10 * CS + [IP] + 500

Intra segment



Inter segment



a)

CS : 7000

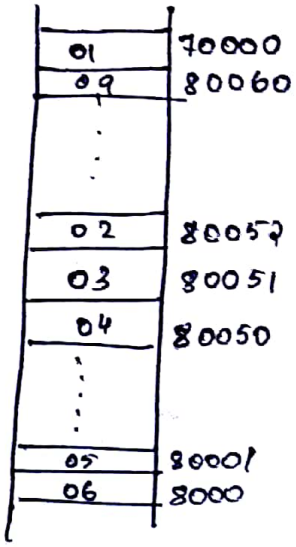
DS : 8000

SI : 0050

DI : 0040

BX : 0052

IP : 0000



1) MOV AX, BX Register

AX ← 0052

2) MOV AX, [0001]

10 * DS + 0001

80000 + 0001

= [80001]

AX ← 05

3) MOV AX, [BX]

10 * DS + [BX]

10 * 8000 + 0052

80052

AX ← 02

4) MOV AX, 08[BX]

10 * DS + [BX] / [BP] + 50

= 10 * 8000 + 0052 + 08

= 800060

AX ← 09

5) CALL 60H

10 * CS + [IP] + 50

10 * 7000 + [0000] + 50

70050

6) 70020 : Jump 60 H

70021 :

70022 :

.

.

.

.

23/8/2017 Wednesday

Instruction set of 8086

Classification of Instruction set

→ data transfer Instructions
→ Arithmetic Instructions
→ Logical Instructions
→ Branch Instructions
→ String Instructions
→ Flag manipulation
→ Process control instructions

Data transfer Instruction

★ MOV Des, Src

- Both Des and source cannot be memory location at the same time.

★ PUSH operand

- pushes the word into top of stack

★ POP Des

Pops the operand from top of stack to Des.

★ XCHG Des, Src

- This instruction exchanges Src with Des.
- cannot exchange 2 memory locations directly.

★ XLAT

- Equals MOV AL, [AL][BX]

AL ← 10*DS + [AL] + [BX]

* IN Accumulator, Port Address:
→ transfers the operand from specified port to accumulator register.

* OUT Port Address, Accumulator
- It transfers the operand from accumulator to specified port

* LEA Register, src
loads a 16-bit register with the offset address of the data specified by src.

* LDS Des, src

* LES Des, src

* LAHF

* SAHF

* PUSHF

* POPF:

Bit manipulation OR Logical Instructions

- NOT src
- OR Des, src
- XOR Des, src
- TEST Des, src
- SHL/ SAL Des, count
- SAR Des, count

29/8/17 Tuesday

ROL Des, Count (without carry)

ROR Des, Count

RCR/RCL Des, Count

Branch Instructions

CALL Des

RET

JMP Des

Jxx Des

STC

CLC

CMC

STD

CLD

machine Control

WAIT

HLT

NOP

LOCK

ESC

String Instructions

Load SB/ LOAD SW

EMPS Desc

CMP

MOVS/MOVSB/MOVSW

Arithmetic Instruction

CMP	DIV
MUL src	IDIV
IMUL src	CBW
	CWD

AAA (ASCII Adjust After Addition)

Step 1: Clear AH, ie AH = 00

Step 2: Move the value to be converted to AL register.

Case 1

If the lower nibble of AL (first digit in number) < 9 and auxiliary carry flag; AF = 0

then set the upper nibble of AL to 0

Case 2

If the lower nibble of AL > 9,

then ADD 06 to AL, set upper nibble of AL to 0

Increment AH

Case 3

Lower nibble of AL < 9 & AF = 1

then add 06 to AL, clear upper nibble of AL,

Increment AH.

Q) 09 + 05 →

```
  0000  1001
  0000  0101   (OE)
  ----------
  0000  1110
  0000  0110
  ----------
  0001  0100
```

0000 0001 | 0000 0100

* 05 + 05

```
0000  0101
0000  0101   0A
-----------
0000  1010
0000  0110
-----------
0001  0000
```

AH: 0000 0001 | AL: 0000 0000

* 08 + 08

```
0000  1000   80
0000  1000
-----------
0001  0000
0000  0110
-----------
0001  0110
-----------
0000  0110
0000  0110
-----------
0000  1100
0001  0110
-----------
0000  0110
```

0000 0001 | 0000 0110

Decimal Adjust after Addition

The value to be corrected must be in AL

If the lower nibble of AL > 9

add 06

then check the upper nibble of AL > 9

add 60

otherwise do nothing

* 49 + 23

```
0100   1001
0010   0011
-------------
0110   1100
0000   0110
-------------
0111   0010
```

72

Q) Assume that the capacity of AL registers is finite and the value in AL is 9C4A. Then what is the result after DAA Instruction

Ans 9 C 4 A

```
1001   1100   0100   1010
                      0110
-------------------------
1001   1100   0101   0000
0                     0
-------------------------
1010   0010   0101   0000
0110   0      0      0
-------------------------
1 0000 0010   0101   0000
```

Result: 10250

Q) For addition operation the user entered two values 09 & 05 through keyboard. Then, what will be the result in HEX format unpacked BCD or packed decimal

30/8/17

8086 Programming

```
ASSUME CS: Code   DS: Data   SS: name
                  Int Data=1
Data    DB   01H
Data1   DB   Dup(?)
msg     DB   "Hello's"
```

Q) Write a 8086 Assembly level to print "Helloworld"

Ans

```
ASSUME DS: DATA  CS: CODE
DATA SEGMENT
     MESSAGE DB "HELLO WORLD!!!$"
ENDS
CODE SEGMENT
     START: MOV AX,DATA
            MOV DS, AX
            LEA DX, MESSAGE
            MOV AH, 09H
            INT 21H
            MOV AH, 4CH
            INT 21H
CODE ENDS
END START
```

Q) Print string using Macro

A

```
ASSUME DS: DATA CS: CODE
DATA SEGMENT
    MESSAGE DB "HELLO WORLD!!!$"
     display MACRO xxx
        LEA DX, xxx
        MOV AH, 09H
        INT 21H
       ENDM
ENDS
CODE SEGMENT
      START: MOV AX, DATA
             mov DS, AX
             display MESSAGE
             mov AH, 4CH
             INT 21H
CODE ENDS
 END START
```

Q) Print String (subroutine)

```
ASSUME DS: DATA CS: CODE
DATA SEGMENT
    MESSAGE DB "HELLO WORLD!!!$"
ENDS
CODE SEGMENT
      START: MOV AX, DATA
             mov DS, AX
           CALL display MESSAGE
             MO
```

```
display PROC yyy
  (PUSH all registers)
  LEA DX, yyy
  MOV AH, 09H
  INT 21H
  (POP all registers)
RET
ENDP
    MOV AH, 4CH
    INT 21 H
CODE ENDS
 END START
```

Q) Write a Program to add two immediate values 02 and 03.

```
Assume cs: code
Code Segment
Start: MOV al, 02 ┐
       MOV cl, 03 ├ mov bl, 02
       add bl, cl ┘ add bl, 03
Code Ends
 End
```

Q) WAP to add 2 no:s located at offset address 5000 and 6000 and store the result to offset address 7000

Ans

```
Assume cs:code
code segment
start: mov bl, [5000]
       add bl, [6000]
       mov [7000], bl

code Ends
End
```

Q) WAP to move 10 no:s located at the offset address, 5000 to 5009 to another offset 6000 to 6009 respectively

Ans

```
ASSUME cs: code
code segment
start: mov cl, 0ah
       mov si, 500
       mov di, 6000
       mov bl, [si]
       mov[di], bl
       INC SI
       INC di
       DEC cl
       JNZ x1
       Ends
```

Q) WAP to find sum of first ten numbers

```
ASSUME   CS:CODE
CODE SEGMENT
START   MOV CL,0A11
        MOV, BL,01
        MOV BH,02
        MOV b
XX:     ADD BL,BH
        INC BH
        DEC CH
        JNZ XX
```

Q) A initialized variable number is even or odd?

```
ASSUME  CS:CODE  DS:data
Data segment
  num  DB 05h
       add DB "number is odd$"
 Ends
 code segment
   Start: mov ax,data
          mov ds,ax
          mov al,num
          SHR
          JNC XX
           display odd
       XX: display even
       Y: End
```

Q) WAP to count the number of one's in a 8 bit number

Ans ASSUME CS:CODE DS:data

multiprocessor

Q) Three processors connected with 8086 ~~config~~ in a master-slave configuration. In this configuration, 8086 processor transmitting a data to 2050H (memory location of another process)

i) Identify the mode of operation of 8086. ~~Identify~~

ii) Identify the type of operation

iii) Draw the timing diagram

Ans i) Maximum mode

ii) ~~Write~~ memory-write

Q) write assembly program to find the factorial

Ans

```
Assume CS:Code DS:DATA
DATA SEGMENT
        INC DB 01H
Ends
Code SEGMENT
start:

Assume CS:Code
Code SEGMENT
        start: MOV AL,05h
               MOV CL,01h
               MOV bl,01h
        Loop:  MUL bl,cl
               INC cl
               CMP cl,al
               JLE Loop
```

0 1 1 2 3 5

```
mov dl,bl
mov al,02h   } print bl
int 21h
Code Ends
End
```

Q) Write a program to print the fibonacci series, (first 6 fibonacci numbers)

```
Assume cs:code
Code SEGMENT
    start: mov
```