PSG COLLEGE OF TECHNOLOGY, COIMBATORE - 4. F 201028

16 Pages

Roll No.	Name of the Student	Signature of the Invigilator
22Z218	K. S. Gayathri	

Branch & Semester		Test No.	Date
BE CSE G1		1	8/8/24

Course Code	Title
19Z502	Microprocessors and Interfacing

Faculty Use	Qn.	a	b		c	Total	Grand Total (Out of .....)	Faculty Signature
	1	4	5	5	10	24	47	
	2	3	5	5	10	23		

INSTRUCTIONS TO CANDIDATES

1. The candidates shall not possess any **handwritten / printed materials / any incriminating materials / Cell Phones / Programmable Calculators / Smart Watches / Calculator Covers** inside the test hall. Possession of any materials / gadgets mentioned above inside the test hall will be treated as malpractice.
2. Malpractice will be viewed very seriously and the **punishment** may be **invalidation of test in one or more courses.**
3. **No additional sheets** will be issued. After completing the test, the candidates shall personally handover the answer booklet to the Hall Superintendent.

I have read all the instructions above

4. I understand and abide by the norms prescribed by the college during assessments.
5. I have neither offered nor received any aid in answering the questions of this test.

K. S. Gayathri
Signature of the Candidate

1)

a) i) d) decremented by 2

ii) a) II, IV

iii) A) 64 KB

iv) unpacked: 0000 1001 0000 1010 packed: 1001 1010

v) 1280 times

2) i) a) MIIO

ii) B) II and III only

iii) d) Stack segment

iv) 0, 4

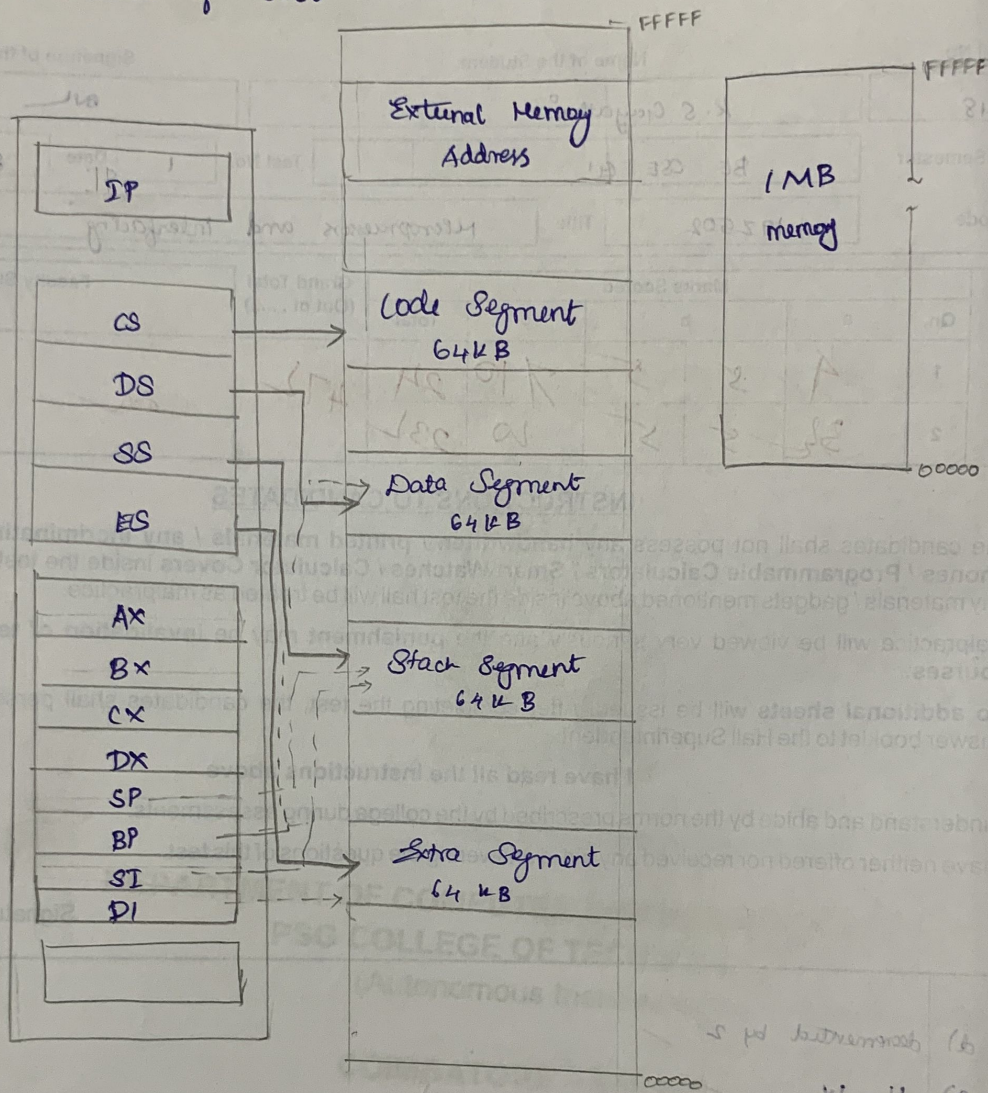
v) 1400 ns
(200×4 + 3×200)

8421
XA
0280 H
0000 0010 1000 0000
1024 256 1280
1366 256 512
0.2 5 | 10 10 0
200×4 800 + 3×200 600
5 MHz (10^6)

b)

(i) 8086

Software Architecture of 8086



Register organisation

(*) General purpose Registers (AX, BX, CX, DX)

⇒ AX Accumulator Register

* 16 bit Register
* AL, AH ⇒ Each 8 bit
* used for doing calculation/operations & for storing temporary results

⇒ BX Base Register

* 16 bit register
* BL, BH ⇒ Each 8 bit
* Base register is used as an offset with Data Segment.

It used to store offset address PA : DS : BX

⇒ CX Counter Register

* 16 bit register
* CL, CH ⇒ Each 8 bit
  - CL ↓ lower 8
  - CH ↓ higher 8 bit
* counter Register used for loop.
* It loops until CX = 0

⇒ DX Data Register

* 16 bit Register
* DL, DH ⇒ Each 8 bit
* Data is stored. used for division, Multiplication for storing remainder, I/O addressing

(1) Segment Registers

Code Segment

* 64K byte size
* code segment contains Program code
* used to address location to access code in memory.

PA : CS : IP ↳ points to next instruction in memory
It is an offset.

Data Segment

* 64K byte size
* Data for operations are stored

* used to access location of data stored in DS

PA : DS:BX
↳ contains offset address
PA : DS:SI (offset)
↳ Source Index

Stack Segment

* 64K Byte size
* used to store the data, parameters in Stack
* 2 operation - PUSH, POP
* push decrement stack pointer by 2, pop increment it by 2
* PA : SS:SP
↳ points to top of Stack (offset)

PA : SS:BP ↳ points to access any parameter in Stack (offset)

*

Extra Segment

* 64K Byte size
* used as another data segment for storing data

PA : ES:DI
↳ destination Index points to dest address

Flag Registers

* 16 bit
* contains 9 flag bits

x	x	x	x	OF	DF	IF	TF	SF	ZF	x	AF	x	PF	x	CF

- OF: Set when result is not able to fit in available bits
- DF: Set, auto decrement SI, DI
- IF: Set when interrupt
- TF: Execute one line of operation used for debugging
- SF: Set when MSB = 1
- ZF: Set when result = 0
- AF: Set when carry from low nibble to high nibble
- PF: Set when even no of 1s
- CF: carry flag Set when there is carry

CPA

(II) Types of Addressing Modes

(i) ~~Direct~~ Immediate Operand Addressing mode

Here, operand is available directly in the instruction. can be accessed directly.

Eg: MOV AX, 100H
↳ operand

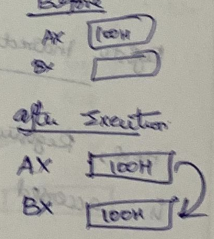
100H moved to AX

(ii) Register Operand Addressing mode

operand from one register (content) is copied to destination register

Eg: MOV AX, BX
dest source

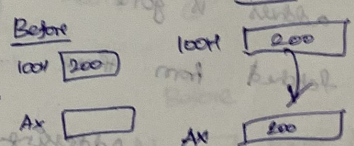
Here, content of BX moved to AX register



(iii) Direct operand Addressing mode

operand is available in the address specified in the instruction

Eg: MOV AX, [1000H]



Here, operand at 1000H is moved to AX

(iv) Implied addressing mode

operand is specified in the instruction implicitly

Eg: CLD // clear directional flag

Register Indirect addressing mode

$$PA = \begin{Bmatrix} CS \\ DS \\ SS \\ ES \end{Bmatrix} : \begin{Bmatrix} BX \\ BP \\ SI \\ DI \end{Bmatrix} + \begin{Bmatrix} 8/16\ bit \\ displacement \end{Bmatrix}$$

$$\text{Effective addr} = \begin{Bmatrix} BX \\ BP \\ SP \\ SI \\ DI \end{Bmatrix} + \begin{Bmatrix} 8/16\ bit \\ displacement \end{Bmatrix}$$

Register Indirect Addressing mode

Here, Register contains the operand address. Operand address is accessed and from there it is taken

Eg:

MOV AX, [BX]

BX has 2000H. So 2000H address is gone & data is fetched from there.

		Addr	Memory
AX			
BX	2000H	2000H	50

After Execution

AX 50

Base Relative Addressing Mode

Here, BX / BP along with displacement is used as offset for operand fetch

PA : DS*: BX + Displacement

PA : SS*: BP + Displacement

Eg: MOV [BP]+1000H, CL

Value of CL moved to [BP]+1000H

Eg: SS: 1000H, BP: 1500

10000 (Shift left SS)
1500
1000 (+)
12500

destination → 12500

Before

CL ⇒ 100H 12500

After Exec

12500 100H

Index Relative Addressing Mode

SI, DI is used as offset for operand fetch along with displacement

PA: DS*: SI + DISP

PA: ES : DI + DISP

Eg: MOV CL, [SI]+1000H

DS: 1000H
SI = 1000H

10000 (Shift left)
01000
11000 (+)
1000
12000

source → 12000

Before

CL 12000 50

After

CL 50 12000 50

Base Indexed Addressing Mode

Here, Base Pointer, Index pointer along with offset is used for operand addressing

Eg: MOV AL, [BX]+[SI]+100H

DS: 1000H,
SI: 500H,
BX: 300H.

PA: 10000 (DS)
500 (SI) (+)
300 (BX)
100
10900

destination → 10900

Before

AL 10900 50

After

AL 50 10900 50

(b)

(i) 8088, 8086 - Min, Max mode

8088 has only 8 bit data bus, 8086 has 16 bit data bus

↳ AD_0 to AD7

↳ AD_0 - AD15

↳ SSO - Status Signal output

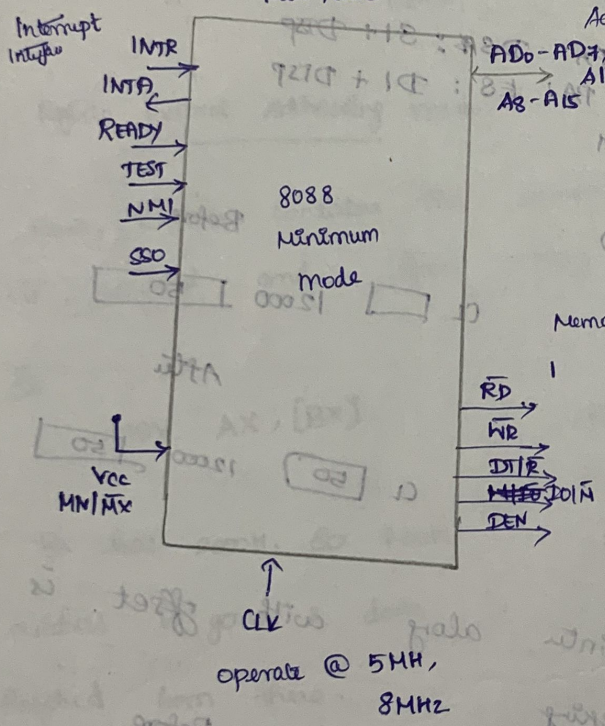
↳ Bus High Enable - BHE

8088 can only read 1 byte in 1 machine cycle

but 8086 read 2 byte (1 word) in 1 machine cycle.

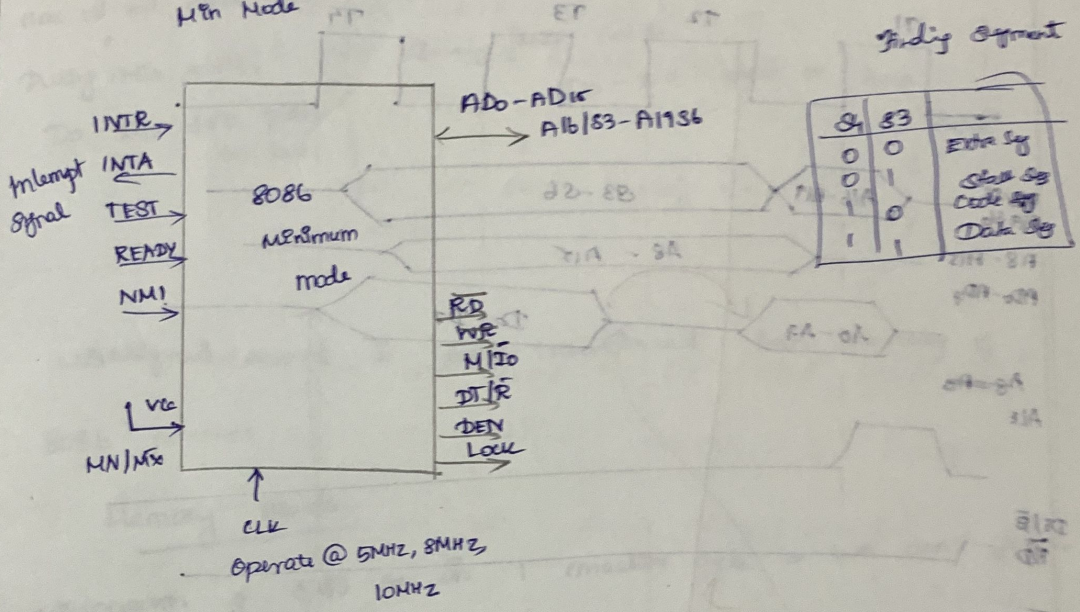
8088 Block diagram

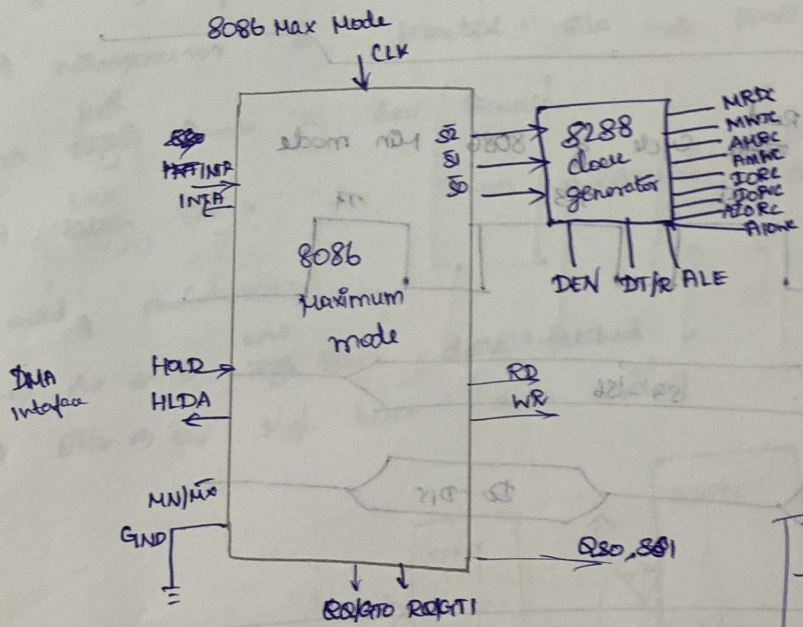
Min Mode



operate @ 5MH, 8MHz

8086 Block Diagram





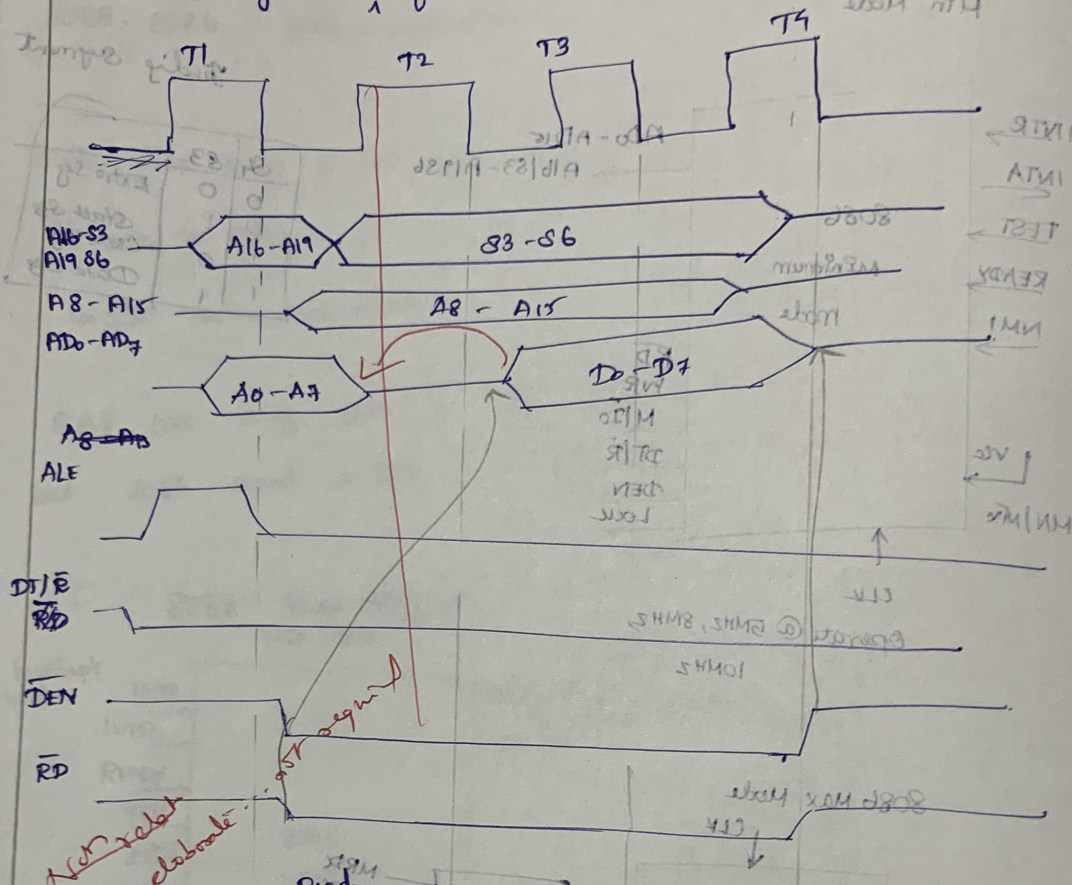
Instruction Queue

QS	QS0	
0	0	No operation
0	1	First Byte Fetch
1	0	Queue Empty
1	1	Next Byte Fetch

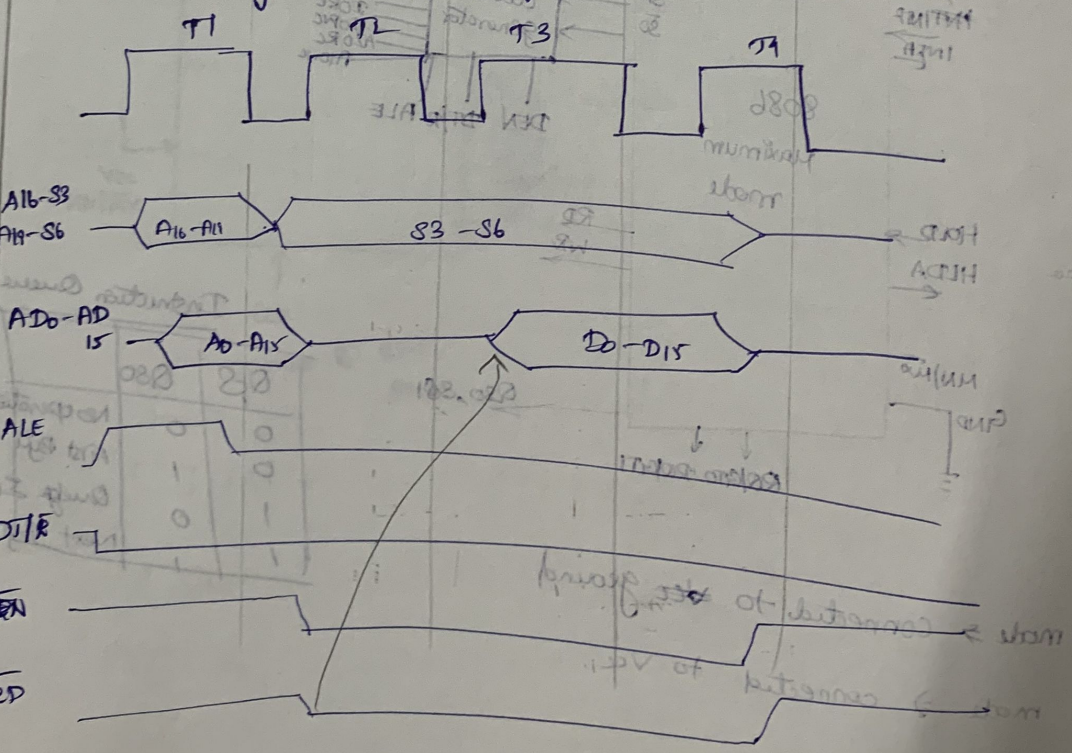
Max mode ⇒ connected to ~~VEE~~ ground

min mode ⇒ connected to Vcc

Memory Read Cycle of 8088 Min mode



Memory Read Cycle of 8086 Min mode



8086	8088
Has 16 bit data bus	8 bit data bus
During Min mode, D0 - D15 data place on bus	During Min mode D0 - D7 place on bus

$\overline{BHE}$	A0
0	0
1	0
0	1
0	1
1	0

(1) Misaligned word is transferred from memory of 8086 processor

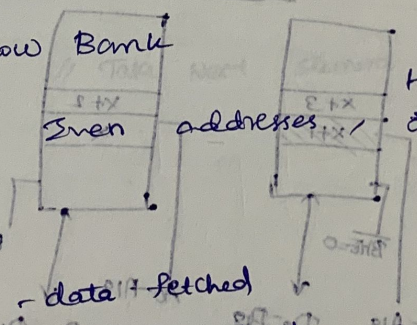
Memory Bank

⇒ Microprocessor of 8086 ⇒ in 1 machine cycle ⇒ we can fetch 2 byte of data

⇒ Microprocessor is bisected into 2 parts of 512k bytes

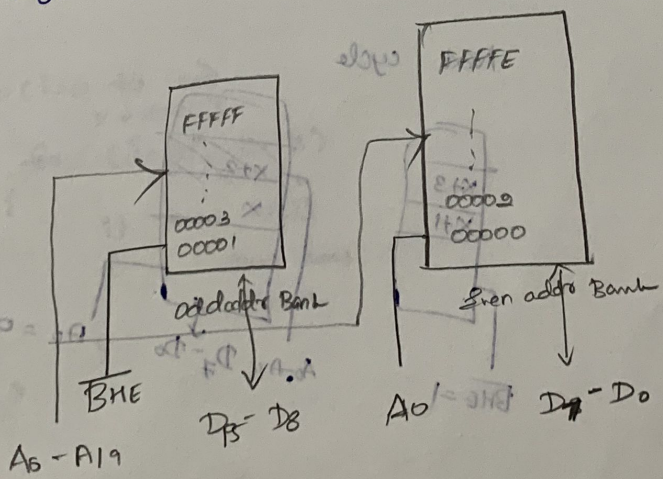
⇒ High Bank & Low Bank

⇒ Low Bank for Even addresses / High Bank has odd addresses



⇒ A0 ⇒ 0, Low Bank - data fetched

⇒ $\overline{BHE}$ ⇒ 0, High bank - data fetched



A_0	$\overline{BHE}$		Machine cycle Req
0	0	16 bit word (1 word fetched) (Even addr Bank	1
0	1	8 bit (byte) fetched from even addr Bank	1
1	0	1 byte fetched from odd memory Bank	2
1	0	1 word fetched from odd addr bank	
0	1		

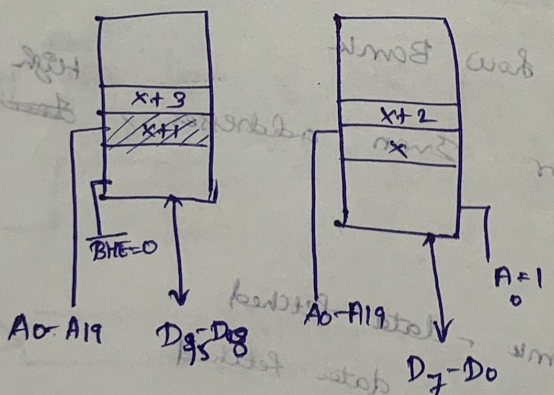
Misaligned word ⇒ 1 word from odd addr Bank.

Required 2 Machine cycle

⇒ In first Machine cycle $D_8 - D_{15}$ odd addr data fetched

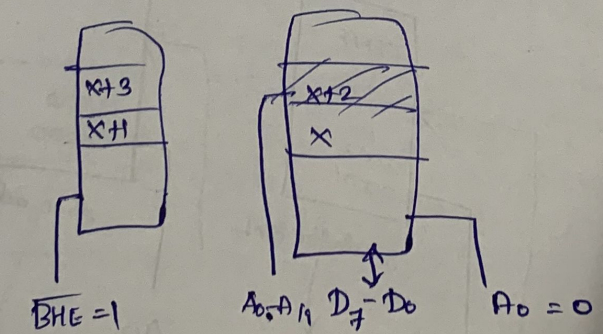
⇒ In next machine cycle, $D_0 - D_7$ Even addr data fetched

First cycle



$D_{15} - D8 \Rightarrow X+1$

Next cycle



$D_7 - D_0 - X$

17

(c)

~~MOV SI, 2500~~

```
MOV  SI, 2000
MOV  CX, 2500

     MOV  AX, [SI]
NXTPT2: ADD  SI, 2     // Take Each element i
     MOV  BX, [SI]     // Take next element i+1
     MOV  DL, CX       // Store the counter value -1 so that remaing element
     SUB  DL, 1        can be iterated
NXTPT: CMP  AX, BX     // i, i+1 is compared
     JE   2700H        // if Equal, loop is broken, goes out & ans stored
     ADD  BX, 2        // or move to next element i+2 etc
     DEC  DL
     LOOP NXTPT
LOOP NXTPT2            // Take Next element from outer loop.

2700H: MOV [2500], AL
     MOV [2503], AH
```

Logic

```
for (i=0 to n-1)
{
    x = arr[i]
    for (j=i+1 to n)
    {
        if x == arr[j]
        {
            Store ans
            Break
        }
    }
}
```

(c)

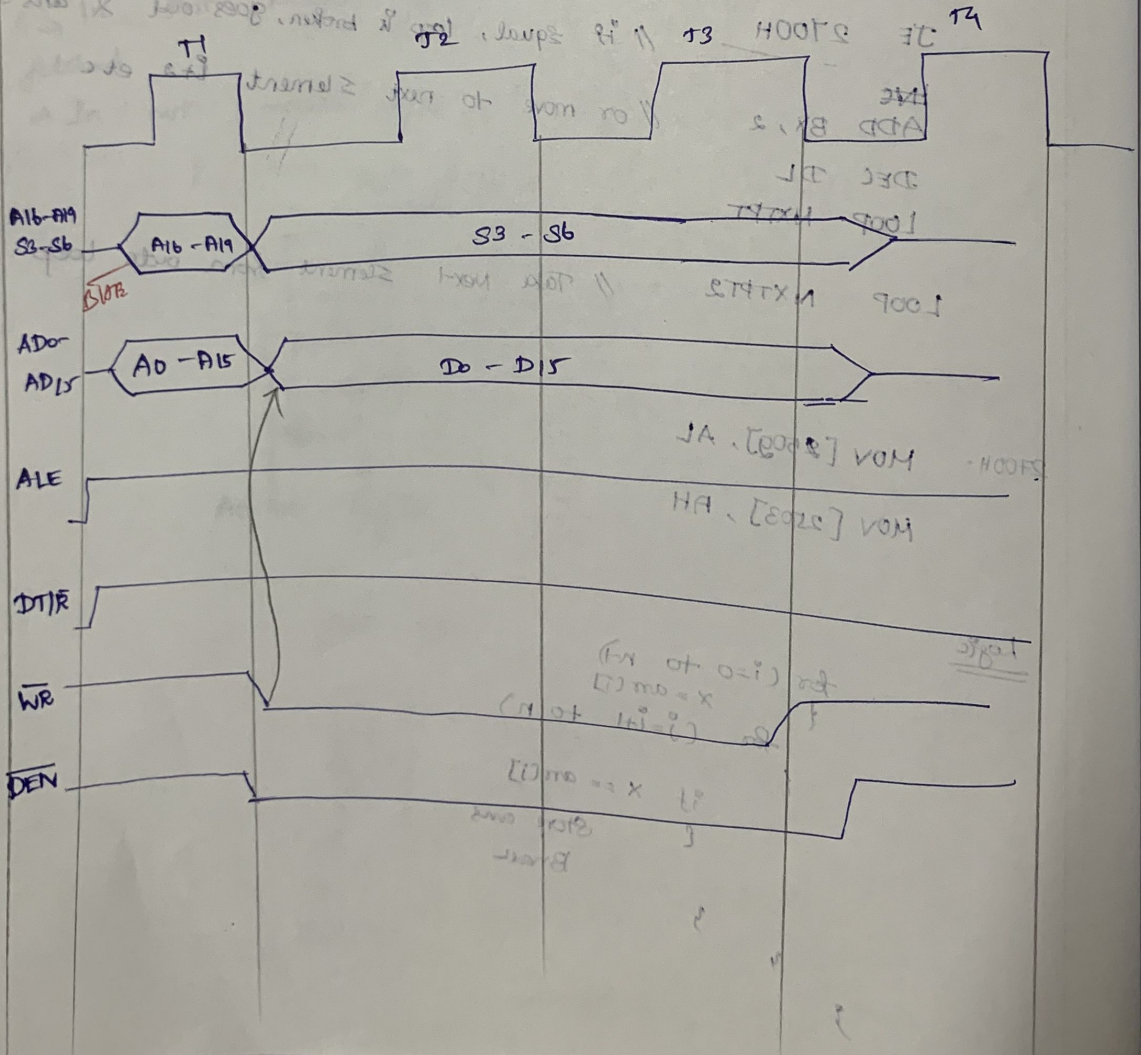
(i) $45_{16} \Rightarrow 0A00D_{16}$

Byte is written to memory 0A00D which is in minimm mode.

Type of bus cycle:

writing to memory

Min Mode Write cycle of 8086



Min Mode - data is written to memory.

So, Min Mode write cycle used.

At time T1,

Address is put on the address bus.

Address bus contains address of memory to be written

At time T2,

data to be written is put on data bus. The data here is byte 45h

Here, it will be put on the data bus.

At time T3, TA, data is maintained in the data bus.

⇒ When ALE = 1,

AD A0 - A15 ⇒ address is passed

ALE = 0 D0 - D15 ⇒ data is passed

DT/R ⇒ Data Transfer happens here.

So, DT = 1

WR ⇒ write happens, WR = 1