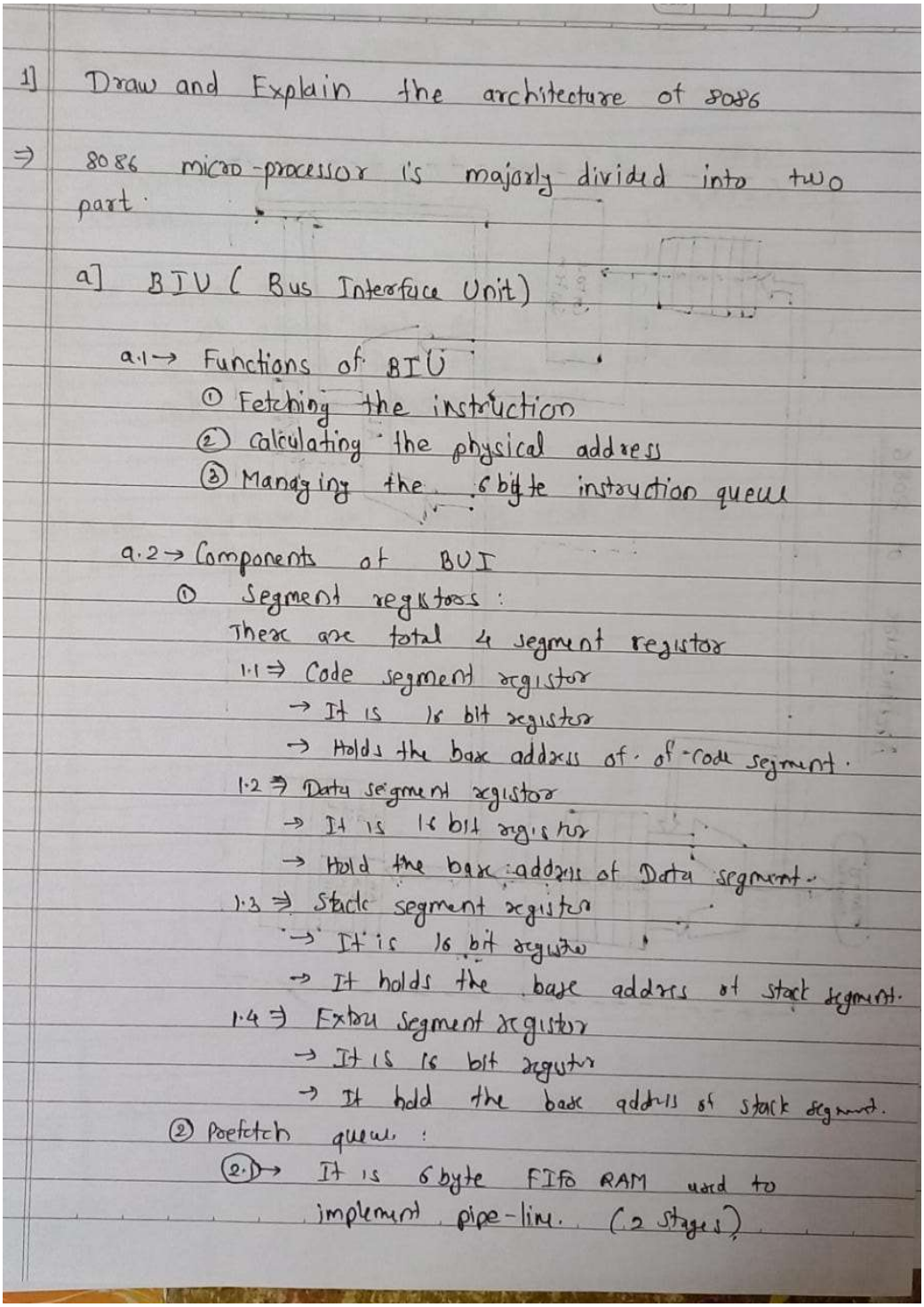
Assignment

1.Draw and explain the architecture of 8086.[10M]



1] Draw and Explain the architecture of 8086

⇒ 8086 micro-processor is majorly divided into two part:

a] BIU (Bus Interface Unit)

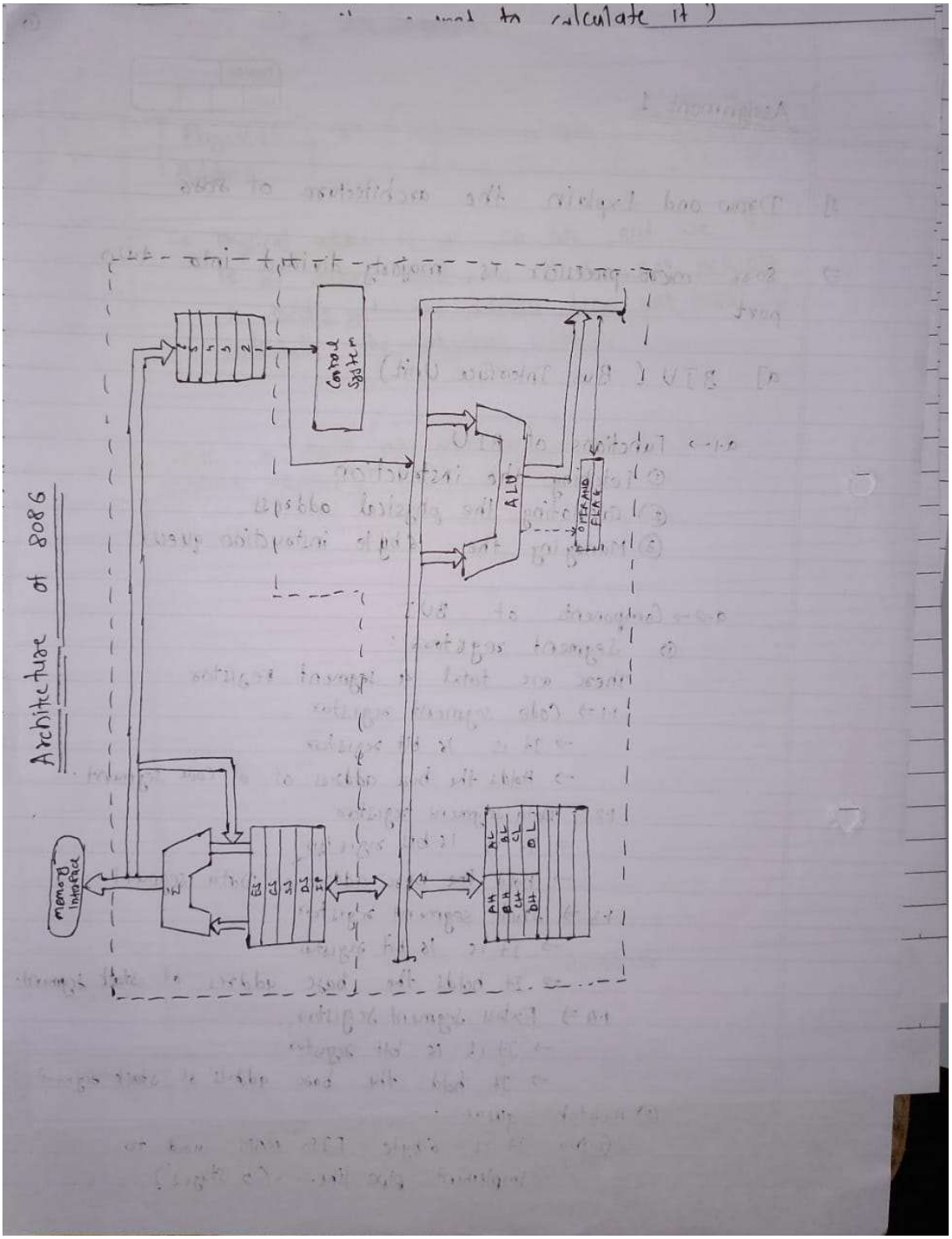
a.1→ Functions of BIU

① Fetching the instruction
② Calculating the physical address
③ Managing the 6 byte instruction queue

a.2→ Components of BUI

① Segment registers:
There are total 4 segment registor
1.1⇒ Code segment register
→ It is 16 bit register
→ Holds the base address of of code segment.
1.2⇒ Data segment register
→ It is 16 bit register
→ Hold the base address of Data segment.
1.3⇒ Stack segment register
→ It is 16 bit register
→ It holds the base address of stack segment.
1.4⇒ Extra segment register
→ It is 16 bit register
→ It hold the base address of stack segment.
② Prefetch queue:
2.1→ It is 6 byte FIFO RAM used to implement pipe-line. (2 stages)

Ans.


Architecture of 8086
Control System
ALU
FLAG
memory interface
ES
CS
SS
DS
IP
AH
AL
BH
BL
CH
CL
DH
DL

(2.2) → BIU Fetches the next 6 instruction byte from the code segment and store it into queue.

(2.3) → EU takes these instruction and executs them.

(2.4) → The queue is filled only when at least two bytes are empty and 8086 has 16 bit of data bus.

(2.5) → These pipelining fail when branching occurs and the queu has to flush, completely and fill it up again from branched location.

(3) Instruction pointer

→ It is a 16 bit register

→ It holds off set of next instruction in the code segment.

(4) Physical address calculator

→ It generate the 20 bit physical address using segment and offset address using formula.

$$\text{Physical address} \Rightarrow (\text{segment} \times 10H) + (\text{offset address})$$

2.explain pre-fetch queue of 8086 [2M]

2] Explain pre-fetched queue of 8086

⇒ ① It is 6 bytes FIFO RAM used to implement pipelining (2 stages)

② BIU Fetches the next 6 instruction bytes from the code-segment and store in into queue

③ Execution unit removes instruction from this queue and execute them

④ The queue is refilled when atleast two byts are empty as 8086 has 16 bit of data bus

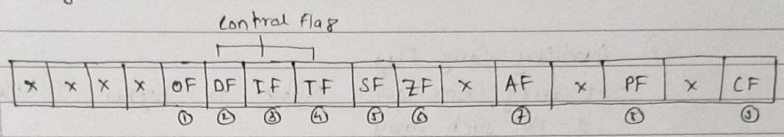
⑤ Pipelines fail, fails when branching occurs because the already fetched instruction are of no use

⑥ In such queues need to be flushed and instruction need to be re-fetched.

3.Draw and explain flag register of 8086.[10M]

3] Draw and explain flag register 8086

⇒ ① Following is the diagram representing the Flag register in 8086



There are total 9 Flag in 8086 microprocessor.

① Overflow Flage
→ 1 ⇒ when overflow occurs
0 ⇒ when there is no-overflow

② Direction Flag
→ 1 ⇒ SI and DI are auto-decremented
0 ⇒ SI and DI are incremented

③ Interrupt Flag
→ It is used to mask (disable) or unmask (enable) the intr interrupt.

④ Trap Flag
→ It is used to set trace mode (single stepping mode) Here, up is interrupted after every instruction so that the program can be debugged

⑤ Sign Flag
→ MSB of result in 1 when int is negative else it is 0

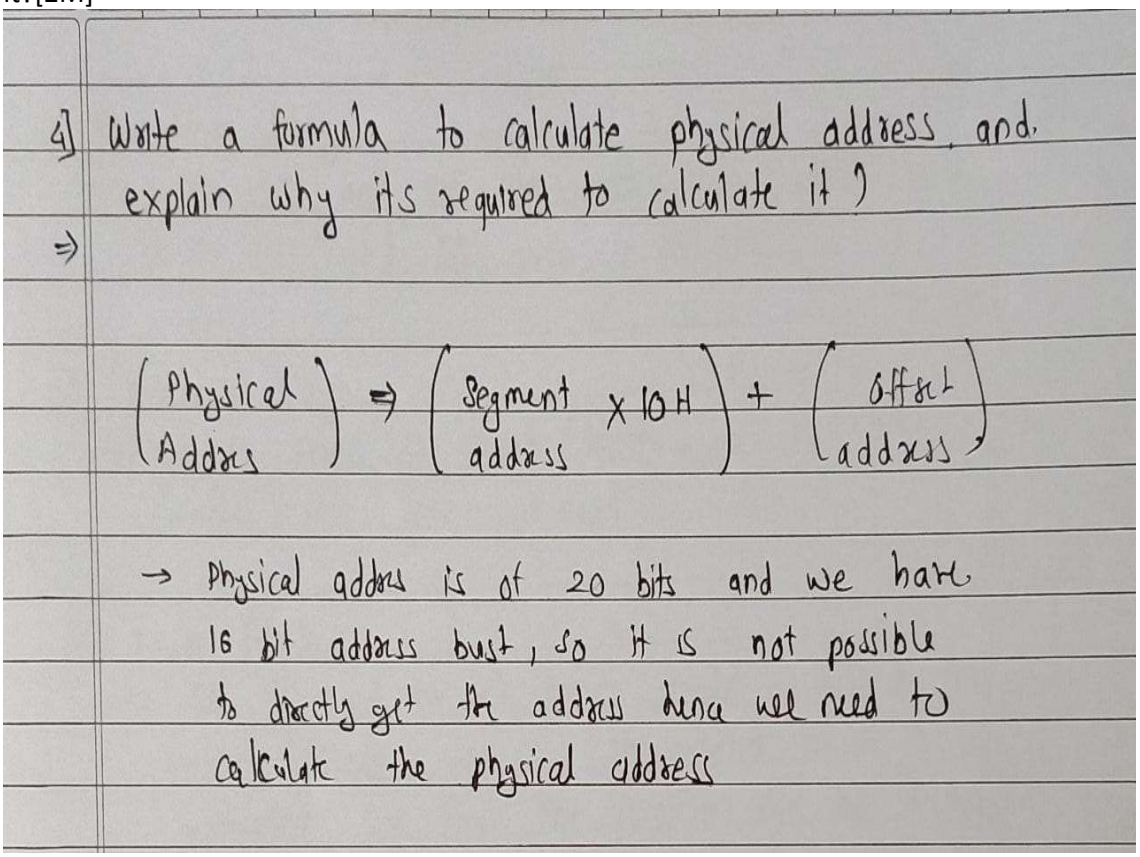
⑥ Zero Flage
→ when result is 0

⑦ Auxiliary Carry Flag
→ 1 = Carry from lower Nibble to Higher Nibble
0 = No such carry

⑧ Parity Flag
→ 1 = Even Parity
→ 0 = odd Parity

⑨ Carry
→ 1 = Carry is generated
0 = No carry is generated

4.Write a formula to calculate physical address. And explain why its required to calculate it?[2M]



5. Write a note on addressing modes of 8086. [10M]

Q.5.

→ The different addressing modes of 8086 are as follows -

① Immediate addressing - In this data operand is part of the instruction itself
eg. MOV CX, 2000H

② Register addressing mode - In this the effective address of memory location is written directly in instruction.
eg. MOV AX, [2005H]

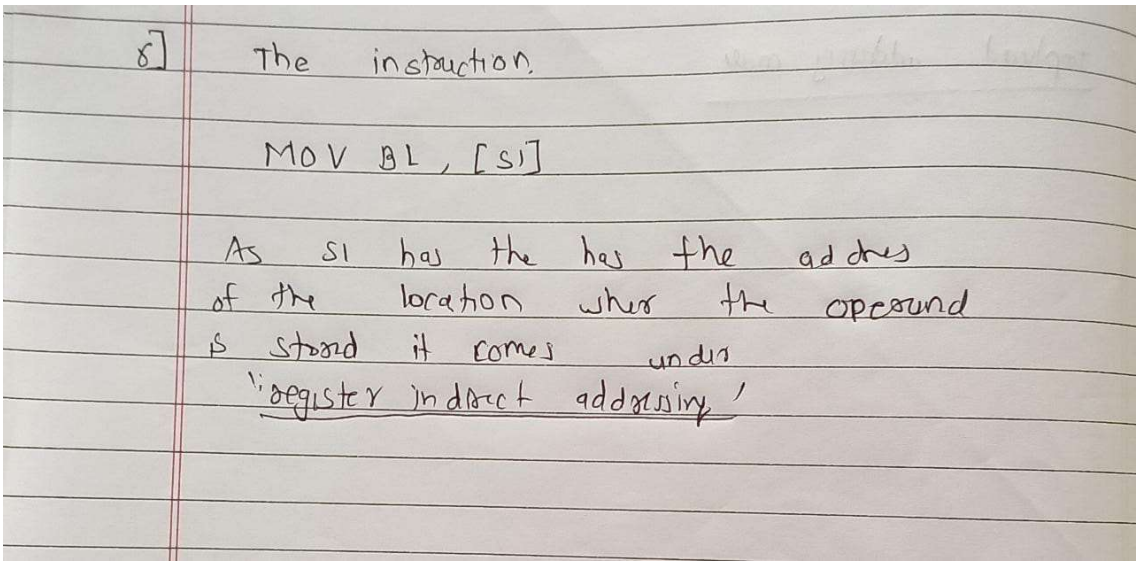
③ Register indirect addressing - In this mode, memory location can be addressed by offset address saved in registers
eg. MOV AX, [BX]

④ Based addressing mode - In this mode, offset address of operand is given by sum of contents of BX/BP registers
eg. MOV DX, [BX+04]

⑤ Indexed addressing modes - In this mode offset address is found by adding contents of SI or DI and 8/16 bit displacements.
eg. MOV BX, [SI+16]

⑥ Based - index addressing mode - In this mode, offset address of operand is computed by summing base register to contents of index register
eg, ADD CX, [AX+SI].

6. The instruction ‘MOV BL,[SI]’ comes under which type of addressing mode [2M]



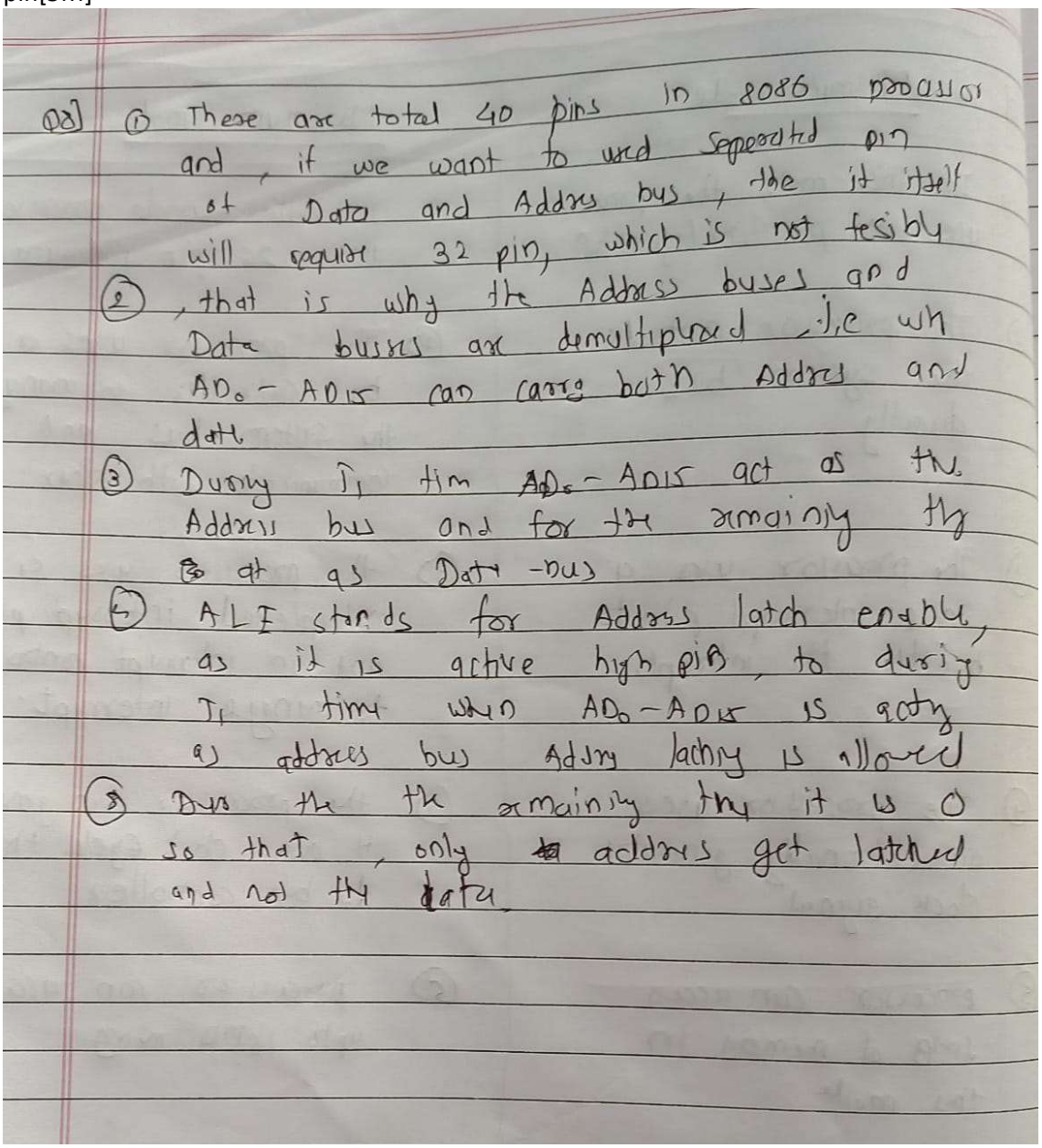
6] The instruction

MOV BL, [SI]

As SI has the has the addres of the loration wher the operand is stord it comes under "register indirect addressing"

7.Design the power on reset and manual reset circuit for 8086 processor. [5M]

8.What is de multiplexing of address and data bus also explain the significance of ALE pin[5M]



9.Why 8284 is needed in 8086 based system?[5M]

PAGE No.
DATE / /

9] Why 8284 is needed in 8086 base System.

→ 8084 clock generator chip is used in an 8086 base system to provide a stable and aurate clock signal for the microprocessor

1) frequency stabilization
8284 provides frequency stabilization to the clock signal generated by an external crystal oscillator. This helps to reduce the effect of temperature & voltage variation on clock signal

2) clock signal timing:
8284 generates several timing signals such as RESET signal & READY signal. that are necessary for proper operation of microprocessor

3) power-up sequencing:
8284 provides power up sequencing which ensures the microprocessor & other components in the system are powered upin correct order.

4) overall 8284 clock generator chip is essential component in 8086 based system providing stable & reliable clock signal, timingsignal, power up sequencing to ensure proper operation of micro processor and other component in the system

10.List Features of 8086 microprocessor[5M]

- Enhanced version of 8085 designed by Intel in 1976
- It is 16 bit µp.
- It has 20 bit address bus so can access upto 2^{20} memory locations.
- 16 bit data lines
- 64K I/O ports
- 14, 16 bit registers
- It has powerful instruction set (multiplication & division)
- It supports two modes of operations
  minimum mode - suitable for single processor
  maximum mode - suitable for multiple processors
- It has instruction queue capable of storing 6 instructions bytes from memory resulting in faster processing.
- It was first 16 bit processor having 16 bit ALU, 16 bit registers, internal data bus resulting in faster processing.
- Available in 3 versions 8MHz, 5MHz, 10MHz.
- Two stages of pipelining : (Fetch + Execute).
- Single phase clock with 33% duty cycle.
- 256 vectored interrupts.

11.What is the memory addressing capacity of 8086 ? and why?{2M]
->The 8086 processor has a memory addressing capacity of 1MB, which means it can address up to 2^20 (1,048,576) unique memory locations.

This memory addressing capacity is a result of the 8086's architecture, which uses a 20-bit address bus to communicate with memory. Each memory address can represent a unique memory location, and with 20 bits, the 8086 can address up to 1MB of memory.

12.Explain memory segmentation in 8086 and list its advantages[10M]

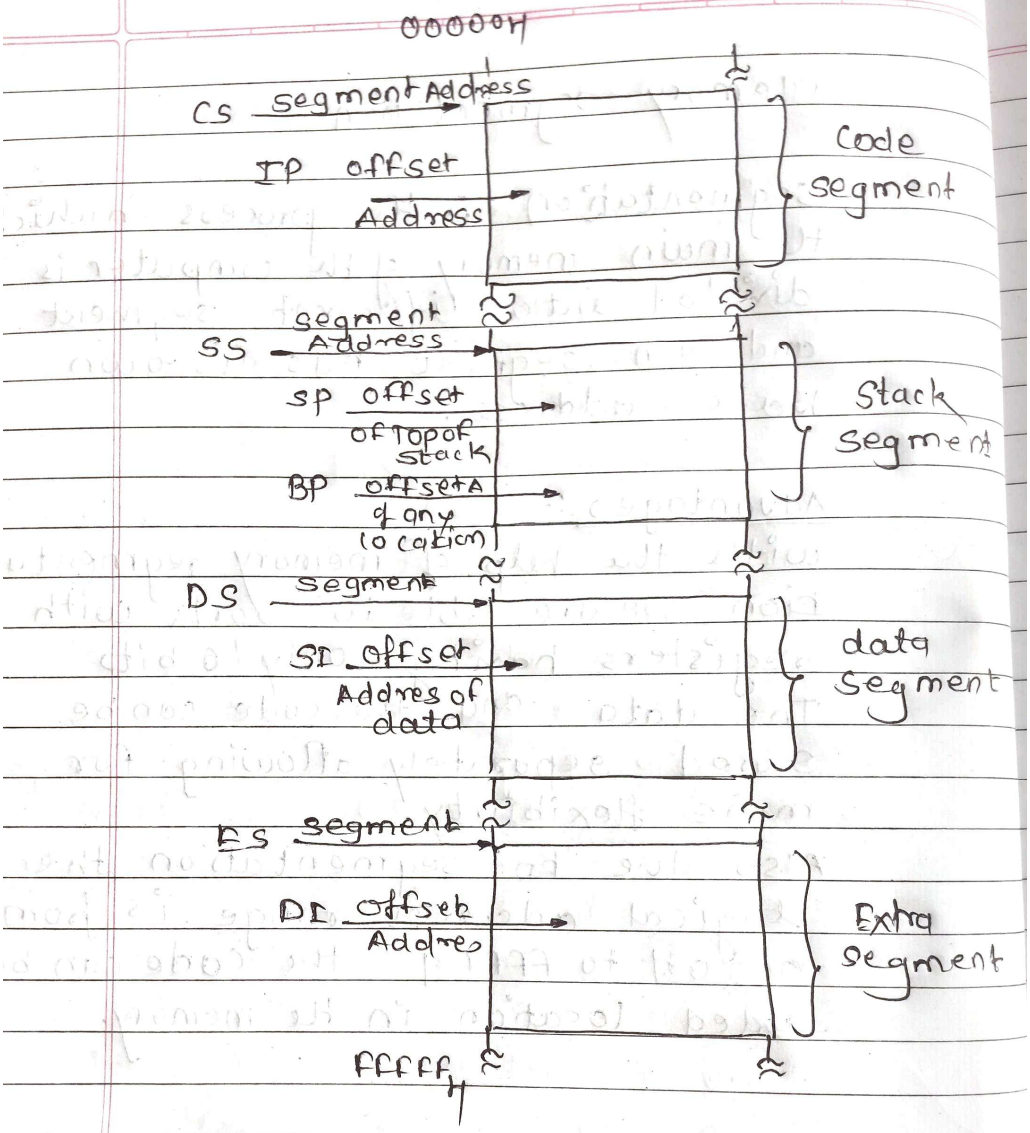
PAGE No.
DATE / /

Memory segmentation

Segmentation is the process in which the main memory of the computer is divided into different segment and each segment has its own base address.

Advantages :

With the help of memory segmentation we are able to work with registers having only 16 bits
The data and the code can be stored separately allowing for more flexibility.
Also due to segmentation the logical address range is from 0000H to FFFFH the code can be loaded location in the memory.


00000H
CS segment Address
IP offset Address
Code segment
SS segment Address
SP offset of Top of stack
BP offset of any location
Stack segment
DS segment
SI offset Address of data
data segment
ES segment
DI offset Addre
Extra segment
FFFFFH

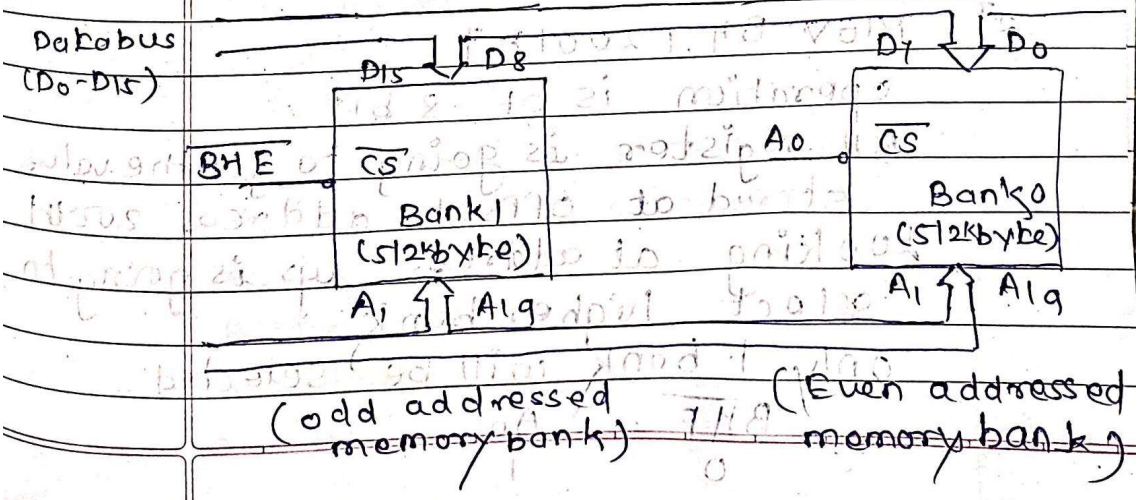
13.Explain memory banking in 8086 system and describe its advantages. [10M]

Memory Banking

The memory address space is equally divided into 2 parts (banks)

one of the banks contain even addresses called Even bank and other contain odd addresses called odd bank.

even bank always stores lower bytes so Even bank is also called Lower Bank (LB) and odd bank is also called a Higher bank (HB)



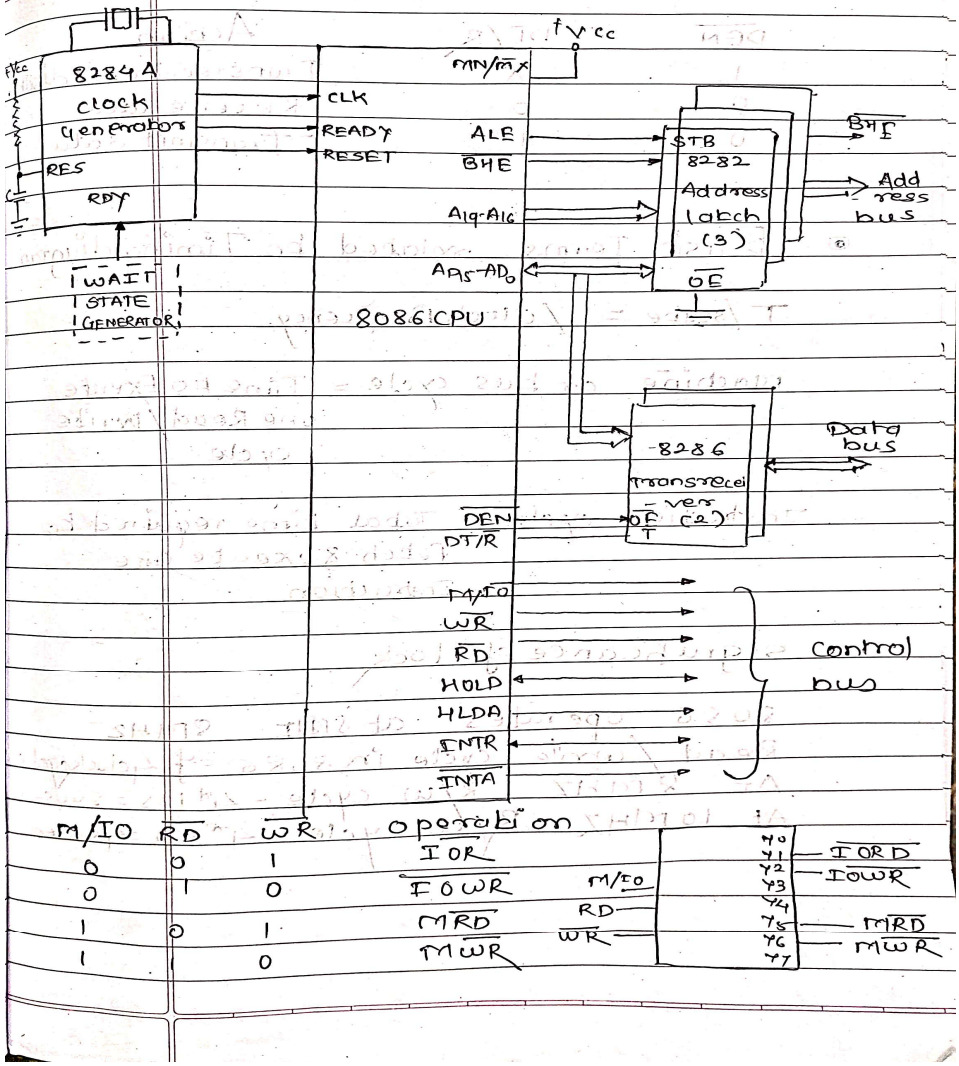
14. Explain minimum mode of operation. [10M]

Minimum mode:

programmed when MN/MX is High clock, Reset, Ready generated by 8284 8086 generates RD#, WR#, M/IO#, ALE, HLDA, HOLD

single processor in the system and that is 8086, does not support additional processors.

MINIMUM MODE CONFIGURATION



M/IO	RD	WR	operation
0	0	1	IOR
0	1	0	IOWR
1	0	1	MRD
1	1	0	MWR

$\overline{DEN}$	DT/R	Action
1	X	Transreceiver is disabl.
0	0	Receive data
0	1	Transmit data

- Basic Terms related to Timing diagram

T/state = 1/clock frequency

machine or bus cycle = Time to Execute one Read/Write cycle

Instruction cycle = Total time required to fetch & execute one Instruction

significance of clock

8086 operates at 5MHz, 8MHz

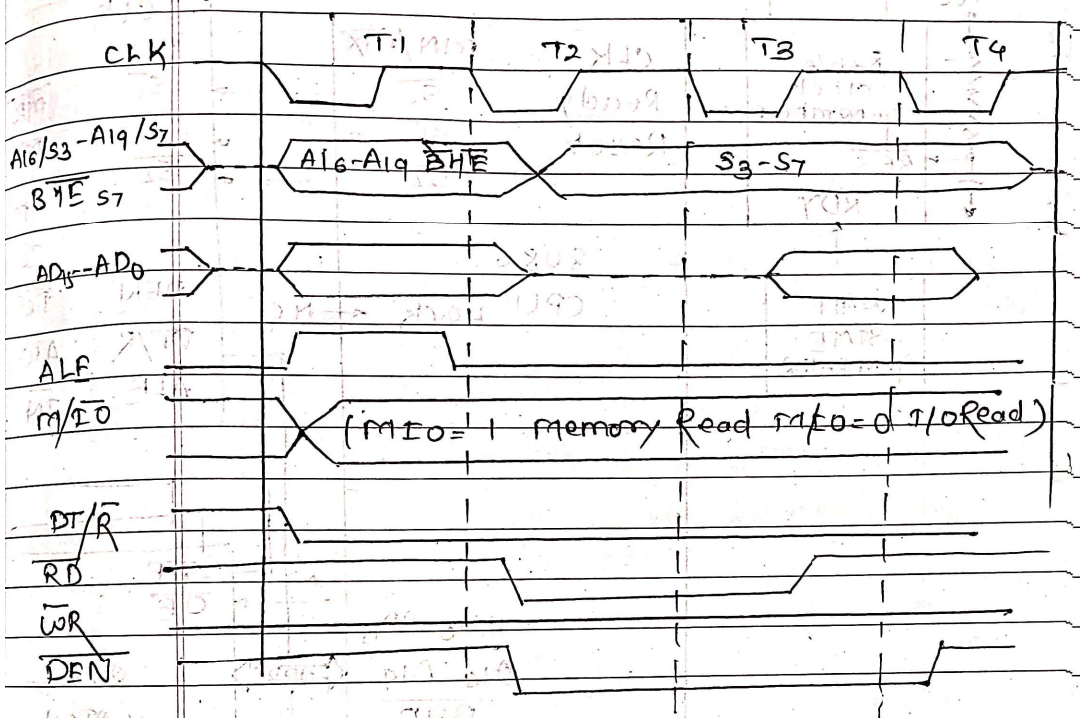
Read/write cycle in 8086 of 4 clock cycle

At 8MHz R/W cycle = 4*125 = 500sec

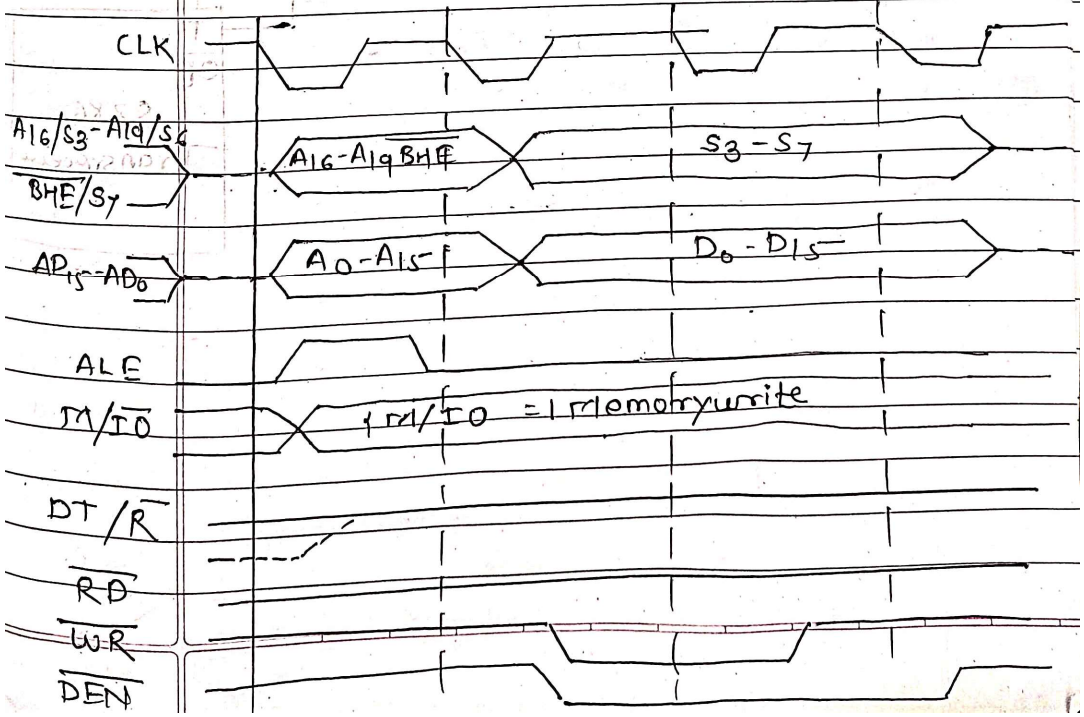
At 10MHz R/W cycle = 4*100 = 400sec

PAGE No.
DATE / /

MINIMUM MODE READ CYCLE



MAXIMUM MODE WRITE CYCLE

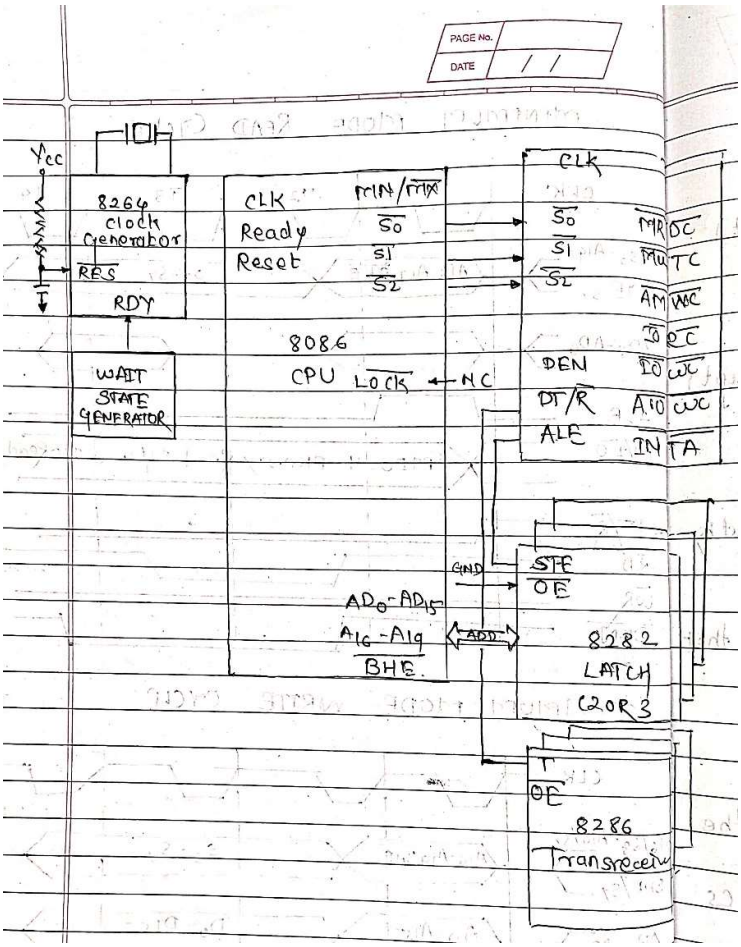


15. Explain maximum mode of operation. [10M]

MAXIMUM MODE

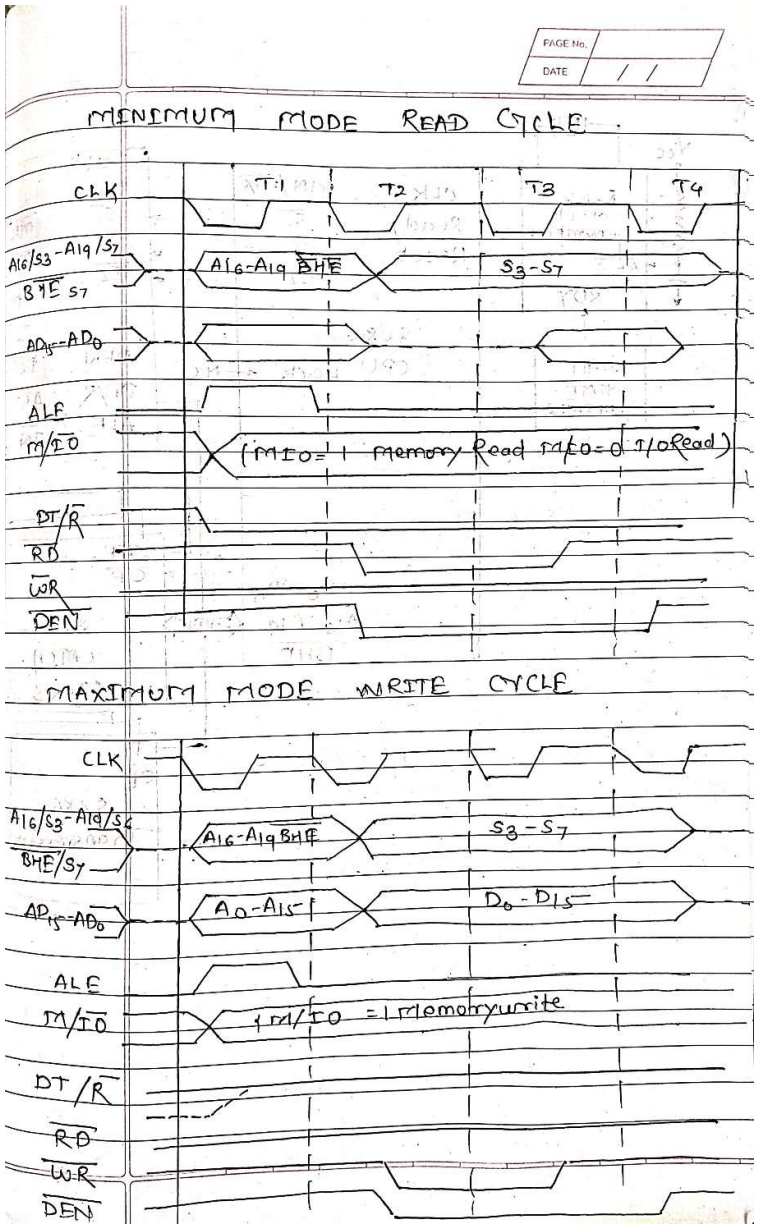
clock provide by 8284 clock generator
The most significant part of maximum mode circuit is the 8288 bus controller
Address from the address bus is latched into 8282 8 bit latch
3 latches are needed as address bus is 20 bit.
ALE is connected to STB of the latch
The databus is driven through 8286 8 bit transreceiver
2 transreceiver is needed as the data bus is 16 bit.
The transreceiver are enabled through the DEN signal.

DEN (of 8288)	DT/R	Action
0	X	Tranreceiver disable
1	0	Receive data
1	1	Transmit data.

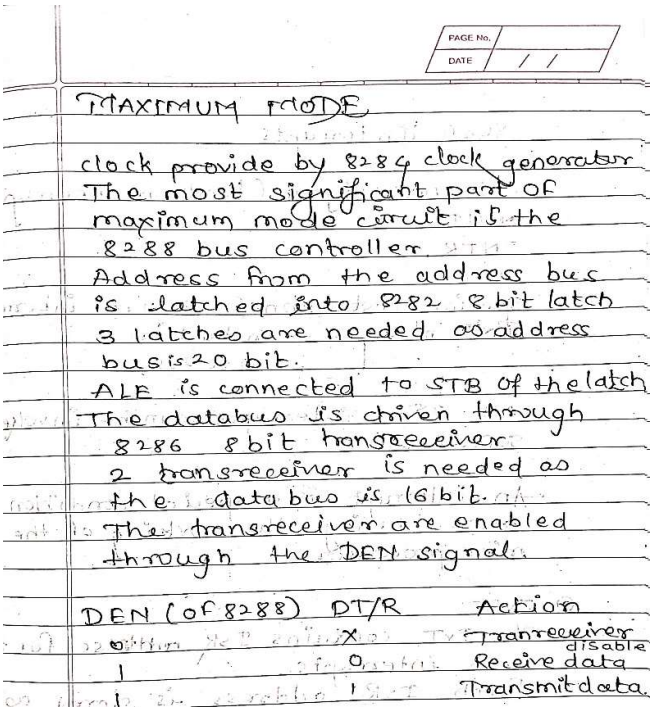


16.Draw and explain timing diagram for RD/WR operation in minimum mode of 8086.[5M]

Minimum mode:
programmed when MN/MX is High
clock, Reset, Ready generated by 8284
8086 generates RD#, WR#, M/IO#,
ALE, HLDA, HOLD.
single processor in the system and
that is 8086, does not support
additional processors.



17.Draw and explain write/read operation timing diagram for maximum mode. [5M]



PAGE No.
DATE / /

MAXIMUM MODE

clock provide by 8284 clock generator
The most significant part of maximum mode circuit is the 8288 bus controller.
Address from the address bus is latched into 8282 8 bit latch
3 latches are needed as address bus is 20 bit.
ALE is connected to STB of the latch
The databus is driven through 8286 8 bit transreceiver
2 transreceiver is needed as the data bus is 16 bit.
The transreceiver are enabled through the DEN signal.

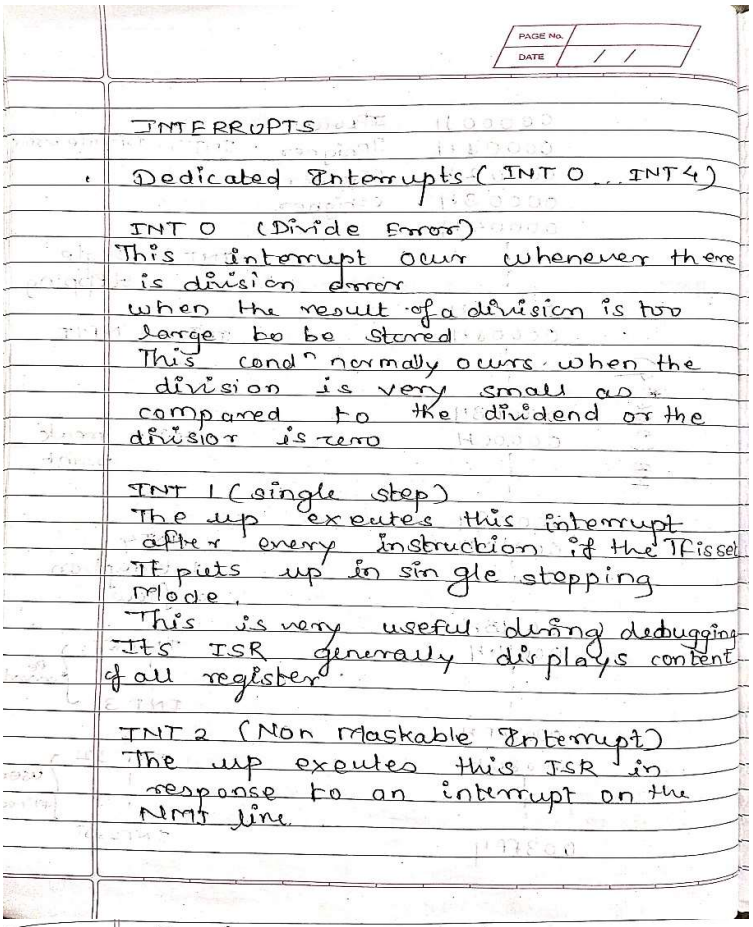
DEN (of 8288)	DT/R	Action
0	X	Tranreceiver disable
1	0	Receive data
1	1	Transmit data

Diagram of write/read operation timing diagram for maximum mode

18.Differentiate between minimum mode and maximum mode of 8086. [5]

18]	Minimum mode	Maximum mode
①	In this mode the processor uses 20 pin configuration.	① In this mode processor uses 26 pin configuration
②	The processor controls the system bus directly	② The processor uses a bus controller to manage the system bus and communicate with other
③	The processor uses a single interrupt pin to handle the interrupts	③ the processor uses a multiplexed interrupt pin and an interrupt controller to manage interrupt.
④	The processor uses an externally generated clock signal	④ the processor can generate it own clock cycle through the bus controller
⑤	processor can access 1MB of memory in this mode	⑤ processor can access upto 16MB memory

19. Explain types of interrupts. [10M]



INTERRUPTS

Dedicated Interrupts (INT 0 .. INT 4)

INT 0 (Divide Error)
This interrupt occur whenever there is division error
when the result of a division is too large to be stored
This cond^n normally occurs when the division is very small as compared to the dividend or the divisor is zero

INT 1 (single step)
The up executes this interrupt after every instruction if the TF is set
It puts up in single stepping Mode.
This is very useful during debugging
Its ISR generally displays content of all register

INT 2 (Non Maskable Interrupt)
The up executes this ISR in response to an interrupt on the NMI line.

Dedicated Interrupts

INT 3 (break point Interrupt)

This interrupt is used to cause Breakpoints in the program
It is caused by writing the instruction INT 03H or simply INT
Its ISR is used to display the contents of all register on the screen

INT 4 (overflow interrupt)
This interrupt occurs if the overflow flag is set AND the up executes the INTO instruction
It is used to detect overflow error in signed arthematic operations.

Reserved Interrupts (INT 5 ... INT 31)
These levels are reserved by INTEL to be used in higher processors like 80386, Pentinum
They are not available to the user

PAGE No.
DATE / /

User defined Interrupts (INT32-INT25

These are user defined software interrupts.
These interrupts are invoked by writing 4 instruction INT n
Its ISR address is obtained by the up from location nx4 in the IVT

HARDWARE INTERRUPTS

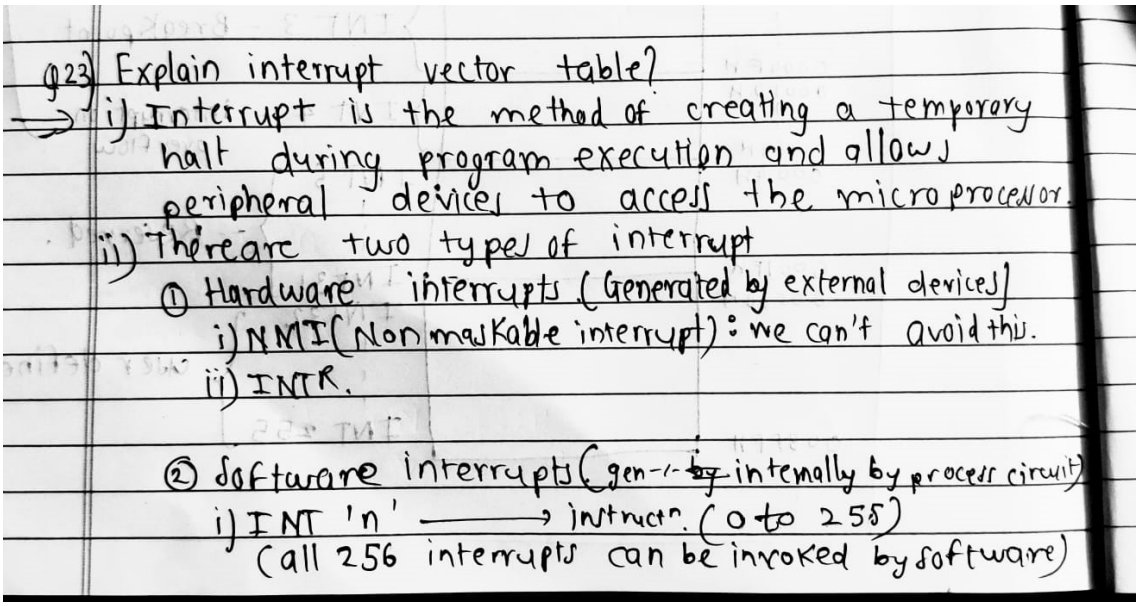
NMI (Non maskable interrupt)
This is a non-maskable edge triggered high priority interrupt.
on receiving an interrupt on NMI line, the up executes INT 2
up obtains the ISR address from location 2x4 = 00008H from the IVT

INTR
This is a maskable level triggered, low priority interrupt.
on receiving an interrupt on INTR line the up exedtes 2 INTR pulses.
INTR is a maskable interrupt
It is masked by marking IF=0 by software through CLI instruction

20. Design the power on reset and manual reset circuit for 8086 processor. [5M]
→REPEATED
21. What is de multiplexing of address and data bus also explain the significance of ALE pin [5M]
→REPEATED
22.Why 8284 is needed in 8086 based system?[5M]
→REPEATED

23.Explain interrupt vector table. [10M]

Ans.



Q23) Explain interrupt vector table?

⇒ i) Interrupt is the method of creating a temporary halt during program execution and allows peripheral devices to access the microprocessor.

ii) There are two types of interrupt

① Hardware interrupts (Generated by external devices)

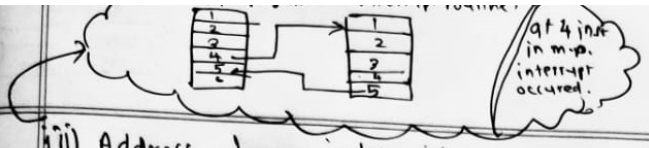
i) NMI (Non maskable interrupt) : we can't avoid this.

ii) INTR.

② Software interrupts (gen-~~by~~ internally by process circuit)

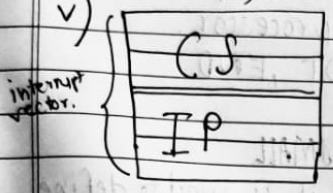
i) INT 'n' ⟶ instruct". (0 to 255)

(all 256 interrupts can be invoked by software)



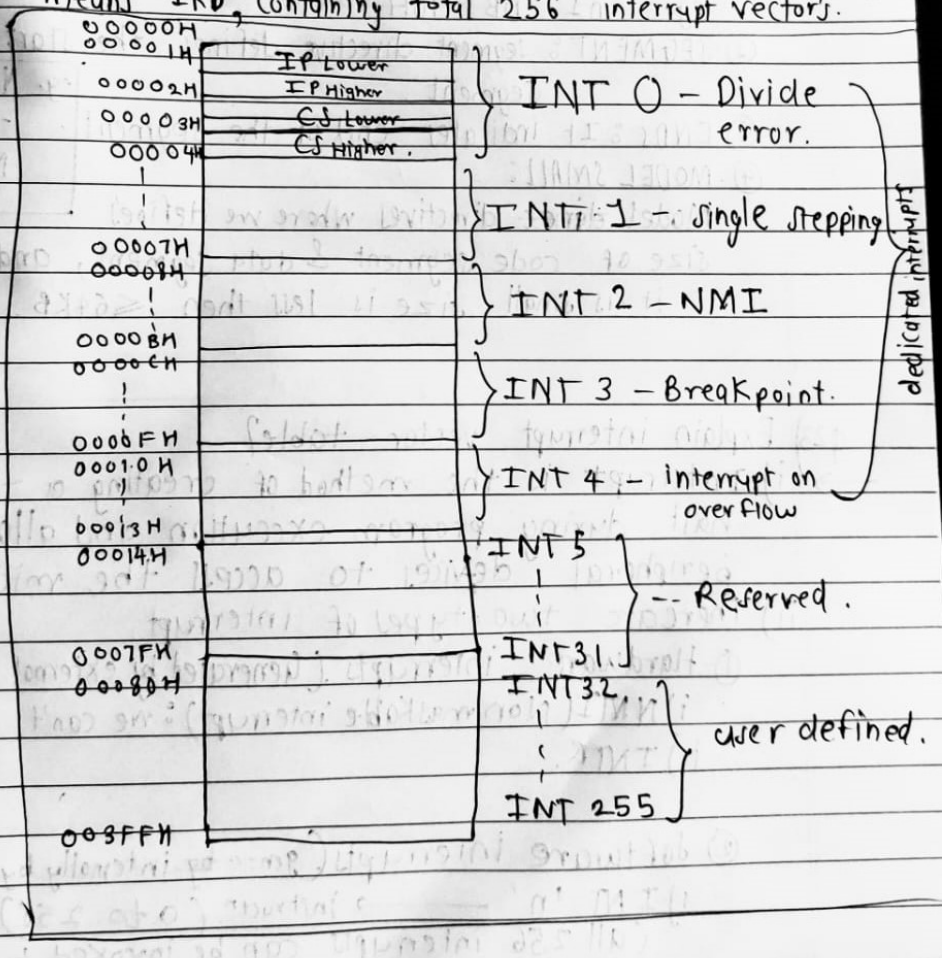
iii) Address where interrupt service routine is saved is known as interrupt vector; where interrupt service routine is set of instruction for execution of interrupt.

iv) Interrupt vector table is collection of interrupt vector's.;

v) 

one interrupt vector table entry is of size. 32 bits (4 bytes).
[16 bit of code segment + 16 bit of Instruction pointer]

vi) in 8086 interrupt vector table is of 1024 bytes. (2^5) (4X256) means 1KB, containing total 256 interrupt vector's.

v) 

vi) Dedicated interrupts: (INT 0 - INT 4)

① INT 0 (Divide error)

→ i) This interrupt occurs whenever there is division error.

ii) eg. when divisor is zero or result is too large

iii) starts at 00000H

② INT 1 (Single step)

→ i) up executes this after every instruction if the TF flag is set. (Puts up in single stepping mode)

ii) useful in debugging.

③ INT 2 (non maskable interrupt)

→ i) NMI pin will send this interrupt.

④ INT 3 (Breakpoint interrupt)

→ i) This interrupt is used to cause breakpoints in the program.

⑤ INT 4 (overflow interrupt)

→ i) occurs when overflow flag is set.

ii) use to detect overflow error.

vii) Reserved interrupt (INT 5 - INT 31)

→ These levels are reserved by INTEL to be used in higher processors like 80386, pentium etc., & not available to user.

viii) Userdefined interrupts (INT 32 - INT 255)

→ This are user defined software interrupts.

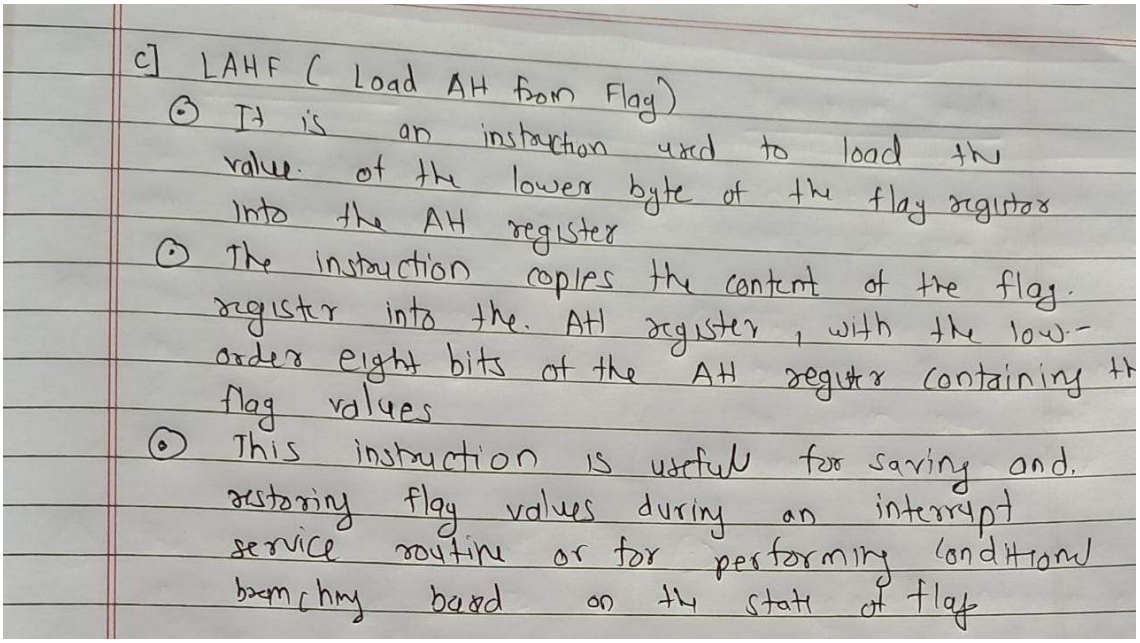
24.Explain following instruction:[10M]

a.DAA

c.XLAT

d.LAHF

e.LAHF



c] LAHF (Load AH from Flag)

- It is an instruction used to load the value of the lower byte of the flag register into the AH register
- The instruction copies the content of the flag register into the AH register, with the low-order eight bits of the AH register containing the flag values
- This instruction is useful for saving and restoring flag values during an interrupt service routine or for performing conditional branching based on the state of flag

DAA (Decimal Adjust Accumulator)

- It is an instruction used to adjust the result of an arithmetic operation performed on two unpacked decimal numbers in AL register. This instruction adjust the result in AL to ensure that it represents a valid decimal no. in the range of 0 to 99.

XLAT (Translate Byte)

- It is an instruction used to translate a byte value in the AL register to another value by using a translation table stored in memory. The instruction is typically used to convert a character code from one character set to another.

SAHF (Store AH into Flags)

- It is an instruction used to load a 32 bit pointer value into a register pair consisting of the DS register and another registers.
- This instruction is typically used to set up a pointer to a data structure or a code segment.

25.Explain string instructions of 8086?

String instructions are a set of instructions in the 8086 microprocessor that are used to manipulate strings of data. A string is a sequence of bytes or words stored in consecutive memory locations, and the string instructions allow for efficient manipulation of these sequences.

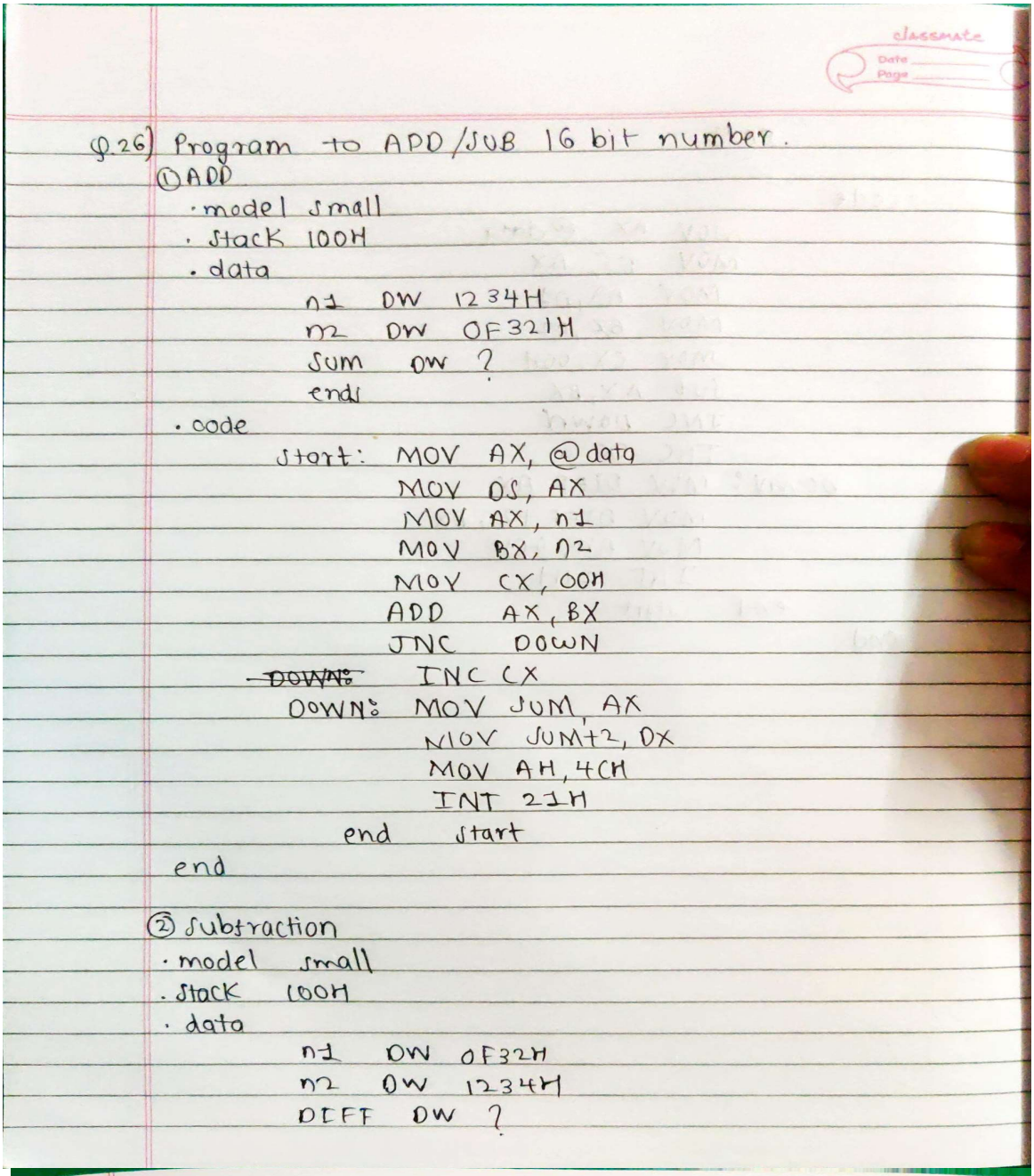
The main string instructions in the 8086 are:

1. MOVSB/MOVSW: These instructions move a byte or a word of data from the source memory location to the destination memory location, and then increment the source and destination pointers. For example, MOVSB copies a byte from DS:SI to ES:DI, while MOVSW copies a word (two bytes) from DS:SI to ES:DI.
2. CMPSB/CMPSW: These instructions compare a byte or a word of data at the source memory location with the byte or word of data at the destination memory location, and then increment the source and destination pointers. For example, CMPSB compares the byte at DS:SI with the byte at ES:DI, while CMPSW compares the word at DS:SI with the word at ES:DI.

3. REP/REPE/REPNE: These are prefixes that are used with string instructions to repeat the instruction a specified number of times or until a condition is met. For example, REP MOVSB repeats the MOVSB instruction until CX becomes zero.

These string instructions are commonly used in string operations such as string copy, string comparison, and string search. They can also be used in other applications where data is stored in consecutive memory locations.

26.Write a program to ADD/SUB two 16 bit number. [5M]



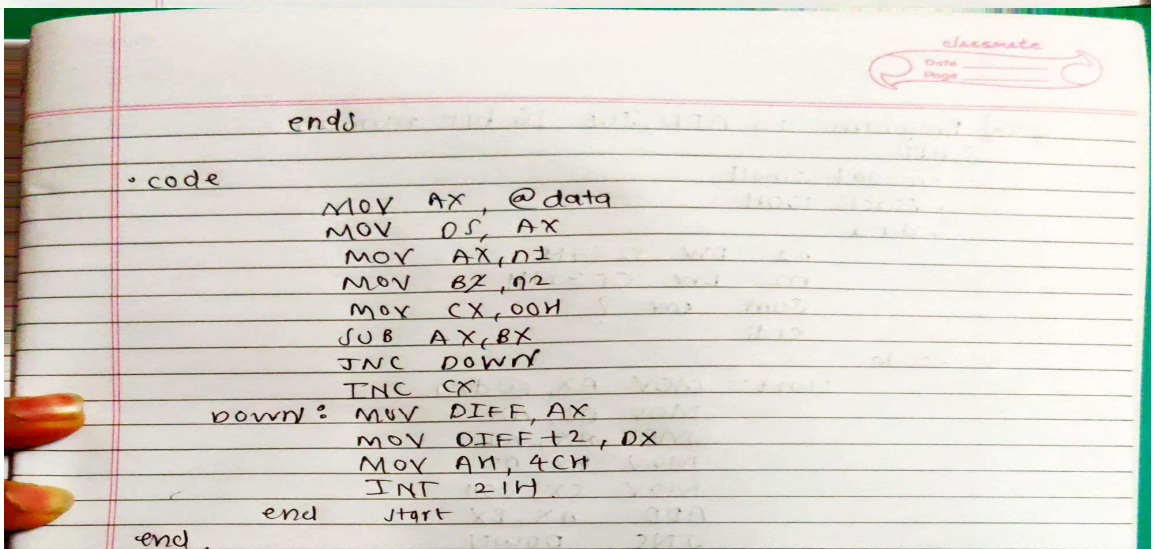
Q.26) Program to ADD/SUB 16 bit number.

①ADD

```
.model small
.stack 100H
.data
        n1  DW  1234H
        n2  DW  0F321H
        sum DW  ?
        ends
.code
    start: MOV AX, @data
           MOV DS, AX
           MOV AX, n1
           MOV BX, n2
           MOV CX, 00H
           ADD AX, BX
           JNC DOWN
    ~~DOWN:~~  INC CX
    DOWN: MOV SUM, AX
          MOV SUM+2, DX
          MOV AH, 4CH
          INT 21H
        end  start
end
```

②Subtraction

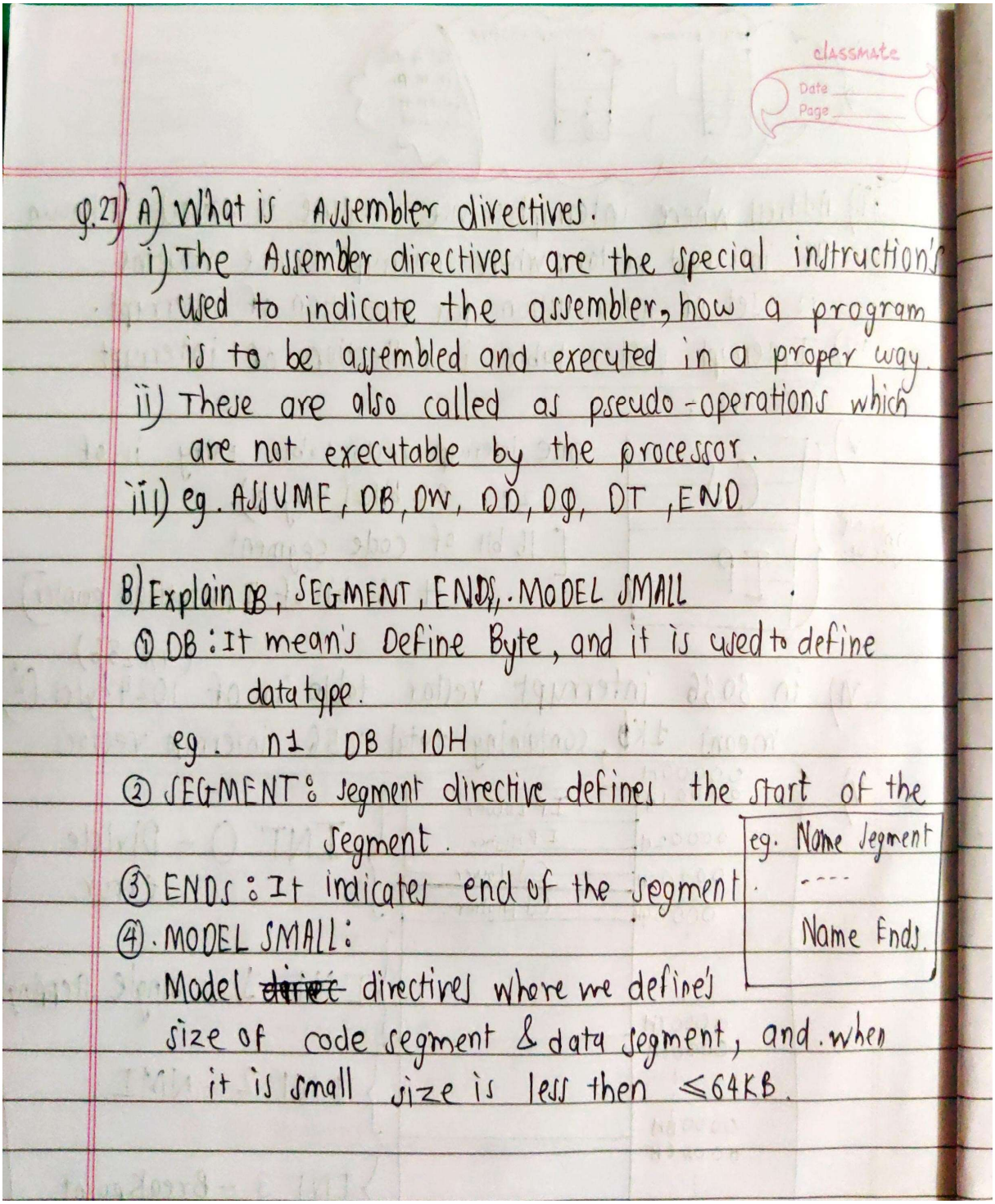
```
.model small
.stack 100H
.data
        n1   DW  0F32H
        n2   DW  1234H
        DIFF DW  ?
```



```
        ends

.code
        MOV AX, @data
        MOV DS, AX
        MOV AX, n1
        MOV BX, n2
        MOV CX, 00H
        SUB AX, BX
        JNC DOWN
        INC CX
    DOWN: MOV DIFF, AX
          MOV DIFF+2, DX
          MOV AH, 4CH
          INT 21H
        end  start
end
```

27.What is Assembler directives? Explain DB, SEGMENT, ENDS, .MODEL SMALL[5M]
Ans.



Q.27) A) What is Assembler directives.

i) The Assembler directives are the special instructions used to indicate the assembler, how a program is to be assembled and executed in a proper way.

ii) These are also called as pseudo-operations which are not executable by the processor.

iii) eg. ASSUME, DB, DW, DD, DQ, DT, END

B) Explain DB, SEGMENT, ENDS, .MODEL SMALL

① DB : It means Define Byte, and it is used to define data type.

eg. n1 DB 10H

② SEGMENT : Segment directive defines the start of the segment.

③ ENDS : It indicates end of the segment.

④ .MODEL SMALL :

Model ~~direc~~ directives where we define's size of code segment & data segment, and when it is small size is less then ≤64KB.

eg. Name Segment
....
Name Ends.

28.Write assembly language program to transfer data stored in data segment to extra segment.[2M]

```
DATA SEGMENT
   DB 'Hello, World!', 0
DATA ENDS

EXTRA SEGMENT
   DB 20 DUP(?)
EXTRA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA, ES:EXTRA

START:
   MOV AX, DATA      ; Set DS to point to the data segment
   MOV DS, AX

   MOV AX, EXTRA     ; Set ES to point to the extra segment
   MOV ES, AX

   MOV CX, 13        ; Set CX to the number of bytes to copy

   MOV SI, 0         ; Set SI to point to the start of the data segment
   MOV DI, 0         ; Set DI to point to the start of the extra segment

   REP MOVSB         ; Copy CX bytes from [DS:SI] to [ES:DI]

   MOV AH, 4Ch       ; Set AH to terminate the program
   INT 21h

CODE ENDS
END START
```

In this program, the data to be transferred is stored in the **DATA** segment, and the destination memory is allocated in the **EXTRA** segment. The **MOVSB** instruction is used in conjunction with the **REP** prefix to copy the data from the data segment to the extra segment.

The program first sets the **DS** and **ES** registers to point to the data segment and extra segment, respectively. It then initializes the **CX** register to the number of bytes to be copied, and sets the **SI** and **DI** registers to point to the start of the data segment and extra segment, respectively.

The **REP MOVSB** instruction then copies the data from the source address **[DS:SI]** to the destination address **[ES:DI]**, incrementing the **SI** and **DI** registers and decrementing the **CX** register with each iteration. Once all of the data has been transferred, the program terminates with an interrupt call to **INT 21h**.

Regenerat