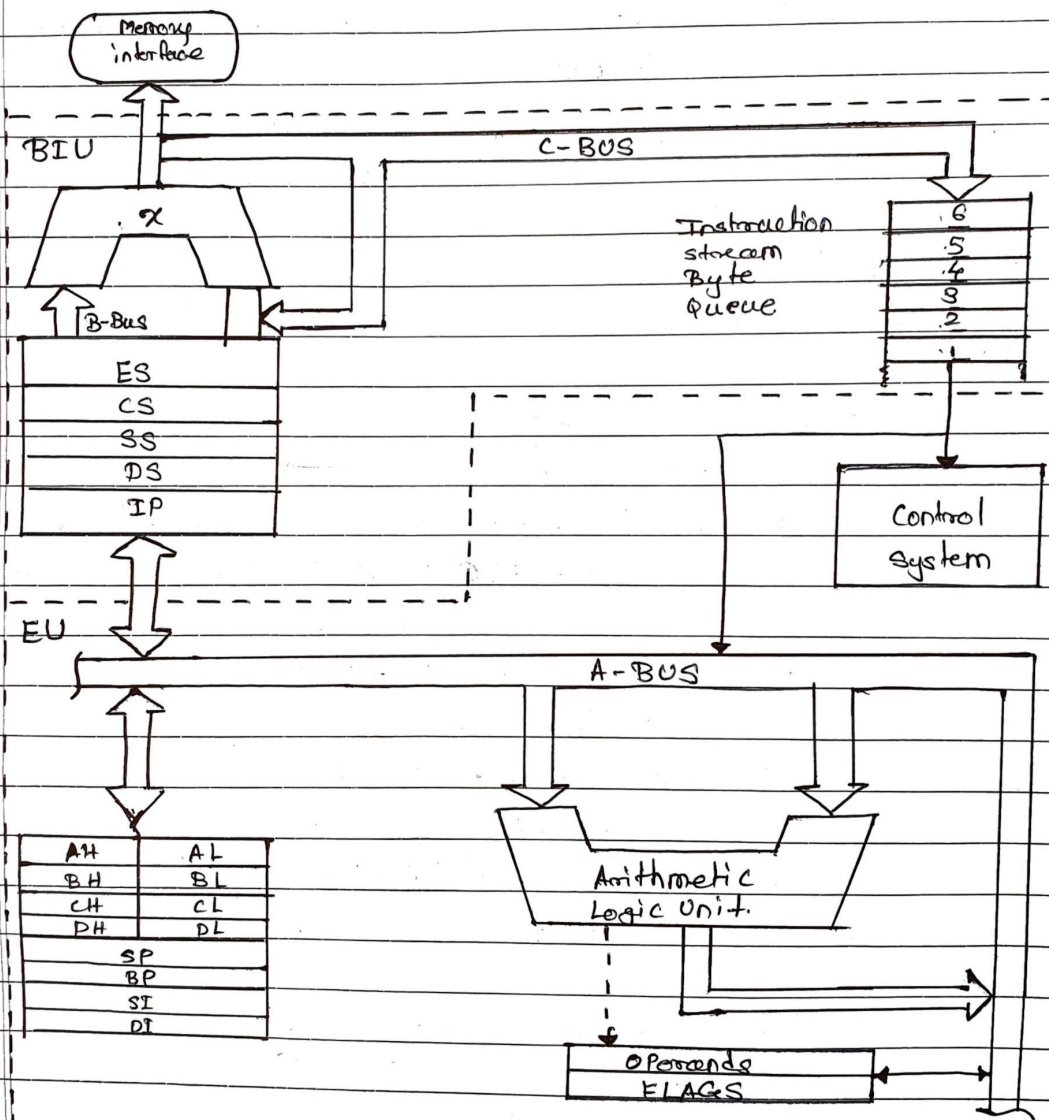
Name: Yash Santosh Sawkare.
Div: CMPNA Roll No: 2110240070
Subject: Microprocessor.

PAGE No.
DATE / /

Assignment 1

1. Draw and explain the architecture of 8086.



The internal architecture of Intel 8086 is divided into 2 units:

a) Bus Interface Unit (BIU)

b) Execution Unit (EU)

a) Bus Interface Unit (BIU)

Fuctions:

i) It generates 20 bit's of physical address for memory access

ii) Fetch instruction from memory.

iii) Transfer data to and from the memory and I/O.

iv) Manages pipelining using 6 byte instruction queue.

Main components:

i) Segment Register:

It consist of 4 (16-bit) segment registers.

a) Code Segment (CS): Holds the base address for code segment.

b) Data Segment (DS): Holds the base address for data segment.

c) stack segment: Holds the base addres for stack segment.

d) Extra Segment (ES): Es holds the base address for extra segment.

ii) Address generation Logic:

It generates the 20 bit physical address using segment & offset addresses.

Using formula given beloco:

$$\text{physical address} = \left(\text{Segment Address}\right) \times 10H + \text{offset address}$$

iii) Instruction Pointer (IP):
It is 16 bit register. It holds offset of the next instruction in the code segment.

iv) 6 Byte Prefetch queue:
i) It is 6 byte fifo RAM, used to implement pipelining (2 stage).
ii) BIU fetches next 6 instruction bytes from the code segment and stored it into queue (Prefetch queue).

b) Execution Unit (EU)
Functions:
i) Decode and execute instructions
ii) Perform arithmetic logic and internal data transfer operation.
iii) Send request signal to BIU to access the external module.
iv) It operates with respect to 'T states' (clock cycle).

Main components:
i) General Purpose Registers
8086 has four 16 bit general purpose registers AX, BX, CX, DX. Which stores intermediate values during execution. Each of these divided into two 8-bit register.

e.g AX → AH, AL BX → BH, BL

ii) Special Purpose Registers.

Special purpose Registers are called offset registers also. Which points to specific memory location under each segment.

- Stack Pointer:
  It holds offset address of top of stack. It used during instructions like PUSH & POP.
- Base Pointer:
  can hold the offset address of any location in the stack segment. Used to access random location of stack.
- Source Index:
  It holds offset address in Data Segment during string operations.
- Destination Index:
  It holds offset address in Extra Segment during string operation.

iii) Arithmetic logic Unit:

Performs 8 and 16-bit arithmetic and logic operations.

iv) Operand:

It is not visible to the user and it holds operand Temporarily.

v) Flag Register:

It has 9 flags that help to change or recognize the state of the microprocessor. Out of 9 flags, 3 are control flags and 6 are status flag.

~~vi) Control system:~~

vi) Control system:
control system in 8086 microprocessor is a component that manages the overall operation of the microprocessor.

2. Explain the prefetch queue.
i) The prefetch unit in 8086 µp is responsible for fetching instructions from memory and storing them in a queue.
ii) It allows µp to perform multiple instructions fetches in parallel, improving the overall performance of µp.
iii) It is a 6 byte FIFO RAM, used to implement pipelining (2 stage)
iv) Fetching the next instruction while executing the current instruction is called pipelining.
v) BIU fetches the next 6 instruction bytes from the code segment and stores it into queue. (Prefetch queue)
vi) Execution unit removes instruction from the queue & execute them.
vii) The queue is refilled when at least two bytes are empty as 8086 has 16 bit of data bus.
viii) Pipelining fails when branch occurs as prefetch instruction are no longer useful. Hence, as soon as 8086 detects a branch operation it clears / Discards entire prefetch queue.
ix) Now the next 6 bytes from the new location are fetched and stored in prefetch queue and pipeling continued.

3. Draw and Explain Flag Register.

→ a) 8086 Flag register:

control flags

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

It has 9 flags that help user ~~can't~~ to change or recognize the state of the microprocessor. ~~Out of th~~ In those 9 flag, 3 are control flags while 6 are status flags as follows:

a) Control Flags:

i) Direction Flag (DF)

1 = Auto Decrement

0 = Auto Decrement

(used in string instructions)

ii) Interrupt Flag (IF)

1 = Enable interrupt

0 = Disable interrupt

(Affects only INTR

iii) Trap Flag (TF)

1 = Perform single stepping

0 = Do not perform single stepping.

b) Status Flags:

i) OverFlow Flag (OF)

1 = overflow occured

0 = NO overflow occured.

ii) Sign Flag (SF)

1 = MSB of result is 1 (:.-ve)

0 = MSB of result is 0 (:.+ve)

(used for "signed" numbers)

iii) Zero Flag (ZF):
1 = Result = 0
0 = Result ≠ 0

iv) Auxilliary Carry Flag (AF):
1 = carry from lower Nibble to Higher Nibble
0 = No such carry
(Used in 8-bit operations)

v) Parity Flag (PF):
1 = Even Parity
0 = odd Parity

vi) Carry Flag (CF)
1 = Carry out of MSB
0 = No such carry

4. Write a formula to calculate physical address. And explain why it's required to calculate it?

→ Formula:

$$\text{physical Address} = (\text{Segment Address}) \times 10H + \text{offset address}$$

As we know the address bus of 8086 microprocessor is of 20 bit. But the registers in segment register are of 16 bit size. Each register stores the base address (starting address) of corresponding segment. Because segment register cannot store 20 bit's, they only store upper 16 bit's. so to access a specific memory location from any segment we need 20 bit physical address. Hence it is required to calculate physical address in 8086 microprocessor.

5. Write a note on addressing modes of 8086.

→ Addressing modes is the manner in which an operand is given in an instruction. The different ways of specifying data to be operated by an instruction is known as addressing modes. This specifies that the given data is an immediate data or an address of data or data through register or register.

Data addressing modes:

1. Immediate Addressing mode:
   i) Operand is specified in the instruction itself.
   ii) Source Operand is 8 bit or 16 bit data.
   iii) Destination operand can never be immediate data.

   E.g. MOV CX, 2000H
   MOV CL, 0AH ; Moves 0AH immidiately into CL register
   MOV CL, 45H

2. Register Addressing mode:
   i) Here the operand is given using a register
   ii) The data that we need to operate on is present in some register.
   iii) We write name of register in the instruction up will access data from register & perform operation.

   E.g. MOV CL, DL ; Moves data of DL reg to CL reg.
   MOV AX, BX ; Moves data of BX reg to AX reg.

3. Direct Addressing mode:
   i) Here the address of operand is directly written in the instruction
   ii) The data that we need to words on is present in some memory location.
   iii) This address is not physical address it is a offset address which we write in instruction

E.g. MOV CL, [2000H] ; moves the data from location 2000H in the data segment into CL register.

; The physical address : DS * 10H + 2000 , let DS = 5000H

P.A : 50000 + 2000 = 52000H

;∴ CL ← [52000H]

4. Indirect Addressing mode:

This addressing mode is divided into 4 types

a) Register Indirect addressing mode.

i) Here the address of operand is given using register.

ii) The data that we need to operate on is present in some memory location and address of that memory location is present in the register.

iii) We provide name of register in instruction.

e.g: MOV CL, [BX] ; CL gets 8-bit data from memory location whose 16-bit address is given by BX, in data segment.

b) Register Relative addressing mode:

i) Here the address of the operand is given as sum of register and a displacement.

e.g: MOV CL, [BX+4] ;

moves a byte from location pointed by BX+4 in data segment to CL.

P.A : DS * 10H + BX + 4H

c) Base indexed Addressing mode:

i) Here operand address given as sum of Base register plus an Index Register.

PAGE No.
DATE / /

e.g; MOV CL, [BX+SI]
Moves a byte from the address pointed by BX+SI in data segment to CL.
PA: DS * 10H + BX + SI

We only can use base registers BX and BP, and the index registers SI and DI.
BX, SI & DI will works for data segment & BP will work for stack segment
∴ Possible combinations: BX+SI, BX+DI, BP+SI, BP+DI

D) Base Relative plus indexed addressing mode:
Here, operand address is given as sum of Base register plus an Index register plus a displacement.
e.g: MOV CL, [BX+DI+20]
Moves a byte from the address pointed by BX+SI+20H in data segment to CL.
P.A = DS * 10H + BX + SI + 20H

5. Implied addressing mode:
In this addressing mode the operands are implied and are hence not specified in the instruction.
e.g: STC ; sets the carry Flag
CLD ; clears the direction Flag.

6. The instruction 'MOV BL, [SI]' comes under which type of addressing mode.

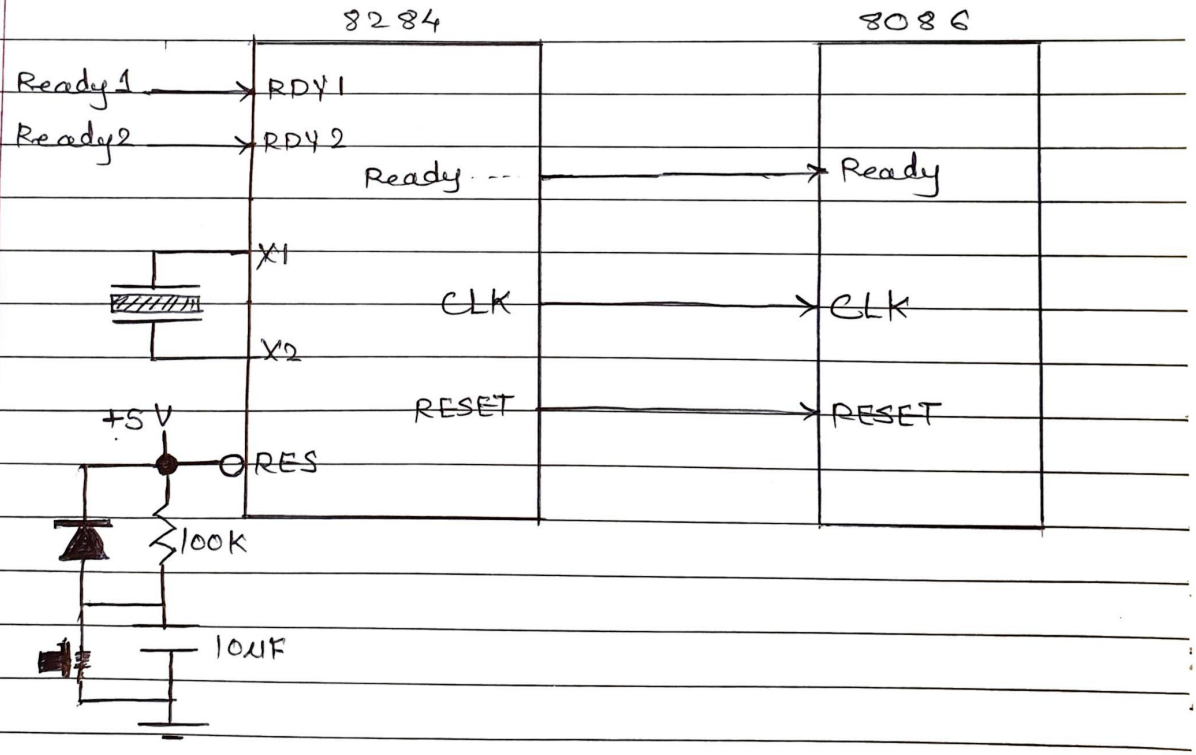
→ MOV BL, [SI]
Here [SI] indicates the ~~data we need to work on is present in~~ address of data that we need to work on is present SI register.

PAGE No.
DATE / /

so, here BL register gets 8-bit data from a memory location whose 16-bit offset address is given by SI, ~~in stack se~~
Hence it is Register indirect addressing mode.

7. Design Power on reset and manual reset circuit for 8086 processor.

→



PAGE No.
DATE / /

8. What is demultiplexing of address and data bus also explain significance of ALE pin.

→ 8086 has a multiplexed address / data bus.

i) First $A_0 - A_{15}$ address busses are multiplexed with $D_0 - D_{15}$ data busses giving us $AD_0 - AD_{15}$ multiplexed Data & address bus.

ii) Whereas $A_{16} - A_{19}$ address busses are multiplexed with $S_3 - S_6$ status signals giving us $A_{16}/S_3 - A_{19}/S_6$ multiplexed address & status bus.

iii) Interestingly, address and data are never given simultaneously in any operation. Now this is where demultiplexing comes into picture.

iv) In $AD_0 - AD_{15}$, During 1st T, they carry lower order 16 bit address and In the remaing ~~time~~ clock cycles they carry data.

v) In $A_{16}/S_3 - A_{19}/S_6$, During 1st T, they carry higher order 4-bit address and In the remaing time they carry status signals.

Significance of ALE Pin:

ALE has very important role in demultiplexing Address and data busses.

i) When ALE = 1 the respective busses carry $A_0 - A_{15}$, $A_{16} - A_1$ and $\overline{BHE}$

ii) When ALE = 0 the respective busses carry $D_0 - D_{15}$, $S_8 - S_7$.

iii) During 1st T state of any machine cycle ALE is ~~High~~ set of High. At that time the busses carry address and $\overline{BHE}$. ~~Then~~

iv) Thereafter BHE remains low for rest of machine cycle, At that time busses carry data and status.

Multiplexed signals	$AD_0 - AD_{15}$	$A_{16}/S_3 - A_{19}/S_6$	$\overline{BHE}/S_7$
when ALE=1	AD - A15	$A_{16} - A_{19}$	$\overline{BHE}$
when ALE=0	$D_0 - D_{15}$	$S_3 - S_6$	S_7

8. Why 8084 is needed in 8086 based system?

→ i) 8084 is a clock generator IC. ~~It provides~~

ii) It provides the $\overline{CLOCK}$ (CLK) signal, a train of pulses at a constant frequency to entire circuit.

iii) It synchronizes the READY signal which indicates that interface is ready for data.

iv) It also synchronizes the RESET signal which is used to initialize the system.

v) 8084 output's the clock frequency = 1/3rd of the Input clock frequency to produce a 33% duty cycle required by the Microprocessor.

vi) Also, RESET signal is provided by 8284 clock generator. Generation of reset signal is done by using power on reset circuit.

vii) The purpose of this circuit is to activate the reset signal as soon as we supply power to 8086.

viii) This is done so that CS & IP acquire the above values and the system can initialize the monitor program (BIOS) as soon as 8086 is powered on. This is also called COLD starting the system.

10. List Features of 8086 microprocessor.

i) Enhanced version of 8085 designed by Intel in 1976.

ii) It is 16-bit microprocessor.

iii) It has 20-bit address bus so can access upto 2^{20} memory location.

iv) 16 bit data lines.

v) 64k I/O ports.

vi) 14, 16 bit registers

vii) It has powerfull instruction set (Multiplication & Divisions)

viii) It supports two mode of operations:

- Minimum Mode: suitable for single processor
- Maximum Mode: suitable for multiple processor.

ix) It has instruction queue capable of storing 6 instructions bytes from memory resulting in Faster processing.

x) It was the 1st 16 bit processor having 16 bit ALU, 16 Bit Registers, internal data bus resulting in Faster processing.

xi) Available in 3 versions:

a) 5 MHz b) 8 MHz c) 10 MHz

xii) Two stages of pipelining (Fetch + Execution)

xiii) single phase ~~of pipelining with~~ 33% clock with 33% duty cycle

xiv) 256 vectored interrupts.

11. What is memory addressing capacity of 8086? and Why?

→ The memory addressing capacity of 8086 µp is 1 MB. This is because 8086 is a

PAGE No.
DATE / /

16-bit microprocessor having 20 bit address bus and 16-bit data bus.

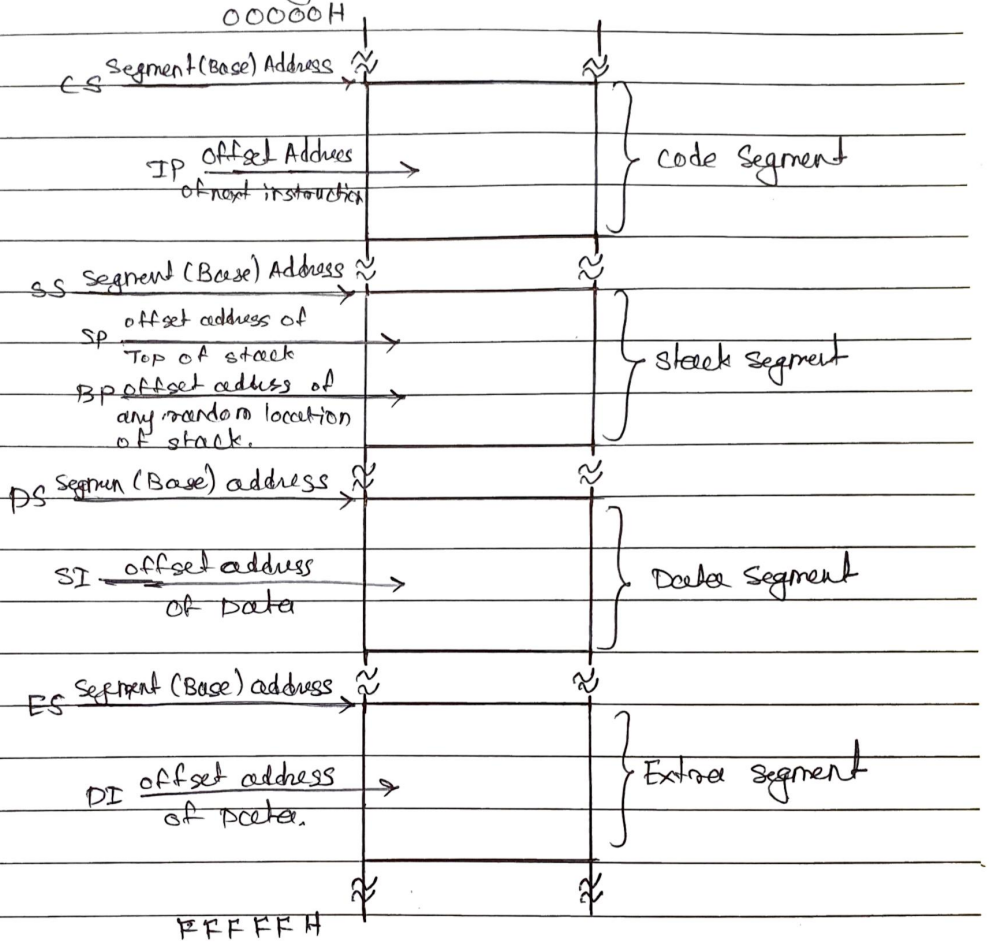
We know an N-bit address bus can totally access 2^N Location. So one memory location contains one byte of data.

This brings us to the conclusion that "A processor with an N-bit address bus can access a memory of 2^N Bytes.

∴ 8086 can access 2^{20} bytes which is equal to 1MB.

12) Explain the memory segmentation in 8086 and list its advantages.

→



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

a) Code Segment:

i) This segment used to hold the program to be executed.

ii) Instructions are fetched from the code segment

iii) CS register holds the 16-bit base address for this segment.

iv) IP register (Instruction pointer) holds 16-bit offset address

PAGE No.
DATE / /

b) Data segment
i) This segment is used to hold the general data.
ii) This segment also holds the source operands during string operations.
iii) DS register holds the 16 bit base address for this segment
iv) BX register is used to hold the 16-bit offset for this segment.
v) SI register (source index) holds 16-bit offset address during string operations.

c) stack Segment
i) This segment ~~operat~~ holds the stack memory which operates in LIFO manner.
ii) SS holds the base address
iii) SP (stack Pointer) holds the 16-bit offset address of the top of the stack.
iv) BP (Base Pointer) holds the 16-bit offset address during Random Access.

D) Extra segment:
i) This segment is used to hold the general data.
ii) Additionally, this segment is used as the destination during string operation.
iii) ES holds the base address.
iv) DI holds the offset address during string operation.

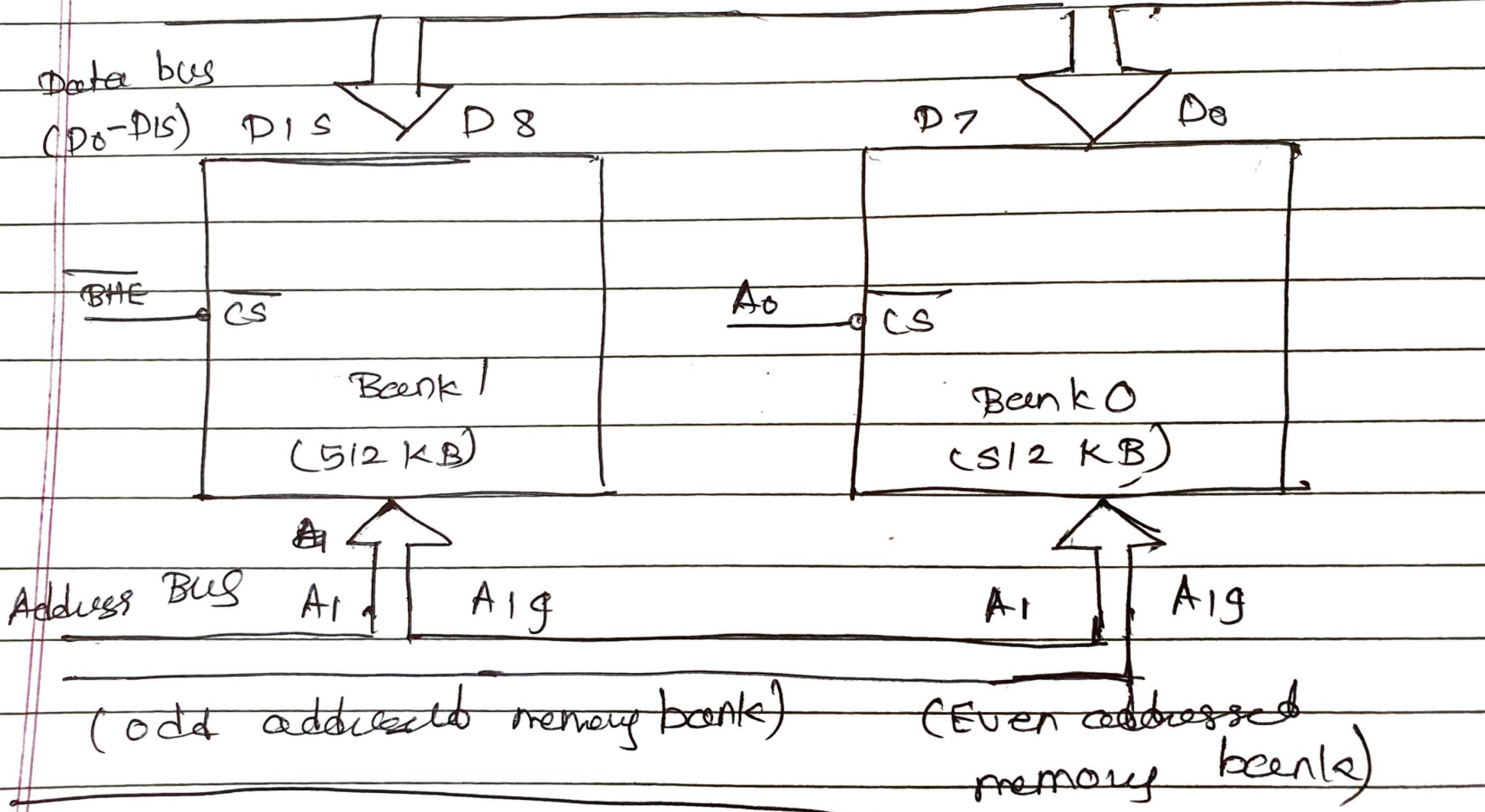
Advantages:
1. It permits the programmer to access 1MB using only 16-bit address
2. It divides the memory logically to store Instructions, data and stack seperately, allowing more flexibility.
3. Due to segmentation logical address range is from 00000H to FFFFFH, the code can be loaded at any location in memory.

13. Explain the Memory banking in 8086 system and describe its advantages.

→

~~Advantages~~:

The 8086 processor provides a 16 bit data bus, so it is capable of transferring 16 bit's in one cycle but each memory location is only of a byte (8 bit's), therefore we need two cycles to access 16 bits from two diffent memory location. For this purpose, The memory address space is equally divided into two (banks). One of the banks contains even address called even bank and other contains odd address called odd bank. Even bank always stores lower byte so even bank is also called lower bank (LB) and odd Bank is also called a Higher Bank (HB)


Data bus
(D0-D15)
D15
D8
D7
D0
BHE
CS
A0
CS
Bank 1
(512 KB)
Bank 0
(512 KB)
A1
Address Bus
A1
A19
A1
A19
(odd addressed memory bank)
(Even addressed memory bank)

$\overline{BHE}$	A_0	Operation
0	0	Read or write 16 bit from both banks
0	1	read or write 8 bit's from Higher bank
1	0	Read or write 8 bits from lower bank.
1	1	No operation (Processor is idele)

Examples: ~~8bit operation on even bank~~

1) MOV BL, [2000H]
   # The operation is of 8 bit.
   # BL register is going to get a value stored at offset address 2000 H.
   # Looking at the address (As it is even) up is going to select lower bank.
   # only one bank will be selected.
   ∴ $A_0 = 0$ and $\overline{BHE} = 1$

2) MOV BH, [2001H]
   # —//—
   # BH —//— 2001 H
   # —//— (As it is odd) —//— Higher bank.
   # —//—
   ∴ $A_0 = 1$ and $\overline{BHE} = 0$.

3) MOV BX, [2000H]
   # The operation is of 16 bit.
   # BX register is going to get a value stored at offset addus 2000H and 2001H.
   # As Data is 16 bit both banks will be selected.
   # ∴ $A_0 = 0$ and $\overline{BHE} = 0$

4) MOV BX, [2001H] (misaligned operation)

It is 16 bit operation.

BX reg is going to get offset address from 2001H and 2002H

when 16-bit operation ~~is going~~ begins with odd address it is said to be misaligned operation. Though misaligned data is a valid data such data transfer cannot happen in one clock cycle. ∴ up will break this operation into two cycle.

1st cycle: 2001H location data is transfered to ~~B~~ Higher Bank.

2nd cycle: 2002H location data is transfered to lower Bank.

PAGE No.
DATE / /

Advantages:

i) Allows 8086 up to access more memory than it would otherwise ~~not~~ be able to.

ii) 8086 can better manage memory ~~resources~~ resources by dividing them into small banks.

iii) Cost-effective: Instead of having to purchase more memory chips, processor can access more memory with same amount of hardware.

iv) Improves the performance of 8086 microprocessor.

15. Explain ~~maximum~~ minimum mode of operation

i) 8086 works in minimum mode, when MN/$\overline{MX}$ = 1.

ii) In minimum mode, 8086 is the only processor in the system.

iii) clk, READY and RESET signals are provided by 8284 clock generator.

iv) Address from the address bus is latched into 8282 8-bit latch. Three such latches are needed, as address bus is 20-bit. The ALE of 8086 is connected to STB of the Latch.

v) The data bus is driven through 8286, 8-bit transreceivers. Two such transreceivers are needed, as the data bus is 16-bit.
The transreceivers are enabled through the $\overline{DEN}$ signal.
Direction of data is controlled by the DT/$\overline{R}$ signal.
The $\overline{DEN}$ is connected to $\overline{OE}$ and DT/$\overline{R}$ is connected to T.

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

vi) control signals for all operations are generated by decoding M/$\overline{IO}$, $\overline{RD}$ and $\overline{WR}$.

M/$\overline{IO}$	$\overline{RD}$	$\overline{WR}$	Operation.
1	0	1	Memory Read
1	1	0	Memory Write
0	0	1	I/O Read
0	1	0	I/O write.

PAGE No.
DATE / /

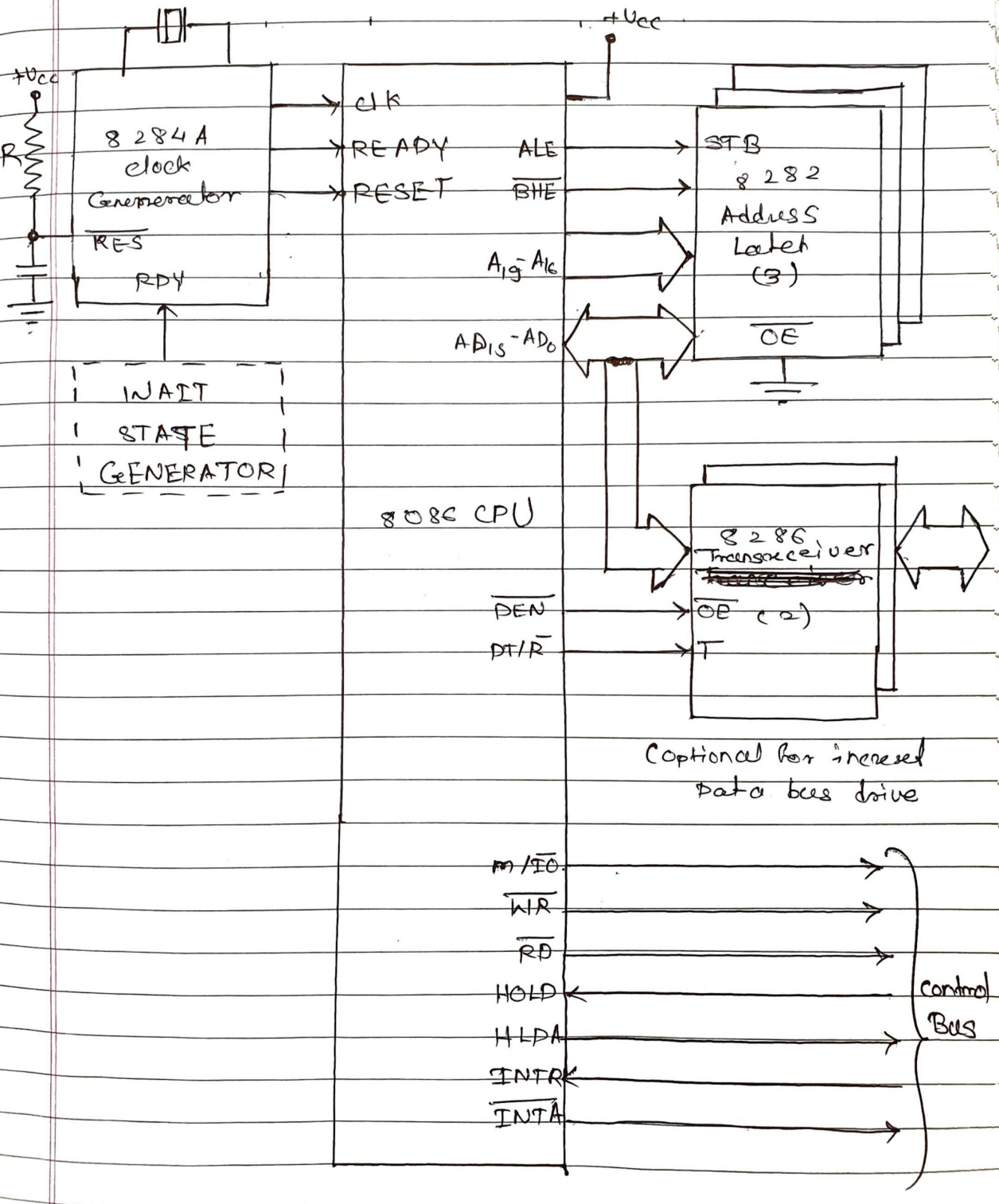
vii) $M/\overline{IO}$, $\overline{RD}$, $\overline{WR}$ are decoded by 3:8 decoder like IC 74138.

viii) Bus Request (DMA) is done using the HOLD, and HLDA signals.

ix) $\overline{INTA}$ is given by 8086, in responce to an interrupt of INTR line.

PAGE No.
DATE / /

* Minimum Mode Configuration



15. Explain maximum mode of operation.

i) 8086 works in maximum mode, when MN/$\overline{MX}$ = 0

ii) In maximum mode, we can connect more processors to 8086 (8087, 8088)

iii) clock is provided by 8284 clock generator.

iv) The most significant part of the maximum circuit is the 8288 Bus-controller.

v) Address from the address bus is latched into 8282-8-bit latch.

Three such latches are needed, as address bus is 20 bit.

The ALE is connected to STB of the Latch.

The ALE for this latch is given by 8288 Bus controller.

vi) The data is driven through 8288 8-bit transreceiver.

Two such transceivers are needed, as the data bus is 16-bit. These transreceivers are enabled through the DEN signal.

The direction of data is controlled by DT/$\overline{R}$ signal.

Both DEN & DT/$\overline{R}$ are given by 8288 Bus Controller.

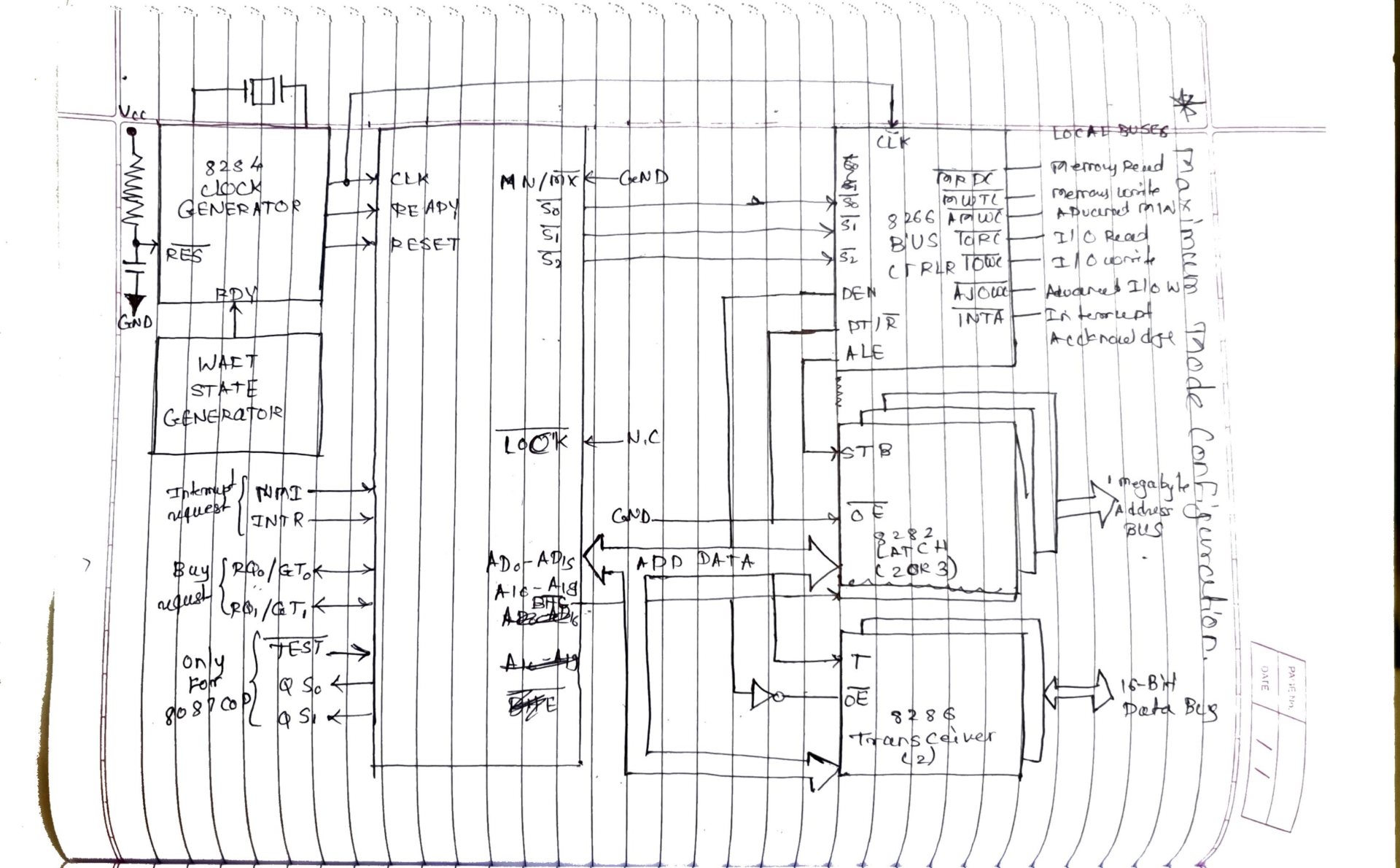
DEN	DT/$\overline{R}$	Action
0	X	Transreceiver is disabled
1	0	Receive data
1	1	Transmit data.

viii) control signals for all operations are generated by decoding $\overline{S_2}$, $\overline{S_1}$ and $\overline{S_0}$ signals.

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	Processor state	8288 Active output.
0	0	0	Int Ack	$\overline{INTA}$
0	0	1	Read I/O port	$\overline{IORC}$
0	1	0	Write I/O port	$\overline{IOWC}$ & $\overline{AIOWC}$
0	1	1	Halt	None
1	0	0	Instruction Fetch	$\overline{MRDC}$
1	0	1	Memory Read	$\overline{MRDC}$
1	1	0	Memory Write	$\overline{MWTC}$ & $\overline{AMWTC}$
1	1	1	Inactive	None

17. ~~Draw and explain write / read operation in minimum mode of~~

17. ~~Draw and ex~~


Maximum Mode Configuration.
Vcc
GND
8284
CLOCK
GENERATOR
RES
RDY
WAIT
STATE
GENERATOR
CLK
READY
RESET
MN/MX
GND
S0
S1
S2
LOCK
N.C
Interrupt request
NMI
INTR
Bus request
RQ0/GT0
RQ1/GT1
only for 8087 CPU
TEST
QS0
QS1
AD0-AD15
A16-A19
ADDRESS
DATA
CLK
8288
BUS
CTRLR
DEN
DT/R
ALE
MRDC
MWTC
AMWC
IORC
IOWC
AIOWC
INTA
LOCAL BUSES
Memory Read
memory write
Advanced MW
I/O Read
I/O write
Advanced I/O W
Interrupt
Acknowledge
STB
OE
8282
LATCH
(2 OR 3)
ADD DATA
1 megabyte
Address
BUS
T
OE
8286
Transceiver
(2)
16-BIT
Data Bus
PAGE NO.
DATE

PAGE No.
DATE / /

18) Draw and explain timing diagram for RD/INR operation in minimum mode of 8086.

→ • Timing Diagram for "read" machine cycle

T_1 = ALE goes high as both multiplexed buses carry address.
This allows the latch to capture the address till ALE goes low.

T_2 = µp asks for data by making $\overline{RD}$ signal low.
As address is no longer present in the multiplexed bus, the transceiver is enabled by $\overline{DEN}$ = 0.
The data of course does reach the µp immediately.
Data is coming from memory hence it will take time to travel and reach the µp. This time is called "Propagation delay".

T_3 = Memory starts sending data on the data bus.
Towards the end of T_3, µp checks the Ready signal.
If Ready = 1 it means device is redy and µp completes the machine cycle in the next T-state (T_4)
if Ready = 0 it means device is not ready
µp will insert a wait state between T_3 and T_4.

T_4 = µp captures the data sent by memory device
The multiplexed buses become tri-stated and will open up in next machine cycle.
All signals like $\overline{RD}$ $\overline{DEN}$ are deactivated.

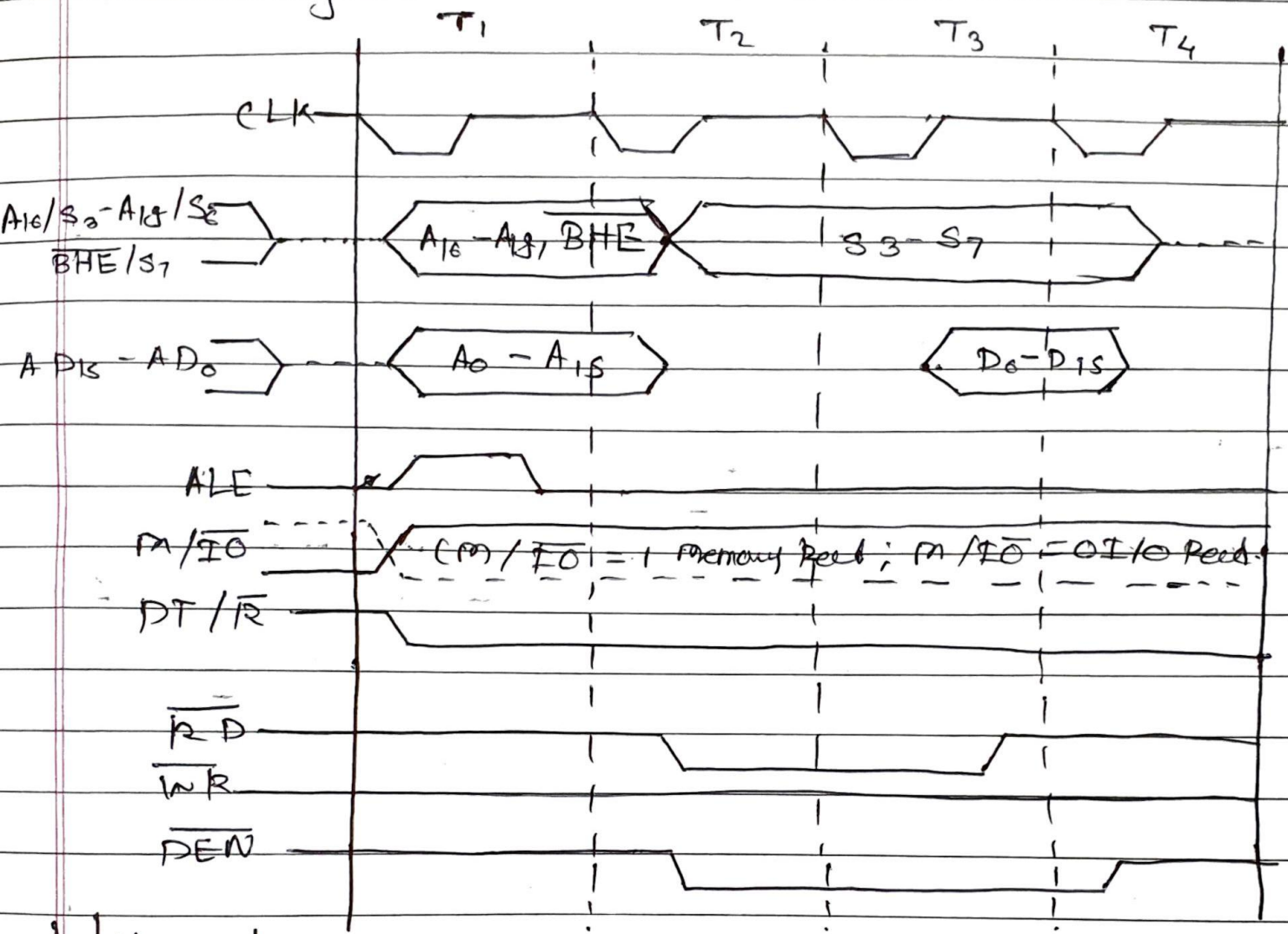
$\overline{RD} \rightarrow 1$

$\overline{RD} \rightarrow 0$

PAGE No.

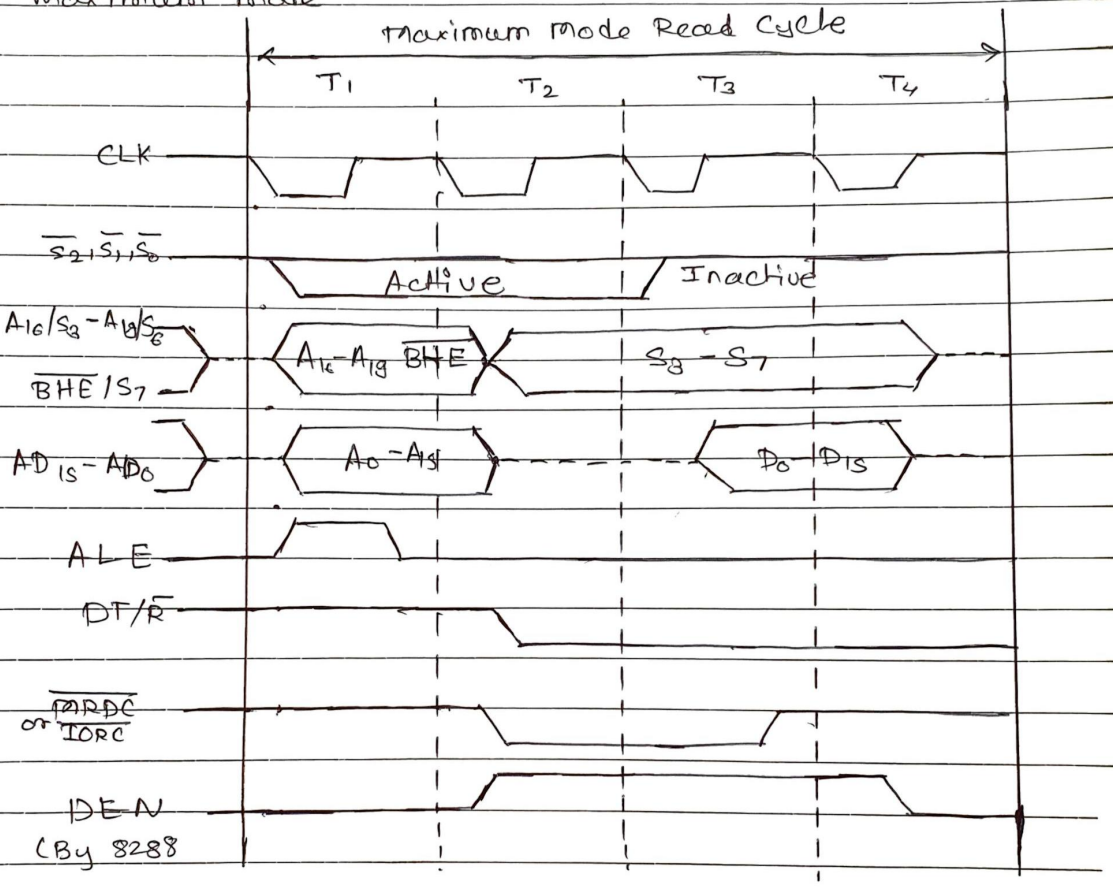
DATE / /

Read Cycle



17. Draw and explain write/read operation timing diagram for maximum mode.

→ ∘Timing Diagram for "read" machine cycle in maximum mode



T_1 = ALE goes high as both multiplexed buses carry address.
This allows the latch to capture the address till ALE goes low.
Also the signals $\overline{S_2}$, $\overline{S_1}$, $\overline{S_0}$ are ~~set to 1~~ are in ~~re~~ active state

T_2 which allows multiplexed bus $A_{16}/S_3 - A_{19}/S_6$

T_2 = ~~up asks for data by making~~
and $\overline{BHE}/S_7$ to carry Address. ~~and~~
Also $AD_{15} - AD_0$ Bus will carry address.
Untill ALE goes low.

T_2 = ~~8288~~ µp asks for data by making $\overline{MRDC}$ ~~or $\overline{MWTC}$ signal~~ or $\overline{IORC}$ signal low.
As address is no longer present in the multiplexed bus, the transreceiver is enabled by DEN=1
A_4/S_3 - A_{19}/S_6 and $\overline{BHE}/S_7$ multiplexed Bus will now convey the signals S_3-S_7 ~~until~~ ~~mid of~~ T_4.
Data is coming from memory hence it will take time to travel and reach the µp ("propogation delay")

T_3 = Memory starts sending data on the data bus

PAGE No.
DATE / /

18) Differentiate between minimum mode and maximum mode.

Minimum mode	Maximum mode
i) It is uniprocessor mode. 8086 is only processor in circuit.	i) It is a multiprocessor mode. Along with 8086, there can be other processors like 8087 & 8089 in circuit.
ii) MN/$\overline{MX}$ is connected to Vcc.	ii) Here MN/$\overline{MX}$ is connected to Ground.
iii) ALE for Latch is given by 8086 itself.	iii) As there are multiple processors, ALE for the latch is given by 8288 bus controller
iv) $\overline{DEN}$ and DT/$\overline{R}$ for the transreceivers are given by 8086 itself.	iv) As there are multiple processors, DEN and DT/$\overline{R}$ for the transreceivers is given by 8288 bus controller.
v) Direct control signals like M/$\overline{IO}$, $\overline{RD}$ and $\overline{WR}$ are produced by 8086 itself.	v) Instead of control signals, all processors produce status signals $\overline{S_2}$, $\overline{S_1}$ and $\overline{S_0}$.
vi) $\overline{INTA}$ for interrupt acknowledgement is produced by 8086.	vi) $\overline{INTA}$ for interrupt acknowledgement is produced by 8288 bus controller.
vii) control signals M/$\overline{IO}$, $\overline{RD}$ & $\overline{WR}$ are decoded by 3:8 decoder (IC 74138)	vii) status signals $\overline{S_2}$, $\overline{S_1}$ & $\overline{S_0}$ requires special decoding are decoded by 8288 bus controller.
viii) Does not support multiprocessing.	viii) supports multiprocessing.

PAGE No.
DATE / /

18] Explain Types of interrupts:
There are Interrupts are categorise into 4 types:

a) Dedicated Interrupts
i) INT 00 (Divide Error)
- This interrupts occurs when there is a division error i.e when the result of a division is too large to be stored.
- This condition normally occurs when the divisor is very small as compared to the dividend or the divisor is zero.
- It's ISR address is stored at location 0x4 = 00000H in IUT

ii) INT 1 (single step)
- up executes this ~~instructi~~ interrupt after every instruction if the TF is set.
- It puts up in single stepping mode i.e up pauses after every instruction
- This is very useful during debugging
- its ISR generally displays contents of all registers
- Its ISR address is stored at location ~~two~~ 1x4 = 00004H in IUT

iii) INT 2 (Non Maskable Interrupt)
- The up executes this ISR in response to an interrupt on the NMI line
- ISR address = 2x4 = 00008H in IUT

iv) INT 3 (Break point Interrupt)
- This interrupt is used to cause Breakpoints in the program.
- It is caused by writing instruction INT 03H or simply INT

- Useful in debugging large programs where single stepping is inefficient.
- It's ISR used to display contents of all registers on the screen.
- It's ISR address is stored at location 3X4 = 0000CH in IUT.

v) INT 4 (Overflow Interrupt)
- This interrupt occurs if OF is set AND the up executes the INTO instruction.
- It is used to detect overflow error in signed arithmetic operations.
- ISR address = 4X4 = 00010H in IUT.

b) Reserved Interrupts

INT 5 ... INT 31
- These levels are reserved by INTEL to be used in higher processors like 80386, Pentium etc.
- They are not available to the user.

c) User Defined interrupts

INT 32 ..... INT 255
- These are userdefined, software interrupts
- ISRs for these interrupts are written by the users to service various user defined conditions
- These interrupts invoked by writing the instruction INT n.
- ISR address = nX4 in IUT

D) Hardware interrupts

i) NMI (Non Maskable Interrupt)
- This is non-maskable, edge triggered, high-priority interrupt.

- On receiving interrupt on NMI line, the µp executes INT 2.
- ~~ISR addre~~ µp obtains ISR address from location $2 \times 4 = 00008H$ from IVT
- It reads 4 locations starting from this address to get the values for IP and CS, to execute the ISR.

ii) INTR

- This is maskable, level triggered, low priority interrupt.
- On receiving an interrupt on INTR line, the µp executes 2 $\overline{INTA}$ pulses.

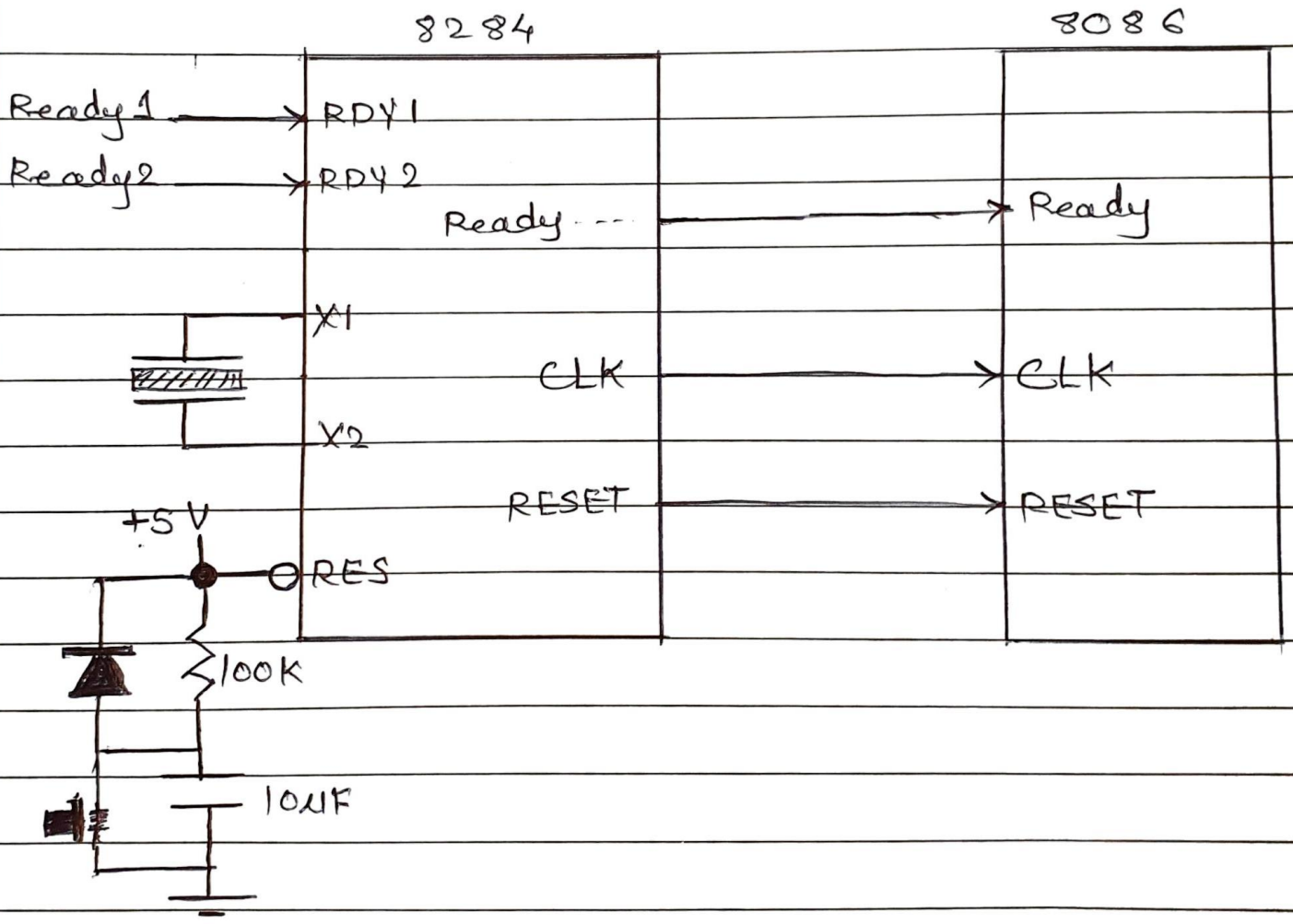
  1st $\overline{INTA}$ pulse -- the interrupting device calculates the vector number.

  2nd $\overline{INTA}$ pulse -- in Interrupting device sends the vector number "N" to the µp

  Now µp multiplied $N \times 4$ and goes to the corresponding location in the IVT to obtain the ISR address.
- It is masked by making IF = 0 ~~by~~ through CLI ins
- It is unmasked by making IF = 1 through STI instruction.

20. Design Power on reset and manual reset circuit for 8086 processor.

→



DATE / / /

PAGE No.
DATE / /

21. What is demultiplexing of address and data bus also explain significance of ALE pin.

→ 8086 has a multiplexed address / data bus.

i) First $A_0 - A_{15}$ address busses are multiplexed with $D_0 - D_{15}$ data busses giving us $AD_0 - AD_{15}$ multiplexed Data & address bus.

ii) Whereas $A_{16} - A_{19}$ address busses are multiplexed with $S_3 - S_6$ status signals giving us $A_{16}/S_3 - A_{19}/S_6$ multiplexed address & status bus.

iii) Interestingly, address and data are never given simultaneously in any operation. Now this is where demultiplexing comes into picture.

iv) In $AD_0 - AD_{15}$, During 1st T, they carry lower order 16 bit address and In the remaing ~~time~~ clock cycles they carry data.

v) In $A_{16}/S_3 - A_{19}/S_6$, During 1st T, they carry higher order 4-bit address and In the remaing time they carry status signals.

Significance of ALE Pin:

ALE has very important role in demultiplexing Address and data busses.

i) When ALE = 1 the respective busses carry $A_0 - A_{15}$, $A_{16} - A_1$ and $\overline{BHE}$

ii) When ALE = 0 the respective busses carry $D_0 - D_{15}$, $S_3 - S_7$.

iii) During 1st T state of any machine cycle ALE is ~~High~~ set of High. At that time the busses carry address and $\overline{BHE}$. ~~There~~

iv) Thereafter BHE remains low for rest of machine cycle, At that time busses carry data and status.

Multiplexed signals	$AD_0 - AD_{15}$	$A_{16}/S_3 - A_{19}/S_6$	$\overline{BHE}/S_7$
when ALE = 1	AD - A15	A16 - A19	$\overline{BHE}$
when ALE = 0	$D_0 - D_{15}$	$S_3 - S_6$	S_7

22.8 Why 8084 is needed in 8086 based system?

→ i) 8084 is a clock generator IC. ~~It provides~~

ii) It provides the $\overline{CLOCK}$ (CLK) signal, a train of pulses at a constant frequency to entire circuit.

iii) It synchronizes the READY signal which indicates that interface is ready for data.

iv) It also synchronizes the RESET signal which is used to initialize the system.

v) 8084 outputs the clock frequency = 1/3rd of the Input clock frequency to produce a 33% duty cycle required by the Microprocessor.

vi) Also, RESET signal is provided by 8284 clock generator. Generation of reset signal is done by using power on reset circuit.

vii) The purpose of this circuit is to activate the reset signal as soon as we supply power to 8086.

viii) This is done so that CS & IP aquire the above values and the system can initialize the monitor program (BIOS) as soon as 8086 is powered on. This is also called COLD starting the system.

PAGE No.
DATE / /

23. Explain ~~Vector~~ interrupt vector table.

i) The IVT contains ISR address for the 256 interrupts.

ii) Each ISR address is stored as CS and IP.

iii) As each ISR address is of 4 bytes (2-CS and 2-IP), each ISR address requires 4 locations to be stored.

iv) There are 256 interrupts: INT 0 .... INT 255
∴ Total size of IVT is 256 × 4 = 1 KB.

v) Whenever an interrupt INT N occurs, µp does N×4 to get values of IP and CS from the IVT and hence perform the ISR.

vi) It contains 3 types of interrupts

a) Dedicated interrupts

i) INT 0 (Divide interrupt): occurs when there is division error.
ISR address = 0×4 = 00000 H

ii) INT 1 (Single step): It puts the µp in single stepping mode after every instruction if TF is set.
ISR address = 1×4 = 00004 H

iii) INT 2 (Non maskable interrupt): used to respond to an NMI line.
ISR address = 2×4 = 00008 H

iv) INT 3 (Breakpoint Interrupt): used to cause breakpoints in the program, useful in debuging longer programs.
ISR address = 3×4 = 0000C H

v) INT 4 (Overflow Interrupt): occurs if OF is set and µp executes INTO instruction.
ISR address = 4×4 = 00010 H

PAGE No.
DATE / /

b) Reserved interrupts

INT 5 to INT 31

These levels are reserved by INTEL to be used in higher processors like 80386, pentium. etc.

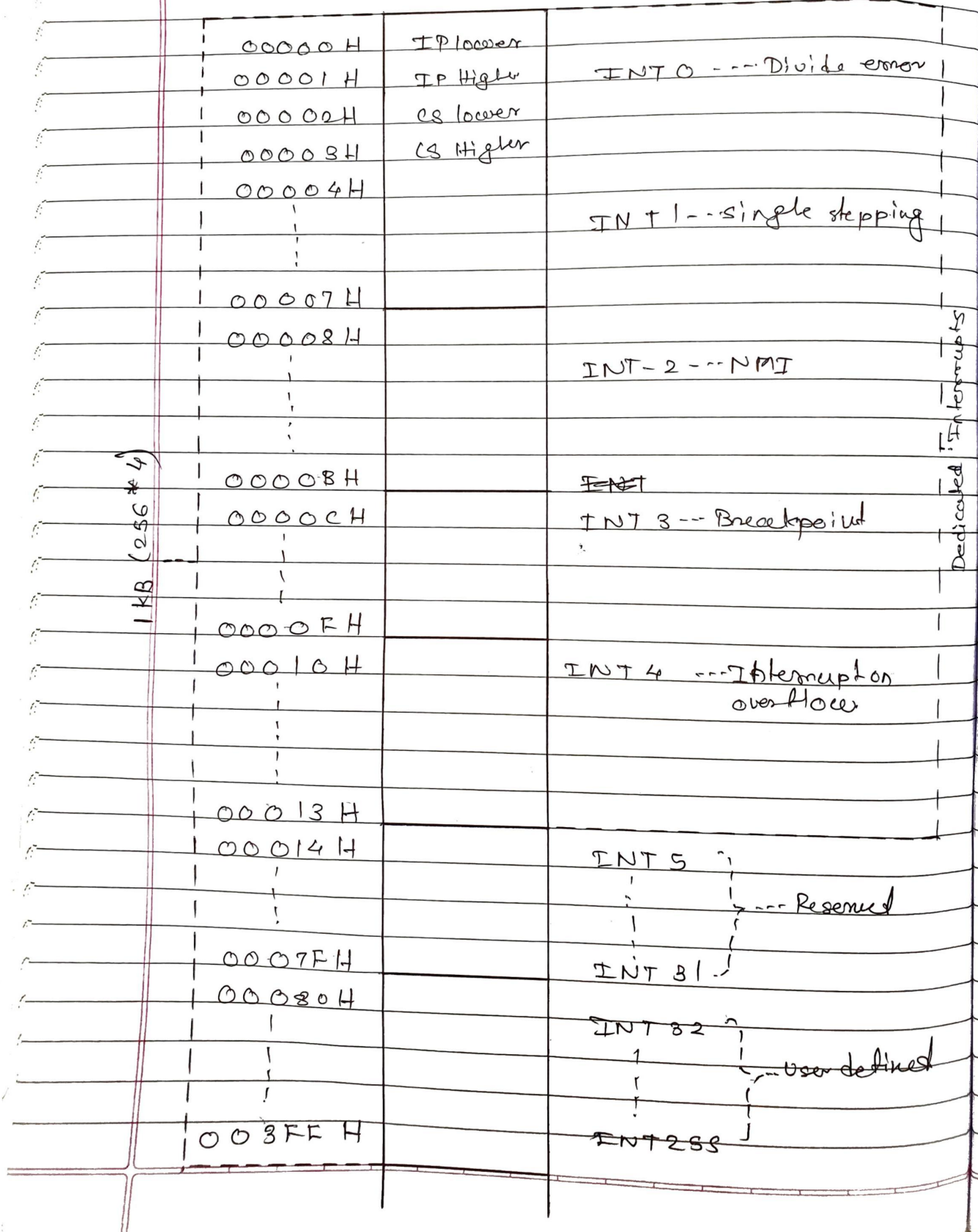
They are not available to user.

c) User defined interrupts

INT 32 ... INT 255

These are user defined software interrupts

These interrupts are invoked by instruction INT n. ISR address = n x 4 in IVT


00000H
00001H
00002H
00003H
00004H
00007H
00008H
0000BH
0000CH
0000FH
00010H
00013H
00014H
0007FH
00080H
003FFH
IP lower
IP Higher
CS lower
CS Higher
1KB (256 * 4)
INT 0 --- Divide error
INT 1 -- single stepping
INT-2 -- NMI
INT 3 -- Breakpoint
INT 4 --- Interrupt on overflow
INT 5
--- Reserved
INT 31
INT 32
--- user defined
INT255
Dedicated Interrupts

Q.24. Explain following instructions.

i) ~~ADD~~ DAA

ii) AAA

iii) XLAT

iv) LAHF

v) SAHF

→

a) DAA:

i) It stands for decimal adjust after addition. It only works on AL register.

ii) It is used when we want to perform addition of two decimal numbers (~~BCD numbers~~) (BCD addition).

iii) We first enter two decimal numbers. We add Then using normal ADD instruction.

iv) Then we perform DAA instruction. DAA will adjust the addition to appear as a decimal addition.

v) The logic of DAA is as follows
- if lower nibble of AL is > 9 or Auxiliary carry flag is "1" then Add 06 to AL.
- if higher nibble of AL is > 9 or carry flag is "1" then Add 60 to AL.

e.g. AL = 14H and CL = 28H

ADD AL, CL gives
AL = 3CH
Now DAA gives
AL = 42H

b) AAA:
i) AAA stands for ASCII Adjust for Addition
ii) It makes the result in unpacked BCD form.
iii) In ASCII codes, 0...9 are represented as 30...39
iv) When we add ASCII codes, we need to mask the higher byte (Eg: 3 of 39)
v) This can be avoided if we use AAA instruction after the addition is performed.
- AAA updates the AF and CF, But OF, PF, SF, ZF are undefined after instruction

e.g. AL = 0011 0100 ... ASCII value of 4.
CL = 0011 1000 ... ASCII value of 8.
Then ADD AL CL gives
AL = 01101100
i.e AL = 6CH ... it is the incorrect temporary result.

Now, AAA gives
AL = 0000 0010 ... Unpacked BCD of 2.
carry = 1 ... this indicates that the answer is 12.

c) XLAT:
Move into AL, the contents of the memory location in Data segment, whose effective address is formed by the sum of BX and AL.
Eg:

```
XLAT        ; AL ← DS:[BX+AL]
            ; i.e if DS = 1000H ; BX = 0200H
                  ; AL = 03H
            ; ∴ 10000  ... DS × 16
            ; +0200    ... BX
            ; +  03    ... AL
            ; = 10203  ∴ AL ← [10203H]
```

In XLAT we can specify the name of the look up table in the instruction
e.g. XLAT sevenSeg.
This will do the translation from the look up table called sevenSeg.
In any case, the base address of look up table must be given by BX.

d) LAHF:
i) It takes no operands.
ii) The instruction is of 1 byte.
iii) This instruction Loads lower byte of Flag register into AH register

e) SAHF:

i) It takes no - operands

ii) It is one byte instruction

iii) It stores the contens of AH into the lower byte of the Flag register

26. Write a program to ADD/SUB two 16 bit number.

→ Addition:

```
.model small
.stack 100H
.data
        n1 DW 1234H
        n2 DW 0F821H
        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
               INC CX
        DOWN: MOV SUM, AX
              MOV SUM+2, DX
              MOV AH, 4CH
              INT 21H
        end start
end.
```

Dedicated Interrupts

Subtraction:

```
.model small
.stack 100H
.data
        n1 DW 0F321H
        n2 DW 1234H
        DIFF DW ?
        ends
.code
    start: 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

→ i) Assembler directives is a statement/ reserved-key-word to give direction to the assembler to perform task of the assembly process

ii) They indicate how an operand or a section of program is to be processed by the assembler

iii) It control the organization of the program and provide ~~assembler~~ necessary info to assembler to understand assembly language program to generate machine codes

~~ii) Assembler supports directives to define data, organize segments to control procedure, to define macros.~~

~~a) .MODEL:~~

~~Defines memory model for the program~~

ii) Assembler supports directives to define data, organize segments to control procedure, to define macros

Examples:

a) DB: stands for Define Byte

Used to define a Byte type variable.

e.g: SUM DB 0

Assembler reserves 1 Byte of memory for the variable named SUM and initialize it to 0.

b) SEGMENT:

Used to indicate the beginning of a code / data / stack / extra segment

c) ENDS: Used to indicate the end of a segment.

e.g:

```
Segname SEGMENT
        ...
        ...
        ...
Segname ENDS
```

} Program & code or instructions

↳ userdefined name of segment

D) .MODEL SMALL

.MODEL defines memory model for the program.

and .MODEL SMALL:

ALL data fit's in one 64KB segment

All code fit's in one 64KB segment

PAGE No.
DATE / /

Q8. Write a assembly language program to transfer data stored in data segment to extra segment.

→

```
Data SEGMENT
        Block1 DB 05H, 35H, 01H, 09H, 78H, 14H,
                  96H, 78H, 15H, 12H.
Data ENDS
Extra SEGMENT
        Block 2 DB 10 Dup(?)
Extra ENDS
Code SEGMENT
        ASSUME CS: Code, DS: Data, ES: Extra.
        MOV AX, Data
        MOV DS, AX
        MOV AX, Extra
        MOV ES, AX
        LEA SI, Block1
        LEA DI, Block2
        MOV CX, 000AH
REP     MOVSB
        INT 3
Code    ENDS
        END
```

Dedicated Interrupt