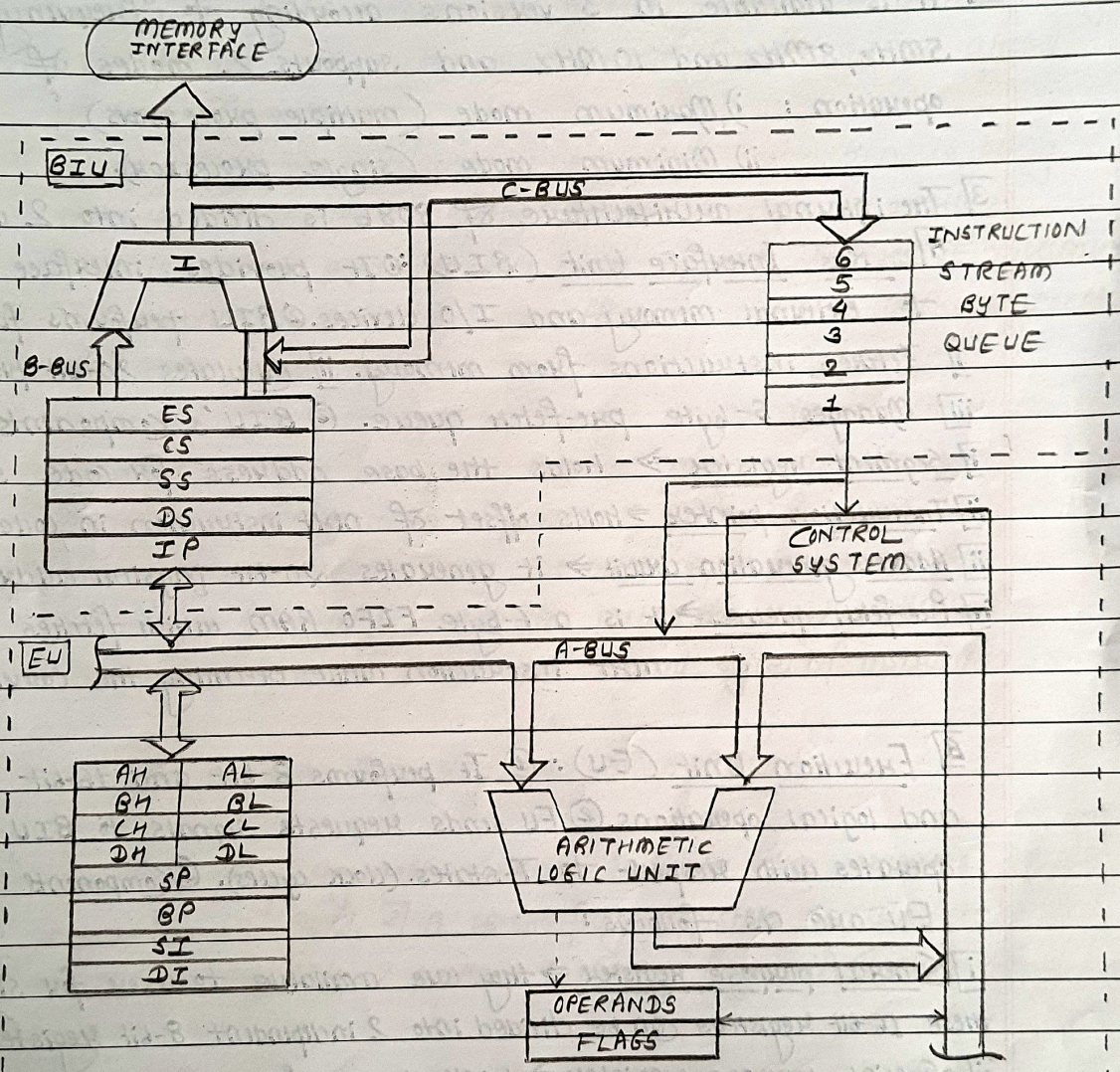
Name : Manjiri Nishikant Naik
Roll no. : 21102B0008
Branch : Computer Engineering (CMPN)
Division : B
Semester : IV

classmate
Date
Page 1

Microprocessor (MP) : Assignment 1

1] Draw and explain the architecture of 8086.

Soln ⇒



Architecture of 8086 microprocessor

⇒ 1] 8086 processor is an enhanced version of 8085 processor introduced by Intel in 1976. It has a 16-bit data bus and 20-bit address bus and supports 2 stages of pipelining.
2] It is available in 3 versions according to frequency as 5MHz, 8MHz and 10MHz and supports 2 modes of operation : i) Maximum mode (multiple processors)
ii) Minimum mode (single processor).
3] The internal architecture of 8086 is divided into 2 units :
A] Bus Interface Unit (BIU) : ① It provides interface of 8086 to external memory and I/O devices. ② BIU performs functions like i] Fetches instructions from memory. ii] Calculates 20-bit physical addrss. iii] Manages 6-byte pre-fetch queue. ③ BIU's components are :
i] Segment register ⇒ Holds the base address for code segment.
ii] Instruction pointer ⇒ Holds offset of next instruction in code segment.
iii] Address generation circuit ⇒ it generates 20-bit physical address
iv] Pre-fetch queue ⇒ it is a 6-byte FIFO RAM which fetches the next instruction while executing the current one.

B] Execution Unit (EU) : ① It performs 8-bit and 16-bit arithmatic and logical operations. ② EU sends requests signals to BIU and it operates with respect to T-states (clock cycles). ③ Components of EU are as follows :
i] General purpose register ⇒ they are available to users for storing values; these 16-bit registers can be divided into 2 independent 8-bit registers ($A_x \rightarrow A_L, A_H$).
ii] Special purpose register ⇒ points to specific memory locations of segment.
ⓐ Stack pointer (SP) ⇒ Points stack-top. ⓑ Base Pointer (BP) ⇒ holds offset address.
ⓒ Source index (SI) ⇒ Holds offset in data segment during string operations.
ⓓ Destination index (DI) ⇒ holds offset in extra segment during string operations.
iii] ALU ⇒ Performs 8-bit and 16-bit arithmatic and logical operations.
iv] Status register ⇒ Has 9 flags that change/recognize state of MP.
1] Carry 2] Parity 3] Aux. carry 4] Zero 5] Sign 6] Overflow 7] Trap 8] Interrupt 9] Direction
(Status) (Control)
v] Operands ⇒ 16-bit register to hold the operands temporarily.

2] Explain pre-fetch queue of 8086.

Sol^n ⇒ 1] The instruction pre-fetch queue of 8086 is 6-bytes in length operating in FIFO manner and receives instructions from memory.
2] BIU fetches the instructions meant for the queue ahead of time from memory.
3] Because of this instruction queue, one instruction is being executed while other is being fetched causing an overlap. This is called pipelining and 8086 supports 2-stages of pipelining.
4] This pipelining increases the efficiency of the microprocessor.
5] MP 8086 starts operation by fetching 1 or 2 byte/s of instruction. The 1st byte is always an opcode which when decoded, 1 byte in queue becomes empty and gets updated.
6] Queue is refilled when atleast 2 bytes of queue are empty
7] For a single opcode byte, the next bytes are treated as data bytes depending upon the instruction length, otherwise, the next byte is treated as second byte of instruction opcode.

3] ① Draw and explain flag register of 8086.

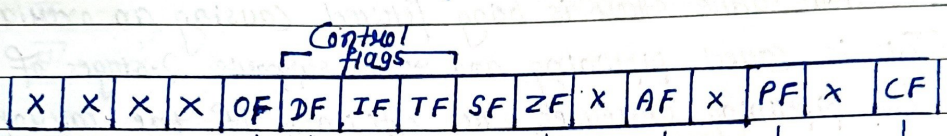
Sol^n ⇒ 1] The 8086 flag register contents indicate the results of computation in ALU. 2] A 16-bit flag register is used in 8086 and is divided into 2 parts : i) Status flags ii) Control flags.

A] Status Flags

① Carry flag ⇒ set to logic 1 whenever there is carry (or borrow).
② Parity flag ⇒ set to logic 1 if the result has even parity.
③ Auxiliary carry flag ⇒ set if carry is generated out of nibble (lower).
④ Zero flag ⇒ set if the result is zero.
⑤ Sign flag ⇒ set if MSB of result is 1 i.e., it is a -ve for signed nos.
⑥ Overflow flag ⇒ will be set if result of operations is too large to fit in no. of bits available.

8 Control Flags

① Interrupt flag ⇒ used to mask or disable interrupt.
② Direction flag ⇒ if it is set, SI and DI are in auto-decrementing mode in string operations.
③ Trap flag ⇒ used to set the trace mode, i.e., start single-stepping mode.



Overflow flag
1 = overflow occurred
0 = no overflow occurred

Direction flag
1 = Auto decrement
0 = Auto increment
(used in string instructions)

Interrupt flag
1 = Enable interrupt
0 = Disable interrupt
(Affects only INTR)

Trap flag
1 = perform single stepping
0 = do not perform single stepping.

Sign flag
1 = MSB of result is 1 (∵ -ve)
0 = MSB of result is 0 (∵ +ve)

Zero flag
1 = Result = 0
0 = Result ≠ 0

Auxillary carry flag
1 = carry from lower to higher nibble
0 = No such carry
(used in 8-bit operations)

Parity flag
1 = even parity
0 = odd parity

Carry flag
1 = carry out of MSB
0 = No such carry

Flag register of 8086

4] Write a formula to calculate physical address. Explain why its required to calculate it.

Soln. ⇒

1] 8086 addresses a segmented memory and it has a 20-bit long address bus.
2] Thus, to access a specific memory location from any segment, we need a 20-bit address.
3] However, all the registers in 8086 microprocessor are of 16-bit length.
4] Instruction pointer (IP) holds 16-bit address of next code byte within code segment and is referred to as offset.
5] Thus, to produce 20-bit address using 16-bit registers, the offset address needs to be added to the segment base address.

Formula for calculating physical address :

Physical address = Segment address × 10H + Offset address

Addressing modes [Imp for MSE ⇒ either write types of AM ⇒ identify which AM.

A] Memory addressing mode

1] Immediate addressing mode

⇒ operand is specified in instruction itself
⇒ source operand is a 8-bit or 16-bit data
⇒

ex.⇒ MOV CX, 2000H
MOV CL, 0AH
ADD CL, 45H

2] Register addressing mode

⇒ both operands are registers, i.e., operands are specified through registers.

ex.⇒ MOV CL, DL
XOR AX, BX
ADD AL, BL

3] Direct addressing mode

⇒ addressing of operand is directly given in instruction. as displacement (offset address).

ex.⇒ MOV CL, [2000H]
⇒ moves data from location 2000H in data segment of CL.
⇒ physical address is calculated as DS * 10H + 2000H (offset)
Assume DS = 5000H ∴ PA = 50004+2000 = 52000 ∴ CL = 52000H

ex.⇒ MOV CX, [2000H]

4] Indirect addressing mode

① Register indirect addressing mode

⇒ address of operand is given using a register.

ex. ⇒ MOV CL, [BX]

⇒ operand is at → 5000H of data segment of operand
⇒ first more address from 5000H to BX
⇒ then MOV CL, [BX]

② Register relative addressing mode

⇒ address of operand is given as sum of register and displacement.

ex. → MOV CL, [BX+4] or MOV CL, 04H [BX]

if BX = 5000H ⇒ actual location.
BX+4 = 5004H.

③ Base Indexed addressing mode

⇒ only BX register is used for addressing along with offset address register.

Base → only B register
Offset → SI and DI for data seg and BP for stack seg.

ex. ⇒
- MOV CL, [BX+SI] → Data seg < SI, DI
- MOV CL, [BP+SI] → Stack seg < SI, DI

④ Base relative plus indexed addressing mode

ex. ⇒ MOV CL, [BX + DI + 20]
MOV [BP + SI + 2000], CL

5] Implied addressing mode

⇒ operands are implied, not specified in instructions.

ex. ⇒ STC → set the carry
CLD → clears the direction flag.

B] I/O Addressing mode

I/O addressing modes

Direct

⇒ 8 bit I/O address
⇒ 00H to FFH
⇒ 256 I/O port address
ex. → IN AL, 80H

Indirect

⇒ 16-bit I/O address
⇒ 0000H to FFFFH
⇒ 65536 I/O port address
ex. ⇒ MOV DX, 2000H

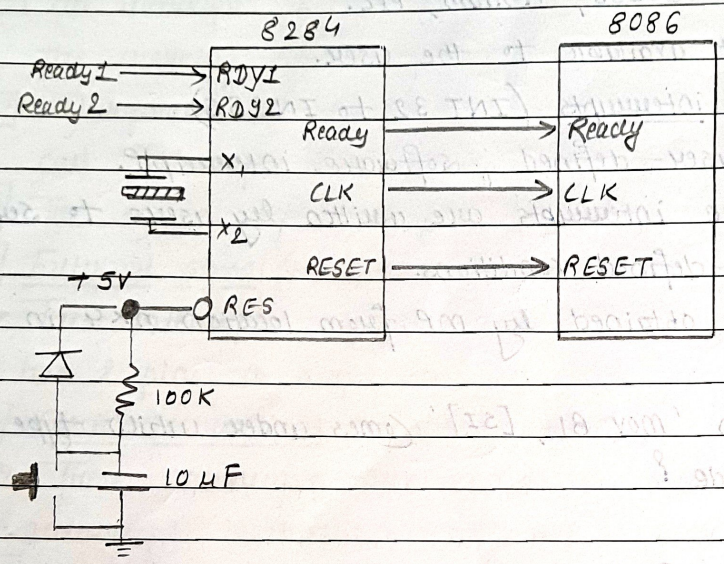
[data = 8 bit
address = 16-bit]

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

Soln. ⇒ Here, address of the operand is given through register SI
∴ This is register indirect addressing mode.

9] Why 8284 is needed in 8086 based system ?

Sol:⇒



1] The 8284A is used in an 8086 microprocessors to generate the system clock signal.

2] CLK is the processor clock and has 33 % duty cycle clock signal.

3] When power is first applied, the capacitor will charge towards +5V.

4] With sufficient time constant, the $\overline{RES}$ input will be held long enough to reset the processor. Hence, 8284 generates reset signal.

5] It also synchronises ready signals from memory and I/O devices.

10] List features of 8086 microprocessor.

Soln ⇒ 1] 8086 is an enhanced version of 8085 processor introduced by Intel in 1976. 2] 8086 has 20-bits of address bus and 16-bits of data bus. and supports 2 stages of pipelining. 3] Memory of 8086 is divided into 4 segments : i] Code ii] Data iii] Stack iv] Extra 4] 8086 supports enhanced version of instruction set (multiplication and division) 5] It supports 2 operating modes : i] minimum mode (single processor) ii] maximum mode (multiprocessor) 6] It has instruction pre-fetch queue storing 6 bytes of instruction from memory. 7] It operates in 3 frequencies : i] 5MHz ii] 8MHz iii] 10MHz 8] It uses single phase clock with 33% of duty cycle and also 256 vector interrupts.

11 What is mimory addressing capacity of 8086 and why ?

Sol^{n}. ⇒ 1] The memory addressing capacity of 8086 microprocessor is 1MB.

2] This is because 8086 is a 16-bit microprocessor having 20-bit address bus and 16-bit data bus.

3] We know that on N-bit address bus, we can access 2^N mimory locations.

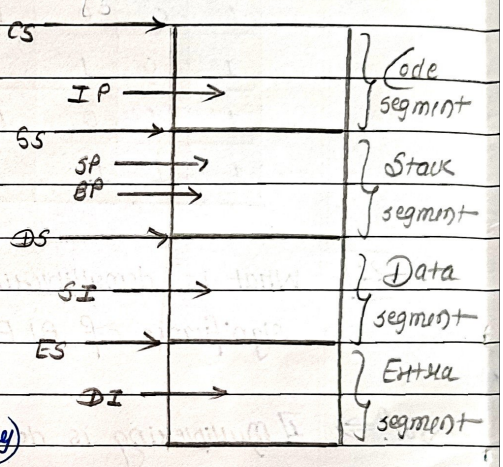
4] Thus, for 20-bit address bus, memory locations = 2^{20} bytes = 1MB

5] Therefore, the memory addressing capacity of 8086 is 1MB.

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

Sol?⇒ 1] Memory segmentation ⇒ It is the process in which the main memory of the ~~processor~~ computer is divided into different segments/partitions each one having its own base address.

2] In 8086 microprocessor, the memory is divided into 4 segments :



A] Code segment :
⇒ Base : CS ⇒ Offset : IP
⇒ it stores all instructions and programs

B] Stack segment :
⇒ Base : SS ⇒ offset : SP and BP
⇒ it stores all the ~~general~~ stack data (memory)

C] Data segment :
⇒ Base : DS ⇒ offset : SI
⇒ it stores all the general data.

D] Extra segment :
⇒ Base : ES ⇒ offset : DI
⇒ it stores general data and it is used as destinating in string operations.

Advantages of memory segmentation :

1] It divides the memory logically to store instructions, data and stack seperately.
2] It permits the programmer to access 1 MB memory using 16-bits of address.

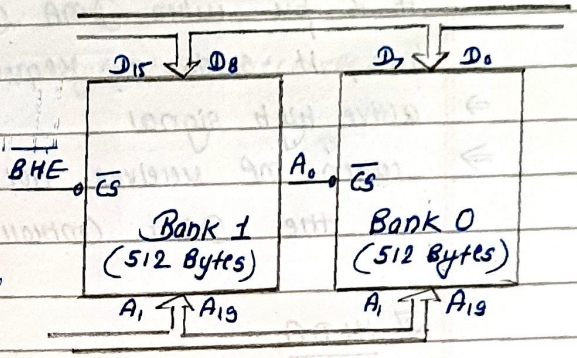
Memory Banking

⇒ Memory address space is equally divided into 2 parts (banks).

⇒ One of the banks containing even addresses is called Even Bank and the other one containing odd addresses is called Odd Bank.

⇒ Even bank always stores lower byte ∴ it is called Lower Bank (LB)

⇒ Odd bank always stores higher byte ∴ it is called Higher Bank (HB)



Why it is done in 8086? ⇒ 8086 has 20-bit addressing model for memory access. Each address represents a single byte, but natural word size of 8086 is 2 bytes (16-bit data bus) ∴ we need a way to read 2 bytes at one time ∴ Even and Odd banks.

$\overline{BHE}$	A0	Operation
0	0	Read or write 16 bit from both banks
0	1	Read or write 8 bits from higher bank
1	0	Read or write 8 bits from lower bank
1	1	No operation (Processor is idle)

ex. ⇒ 8 bit operation on even banks

```
MOV BL, [20000H]
MOV BH, [20000H]
MOV BX, [20000H]   (aligned operation)
MOV BX, [20001H]   (misaligned operation)
```

MOV BL [20000H]

⇒ 8 bit operation

⇒ BL will get value stored at offset address [20000H]

⇒ 8 bit operation ∴ only 1 bank selected

⇒ 20000H ⇒ even/lower bank

⇒ $\overline{BHE}$ → 1, A_0 = 0

14] Explain minimum mode of operation. [supports single processor]

Sol^n ⇒

1] 8086 work in minimum mode when MN/$\overline{MX}$ pin is set to logic 1.

2] In this mode, 8086 is the only processor in the system.

3] Clock, reset and ready signals are generated by 8284 clock generator.

4] Address from the address bus is latched into 8282 8-bit latch.
Thus, 3 latches are needed as address bus is 20-bit.

5] Data bus is given through 8286 transreceiver of 8-bit.
Thus, 2 transreceivers are needed as data bus is 16-bit.

6] Transreceivers are enabled through $\overline{DEN}$ signal and direction of data is controlled by DT/$\overline{R}$ signal. (Both generated by 8086).

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

7] 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

8] $M/\overline{IO}$, $\overline{RD}$, $\overline{WR}$ are decoded by a 3:8 decoder.

9] Bus request (DMA) is done using HOLD and HLDA signals.

10] $\overline{INTA}$ is given by 8086 in response to interrupt on INTR line.

15] Explain maximum mode of operation. [supports multiple processors]

Soln ⇒ 1] 8086 work is maximum mode when MN/$\overline{MX}$ pin is logic 0.

2] In this mode, multiple processors are supported along with 8086.

3] Clock signals are provided by 8284 clock generator.

4] 8288 Bus Controller is the most significant part as it provides various control signals.

5] Address from address bus is latched into 8282 8-bit latch. 3 such latches are needed for 20-bit address bus. ALE is connected to STB of latch and ALE is given by 8288 bus controller.

6] Data bus is driven through 2 8286 8-bit transreceivers as data bus is of 16-bit. Transreceivers are enabled through DEN signal and direction of data is controlled by DT/$\overline{R}$ signal.
DEN is connected to OE and DT/$\overline{R}$ is connected to T.

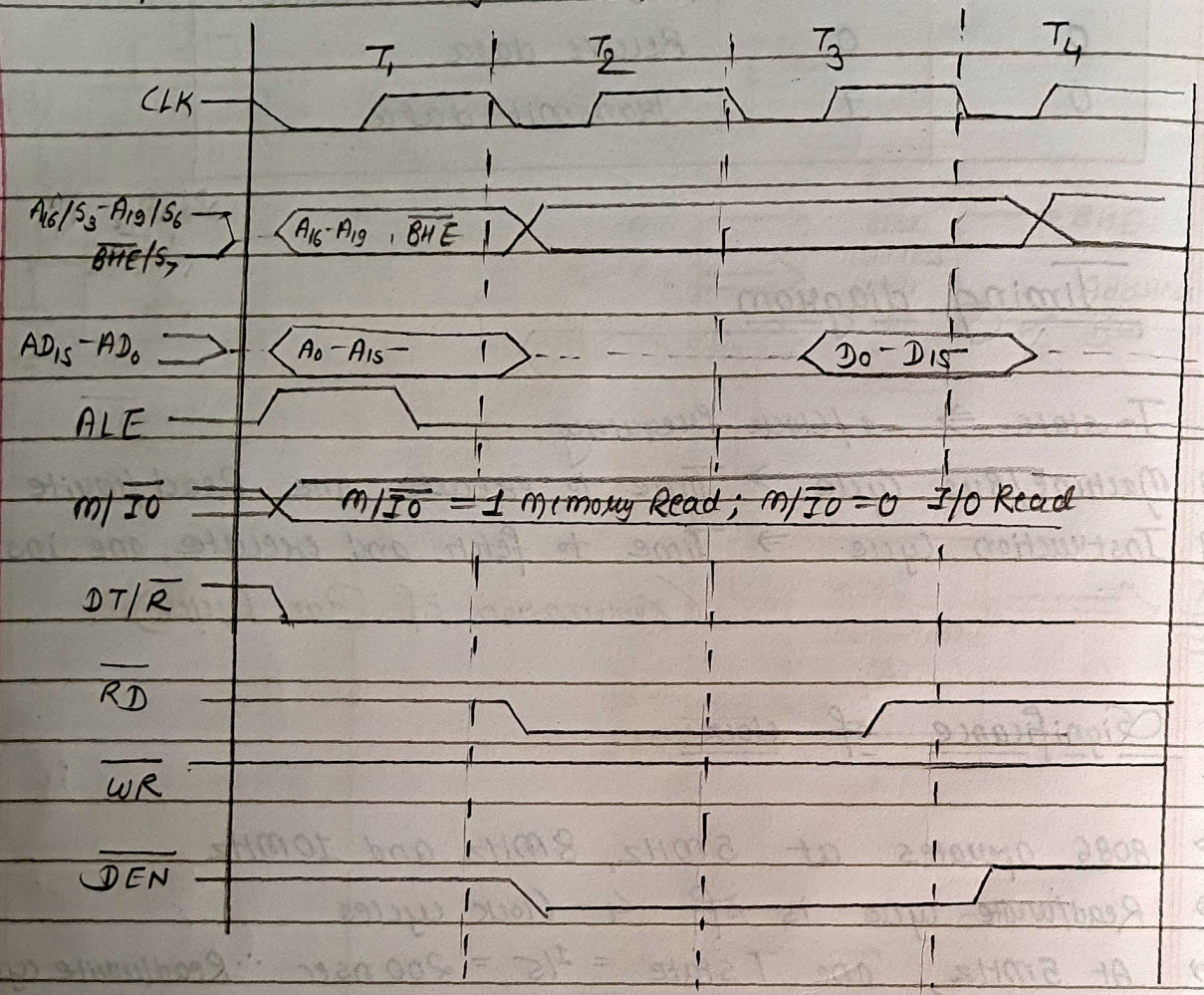
DEN(8288)	DT/$\overline{R}$	Action
0	X	Transreceiver disable
1	0	Receive data
1	1	Transmit data

7] 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 Output
0	0	0	Int. Acknowledge	$\overline{INTA}$
0	0	1	Read I/O port	$\overline{IORC}$
0	1	0	write I/O port	$\overline{IOWC}$ and $\overline{AIOWC}$
0	1	1	Halt	None
1	0	0	Instruction fetch	$\overline{MRDC}$
1	0	1	Memory read	$\overline{MRDC}$
1	1	0	Memory write	$\overline{MWTC}$ and $\overline{AMWTC}$
1	1	1	Inactive	None

Minimum mode read cycle

$M/\overline{IO} = 1$ then mimory read ; $M/\overline{IO} = 0$ then I/O read



Minimum Mode Read Cycle

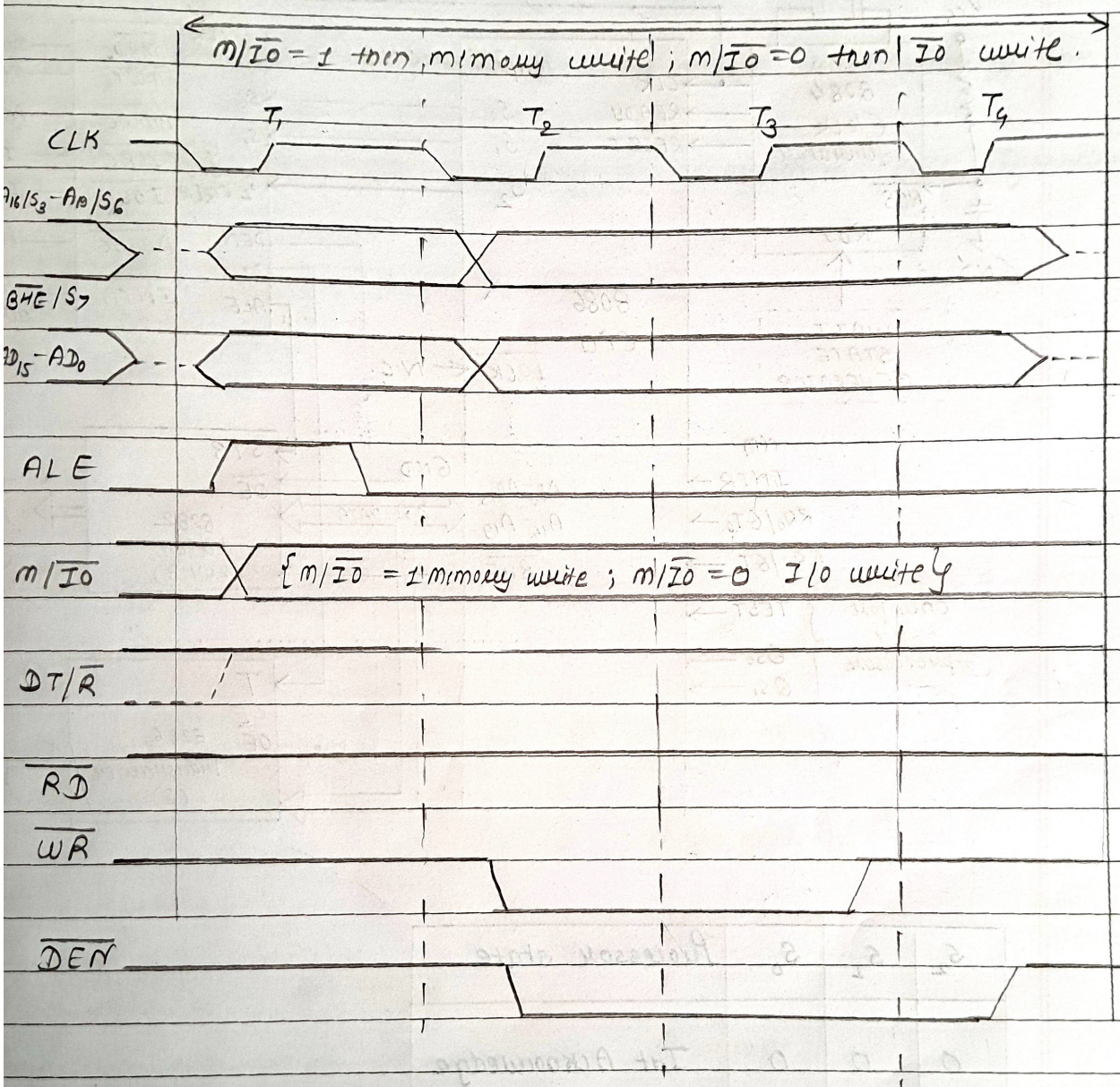
$T_1 \Rightarrow$ ALE goes high as the multiplexed buses carry address till it goes low.

$T_2 \Rightarrow$ MP asks for data by making $\overline{RD}$ low and $\overline{WR}$ high.
Transreceiver is enabled by $\overline{DEN} = 0$
Data takes some time called 'Propogation delay' to reach MP from memory.

$T_3 \Rightarrow$ mimory starts sending data on the data bus and MP checks ready signal : if ready = 1 $\Rightarrow$ MP completes machine cycle in next T_4 state ; if ready = 0 $\Rightarrow$ MP inserts a wait state between T_3 and T_4.

$T_4 \Rightarrow$ MP captures data sent by memory device.
All signals like $\overline{RD}$ and $\overline{DEN}$ are deactivated.

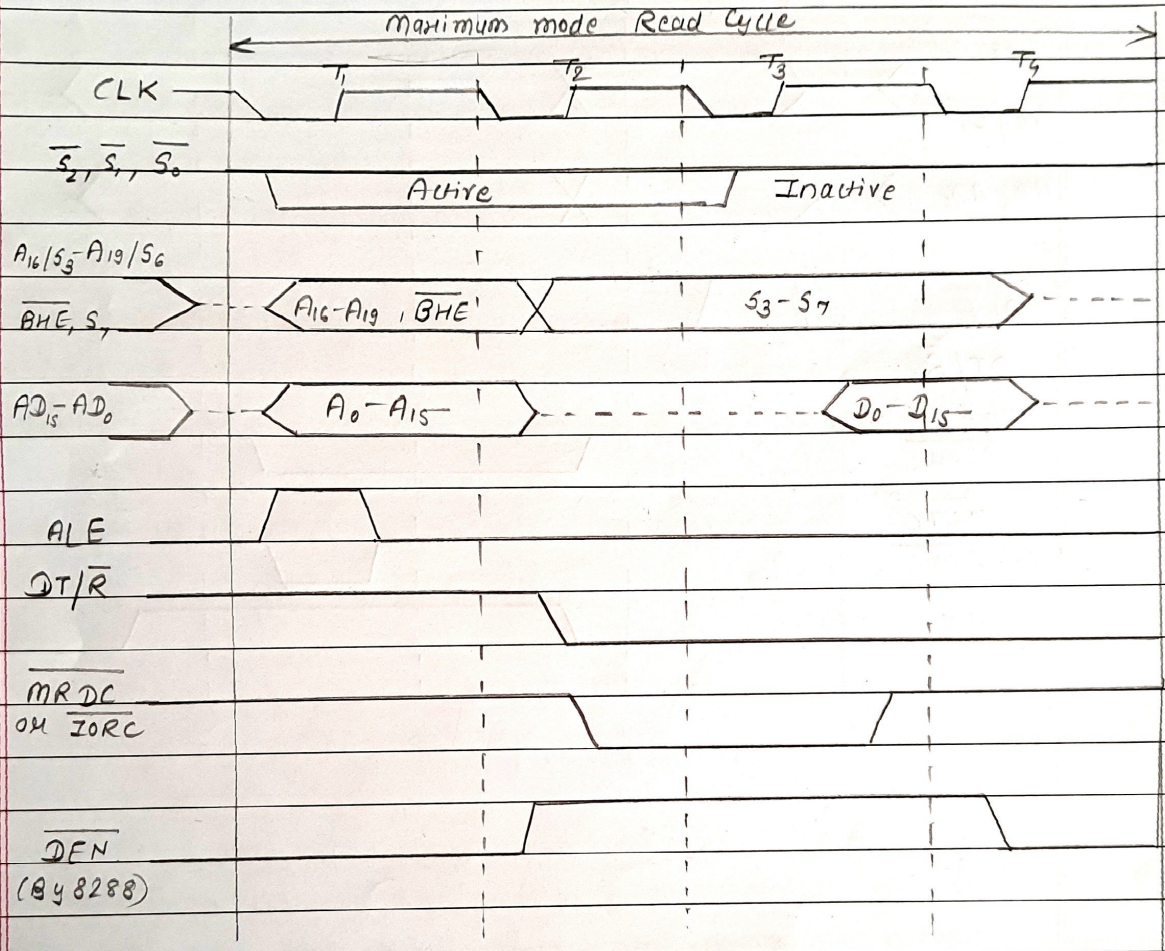
Minimum mode write cycle



Minimum mode write cycle

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

Soln. ⇒ A] Maximum mode - Read Cycle



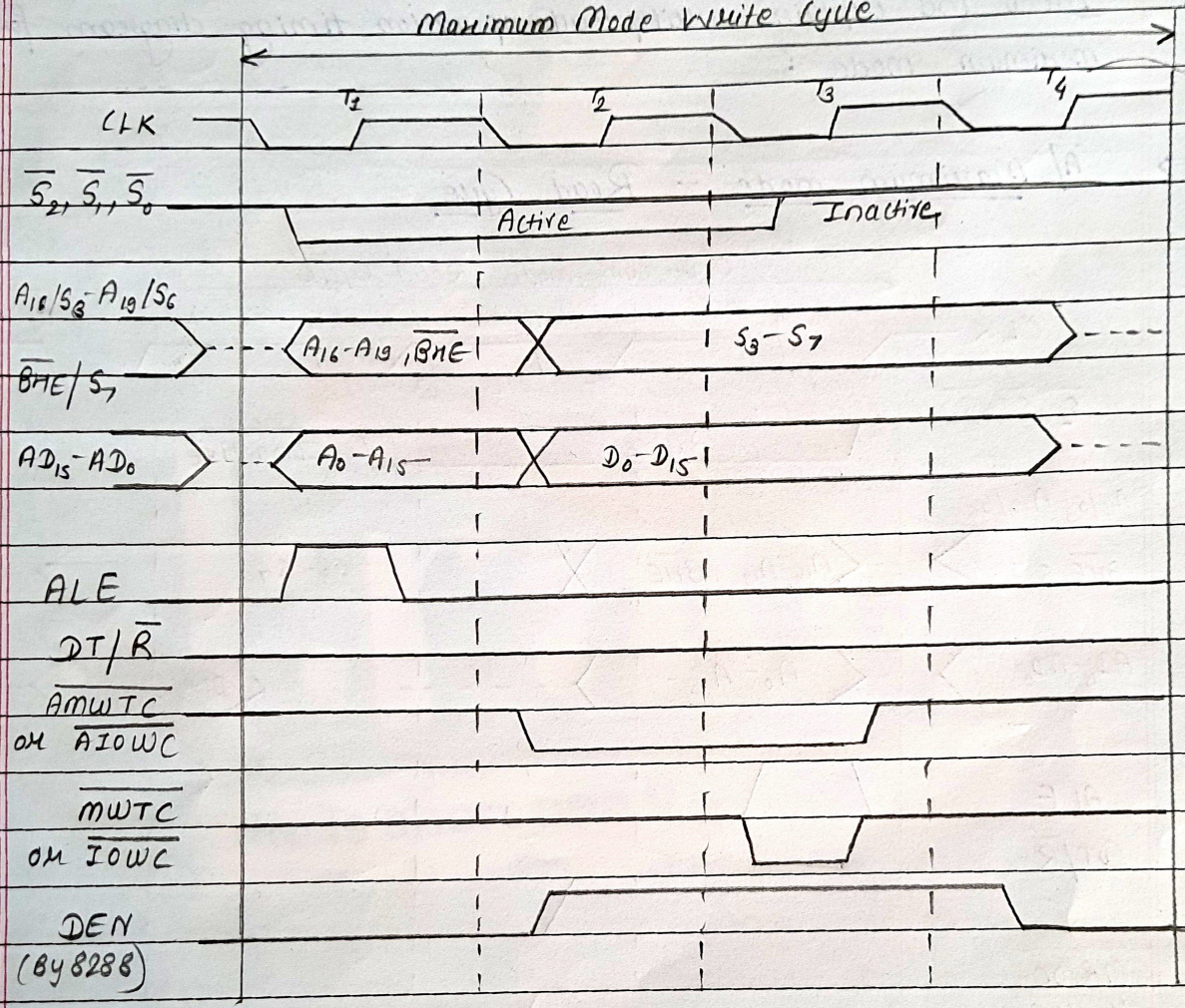
⇒ T_1 ⇒ ALE is high and multiplexed buses carry address and $\overline{S_2}, \overline{S_1}, \overline{S_0}$ are active.

T_2 ⇒ $AD_{15} - AD_0$ buses carry address until ALE goes low and $\overline{BHE}/S_7$ is allowed to carry address.

T_3 ⇒ Address is no longer present in multiplexed bus, transreceiver (half T_2) is enabled by DEN = 1. Bus will now carry signals $S_3 - S_7$ until mid of T_4. Data will take time to travel to MP.

T_4 ⇒ Memory starts sending data on data bus.

B] Maximum mode - Write Cycle



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

Sol^n ⇒

Minimum mode	Maximum mode
1] MN/$\overline{MX}$ pin is set to logic 1.	1] MN/$\overline{MX}$ pin is set to logic 0.
2] There can be only one processor.	2] There can be multiple processors.
3] The 8086 generates INTA for interrupt acknowledgement.	3] The 8288 Bus Controller generates the interrupt acknowledgement signal (INTA).
4] Performance is slower as multiprocessing cannot be performed.	4] Performance is faster as multiprocessing can be performed.
5] The 8086 itself provides an ALE for the latch.	5] The 8288 Bus Controller provides ALE for the latch.
6] It is used for small systems.	6] It is used for large systems.
7] The system is more affordable.	7] The system costs more money.
8] $\overline{DEN}$ and DT/$\overline{R}$ signals for the transreceivers are given by 8086 itself.	8] DT/$\overline{R}$ signals for the transreceivers are given by 8288 bus controller.

23] Explain interrupt vector table. / 19] Explain types of interrupts.

Soln: ⇒ 1] An interrupt is a special condition that arise during the working of microprocessor.
2] Microprocessor services it by executing a subroutine call Interrupt Service Routine (ISR).
3] The sources of interrupts for 8086 are:
i] External signal (Hardware interrupt):
⇒ These interrupts signals occur on external pins of MP. 8086 has 2 pins to accept hardware interrupt ⓘ NMI ⓘⓘ INTR
ii] Special instructions (software interrupt):
⇒ These interrupts are caused by writing the software interrupt instruction which are assigned values in the IVT.
4] Interrupt Vector Table (IVT):
⇒ IVT has some pre-defined values for particular interrupts and can also accommodate user-defined interrupts.
⇒ Each interrupt has dedicated 4 bytes (2-CS and 2-IP) and it contains total 256 (0 to 255) interrupts
⇒ Thus, size of IVT = 4×256 = 1024 bytes = 1KB
⇒ IVT is divided into 3 parts as follows:

A] Dedicated interrupts (INT 0 to INT 4)
① INT 0 (Divide error) ⇒ this interrupt occurs whenever there is division error
⇒ i.e, when result of division is too large to be stored
② INT 1 (Single step) ⇒ it puts MP in single stepping mode
⇒ useful for debugging ⇒ displays contents of all registers.
③ INT 2 (NMI) ⇒ MP executes this ISR in response to interrupt on NMI line
④ INT 3 (Breakpoint) ⇒ used to cause breakpoints in program
⇒ Caused by writing INT03H or INT ⇒ used for debugging.
⑤ INT 4 (Overflow) ⇒ occurs if overflow flag is set ⇒ used to detect overflow error in signed operations.

B] Reserved interrupts (INT 5 to INT31)

⇒ These levels are reserved by Intel to be used in higher processors like 80386, Pintium, etc.

⇒ They are not available to the user.

C] User-defined interrupts (INT 32 to IN255)

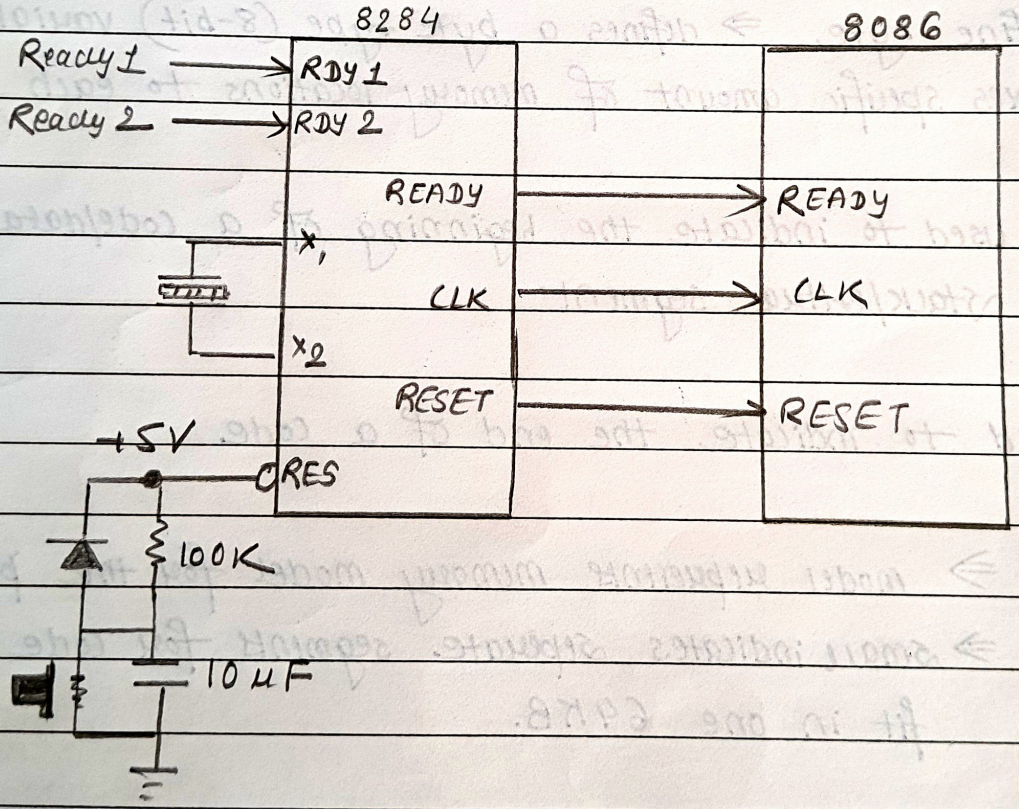
⇒ These are user-defined, software interrupts.

⇒ ISRs for these interrupts are written by users to service various user-defined conditions.

⇒ Its ISR is obtained by MP from location n×4 in IVT.

20] Design the power on Reset and manual Reset circuit for 8086 microprocessor.

Soln: →



21] What is demultiplexing of address and data bus? Also explain the significance of ALE pin.

Sol?⇒ 1] Multiplexing is done to reduce no. of pins of 8086 microprocessor.
2] However, external memory needs to seperate the address for decoding and data transfer.
3] The signal lines AD_0-AD_{15} in 8086 are used for dual purpose:
i] To carry address bits ii] To carry data bits.
This is caled multiplexing of address and data bus.
4] The address bus is required to locate the memory for data transfer to I/O devices and the data bus is used to transfer data from MP to memory devices and vice-versa.
5] To avoid mixing of address and data buses, they must be demultiplexed. This can be done with the help of an ALE latch.
The bus is connected as an input to the latch.
6] ALE pin ⇒ It is an Address Latch Enable signal which indicates that valid address is available on bus AD_0-AD_{15}.
7] It is an active high signal and remains high during T_1 state.
8] If ALE is logic 1 (high), it enables the address bits (A_0-A_{15}) and if ALE is logic 0 (low), it enables the data bits (D_0-D_{15}).

24] Explain following instructions:

Sol:⇒ a] DAA ⇒ Decimal Adjust Accumulator

⇒ It is used to adjust the decimal after the addition/subtraction operation ⇒ it basically converts the result of binary sum into BCD number.

b] AAA ⇒ It is used to adjust ASCII after addition.

⇒ This instruction is mainly useful for adding strings of digits where there is exactly one decimal digit per byte in a string.

c] XLAT ⇒ This instruction is used to translate a byte in AL using a table in the memory. It changes AL register from the table index to table entry

d] LAHF ⇒ Load lower byte of flag register in AH. This instruction copies the contents of lower byte of 8086 flag register, to AH register.

25] Explain string instructions of 8086 microprocessor.

Sol?⇒ String instructions of 8086 microprocessor are as follows:

① MOVS:

i] MOVSB ⇒ moves contents of byte given by DS:SI into ES:DI

ii] MOVSW ⇒ moves contents of word given by DS:SI into ES:DI

② LODS:

i] LODSB ⇒ moves byte address at DS:SI to AL:SI ; SI inc/dec by 1.

ii] LODSW ⇒ moves word address at DS:SI into AX:SI ; SI inc/dec by 2.

③ STOS:

i] STOSB ⇒ moves contents of AL to byte address given by ES:DI ; DI inc/dec by 1

ii] STOSW ⇒ moves contents of AX to byte address given by ES:DI, DI inc/dec by 2.

④ CMPS:

i] CMPSB ⇒ compares byte at ES:DI with byte at DS:SI and sets flags

ii] CMPSW ⇒ compares word at ES:DI with word at DS:SI and sets flags.

⑤ SCAS:

i] SCASB ⇒ compares byte at ES:DI with AL and sets flag.

ii] SCASW ⇒ compares word at ES:DI with AX and sets flag.

⑥ REPZ/REPE ⇒ repeat the given instruction while CX=0 i.e., ZF=1

⑦ REPNZ/REPNE ⇒ repeat the given instruction while CX!=0 i.e., ZF=0

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

Sol^n ⇒

i) Addition:

Program ⇒

```
.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
        inc cx
down:   mov sum, ax
        mov sum+2, dx
        mov ah, 4ch
        int 21h
    end start
end
```

ii) 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

Sol^{n} ⇒ 1] Assembler directives ⇒ It is a statement/reserved key-word to give direction to the assembler to perform task of the assembly process.

2] They indicate how an operand or a section of the program is to be processed by the assembler.

3] It controls the organization of the program and provide necessary information to the assembler to understand the assembly language programs to generate necessary machine codes.

4] An assembler supports directives to define data, to organize segments, to control procedure, to define macros.

DB ⇒ Define byte ⇒ defines a byte type (8-bit) variable
⇒ reserves specific amount of memory locations to each variable.

SEGMENT ⇒ used to indicate the beginning of a code/data/stack/extra segment

ENDS ⇒ used to indicate the end of a code.

.MODEL SMALL ⇒ model represents memory model for the program.
⇒ small indicates seperate segments for code and data fit in one 64KB.

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

Sol^n ⇒

Labels	Mnemonics	Operands
START	MOV	AX, @DATA
	LEA	DS
	MOVSB	CX
	REPNZ	LEN
	INT	ARR1
		ARR2
		SI
		DI
		21H, 4CH
		ES

Program ⇒

```
.model small
.stack 100h
.data
        arr1 db 02h,09h,06h,10h,07h
        arr2 db ?
        len dw $-arr1
        ends
.code   start   mov ax,@data
                mov ds, ax
                mov es, ax
                mov cx, len
                lea si, arr1
                lea di, arr2
                repnz movsb
                mov ah, 4ch
                int 21h
        ends
end     start
```