ACY- 2018-2019

Sri Shridevi Charitable Trust (R.)

SHRIDEVI INSTITUTE OF ENGINEERING & TECHNOLOGY

(An ISO 9001:2008 Ceritfied Institution)

(Recogized by Govt. of Karnataka,

Affiliated to VTU, Belagavi & Approved by the AICTE, New Delhi)

NH-4, Sira Road, TUMAKURU - 572106, Karnataka.

SHRIDEVI EDUCATION

DEPARTMENT OF Electronics and Communication

ASSIGNMENT BOOK

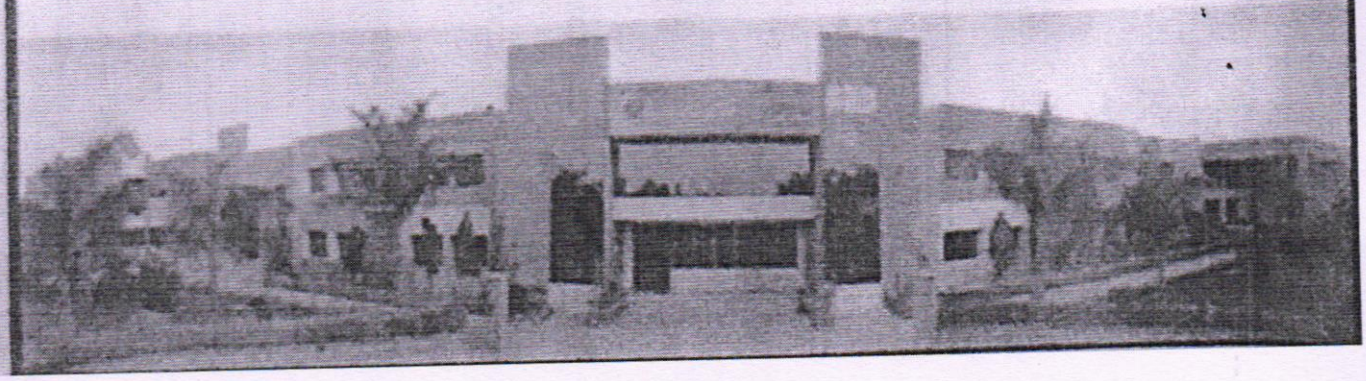
Name : Mr. / Ms. Daniel.S

Subject : Microprocessor

Subject Code : 17EC46

Semester : IV Sem

USN : 1 S V 1 7 E C 0 0 3



PRINCIPAL
SIET., TUMAKURU.

ASSIGNMENT MARKS

Date	Assignment No.	Max. Marks	Marks Obtained	Course Instructor Signature
20/2/19	1	10	10	20/2
5/03/19	2	10	10	5/3
03/04/19	3	10	10	3/4
29/04/19	4	10	10	29/4
10/05/19	5	10	10	10/5
	Average	10	10	23/5

CERTIFICATE

This is to certify that Mr./Ms. Daniel.S

with USN 1SV17EC003 has satisfactorily completed the course of assignments in the subject of Microprocessor as prescribed by the Visvesvaraya Technological University for the 2nd / IV Sem year / semester E & C B.E. / M.Tech. MBA degree course in the year 2018 -2019

Signature of the Student

23/5/19
Course Instructor

23/5/19
Head of the Department

PRINCIPAL
SIET., TUMAKURU.

Assignment-1

1. What is microprocessor? briefly Discuss the evolution of Microprocessor.

⇒ The microprocessor is a program controlled semiconductor device (IC), which fetches (from memory), decodes and executes instructions. it is used as CPU in computers The Basic functional blocks of a Microprocessor are ALU (Arithmetic and logic unit), an array of Registers and a control unit. The Microprocessor is identified with time. The 8086 processor. The 80486, size of data, the ALU of the processor can work with time. The 8086 processor has 16 bit ALU, hence it is called 16-bit processor The 80486 processor has 32 bit ALU. hence it is called 32-bit processor.

The evolution of Microprocessor:

History shows us that the ancient Babylonians first began using the abacus in about 500BC This calculating machine eventually sparked human mind into the development of calculating machinary that uses gears and wheels.

In 1971 Intel corporation released the world's first microprocessor the Intel 4004, a 4bit microprocessor it addresses 4096 memory locations of word size 4-bit The Instruction set consist of 45 different Instructions. It is manufactured using PMOS Technology. The Intel 4004, was soon followed by a variety of microprocessors with most of the major semiconductors manufacturers producing one or more types.

* First generation Microprocessors
* Second generation Microprocessors
* Third Generation Microprocessors
* Fourth generation Microprocessors

PRINCIPAL
SIET., TUMAKURU.

What is Microcomputer? what are the components required to build the minimum microcomputer system explain with neat Diagram.

⇒ A Micro-computer system includes two principal components, hardware and Software. The Hardware is the name given to the physical devices and circuits of the computer System refers to the program written for the computer Firmware is the term given to the program stored in ROM's or in other devices that keep their stored information even when the power is turned off.

A Microcomputer is a system of one or more integrated circuit devices, you using semiconductor Technology and digital logic to implement larger computer functions on a smaller scale.

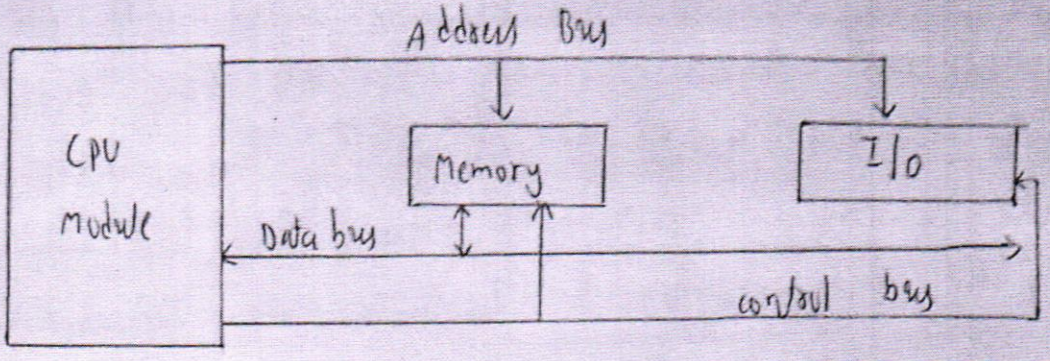
There are three main elements in a Microcomputer each has a special role to play in the overall operation of the computer System.

There are three main Elements

1. Central processing unit (CPU)
2. Memory and
3. The I/P and o/p device or ports

The CPU → The heart of the microcomputer system is the CPU. It performs the Numerical processing logical operations and timing functions.

PRINCIPAL
SIET., TUMAKURU.



The CPU operations are controlled by a set of Instructions called a program.

Programs are stored in the memory. Data is also kept in memory and processed according to programmed Instructions. The CPU reads in data and control signals through i/p ports, executes one Instruction at a time, and sends data and control signals to the outside world through the o/p ports. A Typical CPU consists of the following three units

i) Registers
ii) ALU
iii) The control unit

With a neat block Diagram explain the Architecture of 8086

⇒ The 8086 Architecture can be broadly divided into two groups

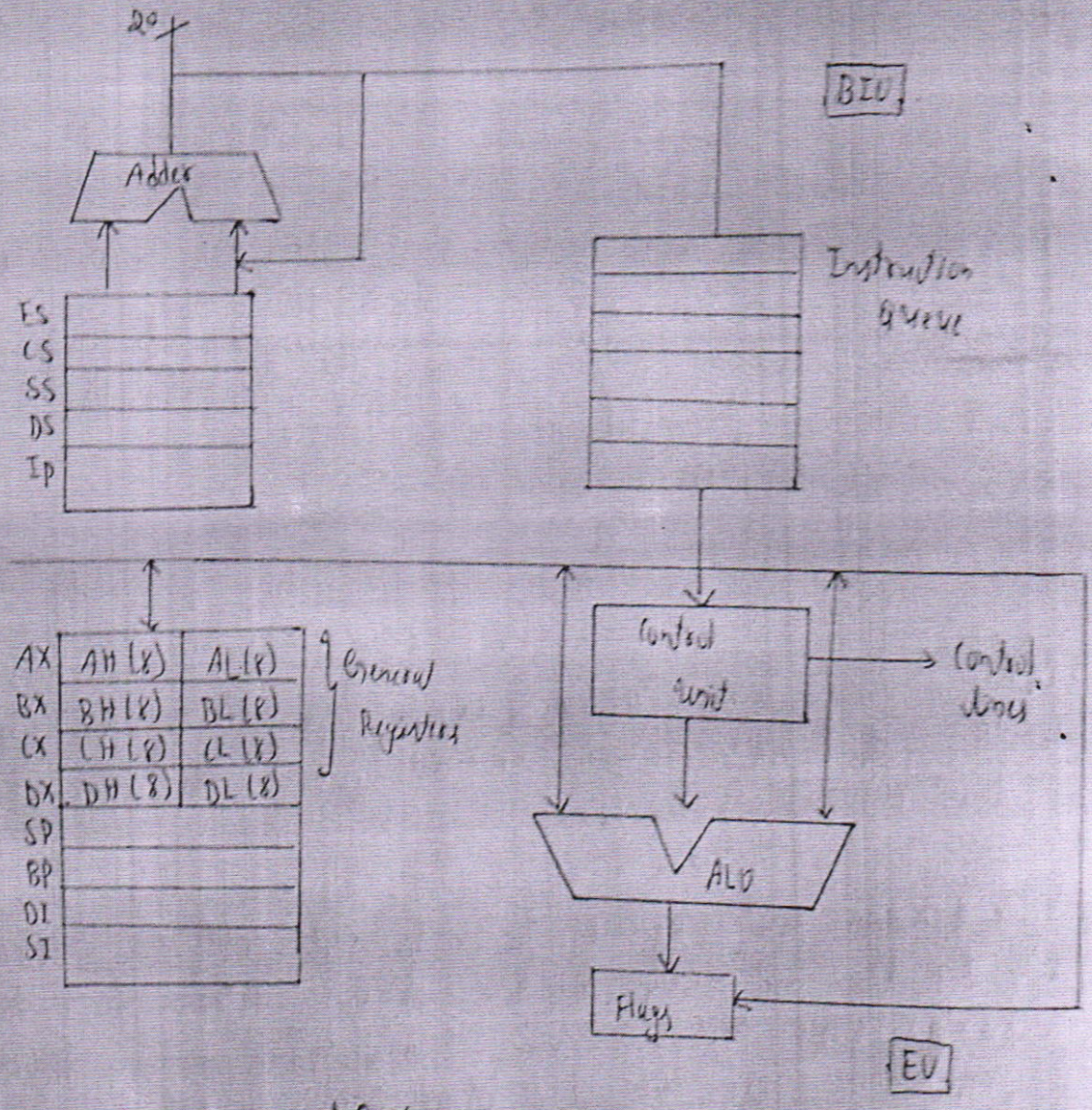
i) Execution unit [EU]
ii) Bus Interface unit [BIU]

The Execution unit contains the data and address Registers, the arithmatic and logic unit and the control unit. The Bus interface unit contains segment registers, memory addressing logic and a six byte Instruction object code queue

The Execution unit and the BIU operate asynchronous The EU waits for the instruction object code to be fetched from the memory by the BIU

PRINCIPAL
SIET., TUMAKURU.

The BIU fetches or prefetches the object code (16-bits at a time) and loads it into the six byte queue whenever the EU is ready to execute a new instruction, it fetches the instruction object code from the front of the instruction queue and executes the instruction in specified number of clock periods



* 8086 Architecture *

PRINCIPAL
SIET., TUMAKURU.

Explain the structure of flag registers explaining each bits

→ The Execution unit has a 16-bit flag Register which indicates some conditions affected by the execution of an Instruction some bits of the flag register control certain Operations of the EU the flag Register in the EU contains nine active flags as shown in figure

D_{15}	D_{14}	D_{13}	D_{12}	D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
U	U	U	U	OF	DF	IF	TF	SF	ZF	U	AF	U	PF	U	CF

Flag Register

Six of the Nine flags are used to indicate some condition produced by an Instruction. These condition flags are also called status flags of 8086/8088 microprocessor. These are the carry flag, parity flag, Auxillary carry flag, zero flag and sign flag The other three control flags are Trap flag Direction flag and Interrupt flag.

* Condition flags: Carry_Flag (CF) :→ This flag will be set to one if the addition of two 16-bit binary numbers produces a carryout of the most significant bit position or if there is a borrow to the MSB after subtraction. This flag is also affected when other Arithmetic Logical Instruction are executed

Parity_Flag (PF) :- This flag is set, if the result of the operation has an even number of 1's in the lower 8 bits of the Result.

Auxillary_Carry_Flag (AF) :- This flag is set, when there is a carry out of the lower nibble to the higher nibble or a borrow from the higher nibble to the lower The Auxillary carry flag is used for decimal adjust

PRINCIPAL
SIET., TUMAKURU.

operation. The AF flag is of significance only for byte operations during which the lower order byte of the 16 bit word is used.

Zero Flag (ZF):- This flag is set when the result of an operation is zero. The flag is reset when the result is nonzero

Sign Flag (SF):- The sign flag holds the Arithmetic sign of the result after an arithmetic or logic instruction executes if S=1, the result is negative and if S=0 the result of the arithmetic operation is +ve

Overflow Flag (OF):- This flag is set, when an Arithmetic overflow occurs. overflow means that the size of the result exceeded the storage capacity of the destination and a significant has been lost

* Control flags:- Trap Flag (TF): This is used for single stepping through a program. It is used for debugging the programs

Interrupt Flag (IF): It is used to allow/prohibit the Interruption of a program when the flag is set, it enables the Interrupt from INTR when the flag is reset (0), it disables the Interrupt.

Direction flag (DF): It is used for string Instruction If the direction flag is set, the Index registers are decremented else the Index Registers are Incremented

5. What is addressing mode explain the various Addressing mode with examples.

=> Addressing Mode:

PRINCIPAL
SIET., TUMAKURU.

An instruction is divided into group of bits called fields one field, called the operation code (or opcode), indicates what the operation code computer is to do, and the other fields, called the operands, indicate the information needed by the instruction is carrying out its task. Instruction format is shown in fig

opcode	operand-1	operand-2

general Instruction format

An Operand may contain a datum, at least a part of the address of a datum, an indirect pointer to a datum, or other information pertaining to the data to be acted on by the instruction. "The way in which an operand is specified in an instruction is called as Addressing Mode"

The 8086/8088 provides many different ways of addressing operands operands may be contained in registers, within the instruction itself, in memory or in I/O ports. In addition, the addresses of memory and I/O port operands can be calculated in several different ways. These Addressing modes greatly extend the flexibility and convenience of the Instruction set. The various 8086/8088 Addressing modes are:

Addressing modes

1) Register A.M
2) Immediate A.M
3) Relative A.M
4) Implied A.M

Memory Reference modes

5) Direct AM
6) Register Indirect A.M
7) Base A.M
8) Indexed A.M
9) Based Indexed A.M
10) String A.M

I/O Reference Mode

11) Direct I/O A.M
12) Indirect I/O A.M

PRINCIPAL
SIET., TUMAKURU.

* Registers Addressing Mode:

In this A.M, the operands are available in the Register itself. This A.M is generally the most compact and fastest executing. This is because the Registers addresses are encoded in instructions in just a few bits and because these operands are stored entirely within the CPU Registers. Any way may serve as source operands, destination operands or both. The Registers may be AX, BX, CX, DX, SI, DI, SP or BP and for an 8-bit operand the Registers may be AL, AH, BL, BH, CL, CH, DL, or DH

Ex: Inc Reg or Inc AX
Add Reg1, Reg2 Add AX, BX

* Immediate Addressing mode:

Immediate operands are constant data contained in an instruction. The data may be either 8-bits or 16-bits long. Immediate operands can be accessed quickly because they are available directly from the Instruction queue. The limitations of Immediate operands are that they may only serve as source operands and they are constant values

Ex: ADD Reg, Data
mov cl, 04H
Add CX, 1234H.

* Relative Addressing Mode:-

Instructions using this mode of addressing specify the operand as a signed 8-bit displacement relative to Instruction pointer (IP).

Ex: JNC Start ; if CF=0, then IP is loaded with current IP + 8-bit signed value of start; otherwise the next Instruction

PRINCIPAL
SIET., TUMAKURU.

is executed

* Implied Addressing mode: Instructions using this mode have no operands The operands used to perform the operation depends on the particular Instruction.

Ex: CLC ; Clear carry flag.

What is an instruction set and explain the different Types of Instructions.

⇒ Instruction set of 8086:

The 8086/8088 has approximately 117 different Instructions with about 300 opcodes. The 8086/8088 instruction set contains zero operand, single operand and two operand Instructions except for string Instructions that involve array operations. The 8086/8088 instructions do not permit memory to-memory operations.

The Instructions are divided into the following functional groups

1) Data transfer
2) Arithmetic
3) Bit manipulation
4) String manipulation
5) control transfer
7) Process control

* Data transfer Instructions:

Data transfer instructions move single byte, word and Double word b/w memory and Registers as well as b/w registers and b/w AL or Ax and I/O ports. Stack manipulation instructions are included in this group as well as instructions for transfering flag contents of for loading segment registers. The data transfer group instructions can be sub grouped as follows

a) General purpose Data transfer
b) Input/output
c) Address object
d) Flag transfer

PRINCIPAL
SIET., TUMAKURU.

7. With examples explain the following instruction

i) MOV ES, CX

⇒ Register to Segment Register

MOV ES,CX ; Move the contents of CX Register to ES

Before execution: ES - [2000H] - 6556H

CX - 1000H

ex: MOV, ES, CX

After execution: CX - 1000H

ES - [2000H] - 1000H

ii) PUSH Destination (PUSH CS)

⇒ POP transfers the word at the current top of stack (pointed by SP) to the destination operand and then increments SP by 2 to point to the new top of the stack. POP can be used to registers or memory. No flags are affected

Destination ← (Top of Stack)

iii) XCHG AX, BX

⇒ XCHG [Exchange] Switches the contents of source and destination operands. None of the flags are affected

Dest ↔ Source

XCHG AX, BX ; Exchange the contents of Ax and Bx

iv) XLAT

⇒ XLAT (Translate Table) replaces a byte in the AL register with a byte from a 256 byte, user-coded translate Table. Register BX is assumed to point to the beginning of the table. It is replaced by the byte of the offset in the table corresponding to AL's binary value. The 1st byte in the table has an offset Zero.

PRINCIPAL
SIET., TUMAKURU.

Memory Address

30	34000
31	34001
32	34002
33	34003
34	34004
35	34005
36	34006
37	34007

Before Execution

AL = 05H
BX = 4000H
DS = 3000H

AE:

AL = 35H
BX = 4000H
DS = 3000H

The 6th element of the translate table contains 35H, then AL will contain 35H.

AL = [DS * 16 + BX + AL].

---

v) LEA BX, [BX + DI + 1000H]

=> LEA (Load effective Address) transfers the offset of the source operand to the destination operand. The source operand must be a memory operand and the destination operand must be a 16 bit general registers LEA does not affect any flags

Dest ← offset Address

LEA BX, [BX + DI + 1000H]

BX = 400H

DI = 003CH

16-bit Disp = 1000H

BE: BX = 0400H , DI = 003CH

AE: BX = 0923H , DI = 003CH

---

vi) LDS SI, [10H]

=> LDS Loads 32-bit or two words from memory into specified destination and DS Register. Here the source must be a memory operand. A word is taken from memory location whose address is specified as source and loads it into specified destination Register. Next word is taken from memory location whose address is calculated by adding two to source address and is loaded into DS Register.

Dest ← Address of source

DS ← Address of source + 2

Ex:- LDS SI, [1011]

BE : DS = 0000H

SI = 3000H

M[00010] = 0180H

M[00012] = 2000H

AE : DS = 2000H

SI = 0180H

M[00010] = 0180H

M[00012] = 2000H

viii) ADC CX, DX

⇒ ADC (Add with carry) Sums the operands, which may be bytes or words, adds one if CF is set and replaces the destination operand with the result. Both operands may be signed or unsigned binary numbers. ADC updates AF, CF, OF, PF, SF and ZF. Since ADC incorporates a carry from a previous operation, it can be used to write routines to add numbers longer than 16-bits

Ex:- ADC CX, DX ; CX = CX + DX + CF

BE : CX = 1000H, DX = 2000H DF = 0 CF = 0

AF : CX = 3000H, DX = 2000H PF = 1 CF = 1

viii) DAA

⇒ DAA - Decimal Adjust for Addition

corrects the result of previously added two valid packed decimal operands, (the destination operand must be Register AL). DAA changes the contents of AL to a pair of valid packed digits it updates AF, CF, PF, SF and ZF; then..

ix) CMP CX, BX

⇒

BE CX = 1001H BX = 1000H CX-BX ZF = 1 PF = 1

AE CX = 1001H BX = 1000H

PRINCIPAL
SIET., TUMAKURU.

x) DAS

⇒ DAS [Decimal adjust for Subtraction] corrects the results of a previous subtraction of two valid packed decimal operands (the destination operand must have been specified as Register AL)

xi) IMUL

⇒ IMUL (Integer Multiply) performs a signed multiplicate of the source operand and the Accumulator if the source is a byte, then AL is multiplied and the double byte result is returned in Register AH and AL. If the source is a word then it is multiplied by Register AX and the double word result is returned to registers DX and AX i.e the upper half of the Result is not sign extended form of the lower half of the result, CF and OF are set, otherwise they are cleared.

xii) IDIV

⇒ This Instruction is used to divide a signed word by a signed byte or to divide a signed double word by signed word.

xiii) call Delay

⇒ A subroutine or a procedure is a set of code that can be branched to and returned from in such a way that the code is as if it were inserted at the point from which it is branched to. The branch to a procedure is released to as the 'call' and the corresponding branch back is known as 'Return'.

xiv) JBE / JNA

⇒ Jump on Below or equal / Jump on not above transfers control to the target Instruction if CF or ZF = 1

Assignment - 02

2. Explain the following instructions

i) CMPS

⇒ CMPS → (compare string) Subtracts the destination byte or word (addressed by DI) from the source byte or word (addressed by SI). CMPS affects the flags but does not alter either operand updates SI and DI to point to the next string element and updates DF, SF, ZF, CF and AF

[DS:SI] - [ES:DI]

ii) SCAS

⇒ Scan string subtracts the destination string element (byte or word) addressed by DI from the contents of AL or AX and updates the flags, but does not alter the destination string or the Accumulator.

AL - [ES:DI]

iii) REPNZ

⇒ REPNZ [Repeat while not zero]

REP is used as a prefix for MOVS and STOS instructions and is interpreted as "repeat while not end of string" (CX ≠ 0)

REPNZ and REPNE operate mnemonics for the same prefix that Instructions functions as same as REPE and REPZ except that the zero flag must be cleared or the repetition is terminated

iv) INTO

⇒ Interrupt on overflow generates a slw Interrupt if the overflow flag (OF) is set, Other wise control proceeds to the

PRINCIPAL
SIET., TUMAKURU.

following Instruction without activating an Interrupt procedure. INTO calls the target Interrupt procedure (type 4) through the Interrupt pointer at location 10H if Users the TF and IF INTO may be written following as Arithmetic or logical operation to activate an Interrupt procedure of overflow occurs

v) HLT

⇒ HLT (Halt) causes the processor to enter the halt stage when RESET is activated processor comes out of halt stage.

* upon the receipt of an maskable Interrupt request or NMI
* or if interrupts are enabled, upon receipt of maskable Interrupt request on INTR
* It may be used as an alternative to an endless s/w loop in situations where a program must wait for a Interrupt.

vi) CLD

⇒ CLD (Clear Direction flag) → DF=0

vii) Lock

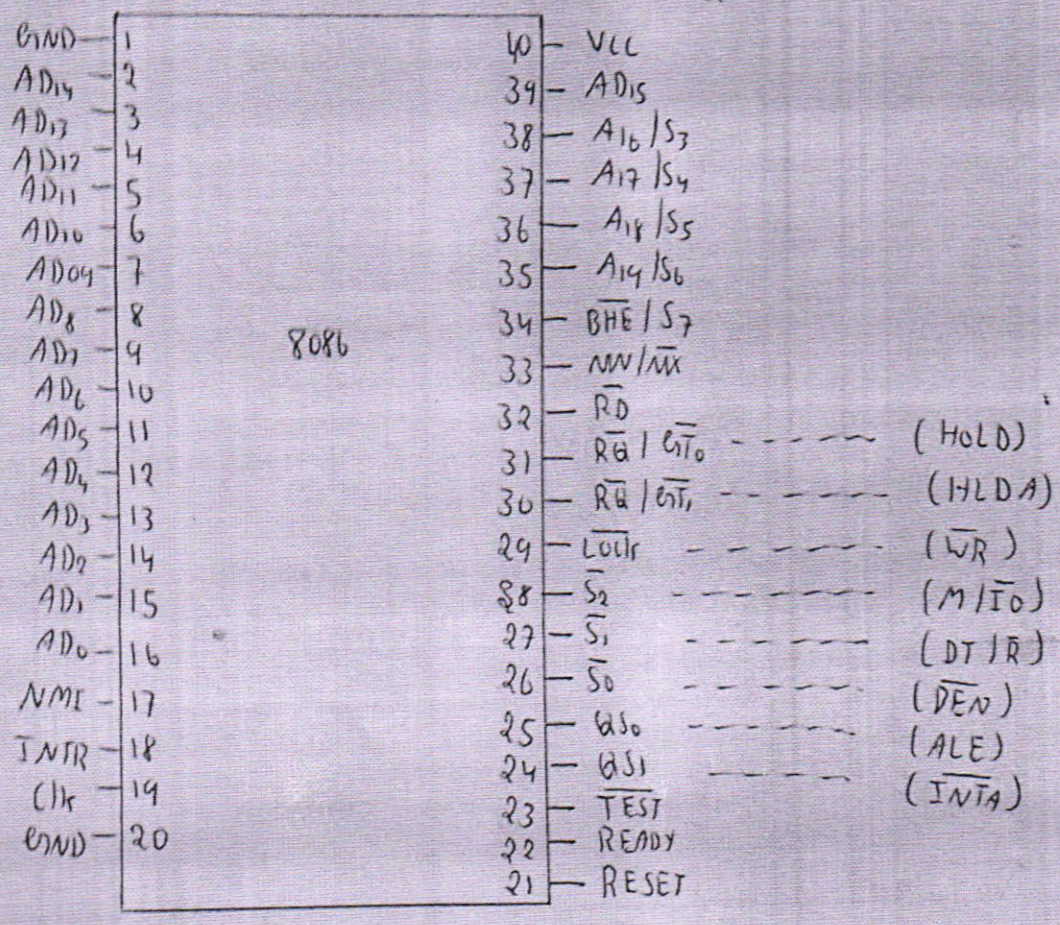
⇒ Lock prefix causes the processor to assert the bus lock signals. Lock doesn't affect any flag. This Instruction is mostly used with exchange registers with memory Instruction. This Instruction is used in Multiprocessor environment to tell all the other processors that they all disabled from using 8086's System Bus.

2. With a neat diagram explain the pin diagram of 8086

⇒ The Microprocessor 8086 is a 16 bit CPU available in three clock rates i.e 5, 8 and 10MHz, packaged in a 40 pin CERDIP or plastic package. The 8086 operates in single

PRINCIPAL
SIET., TUMAKURU.

processor or multiprocessor configurations to achieve high performance. Some of the pins have a particular function in minimum mode and others function in maximum mode.



Pin configuration of 8086

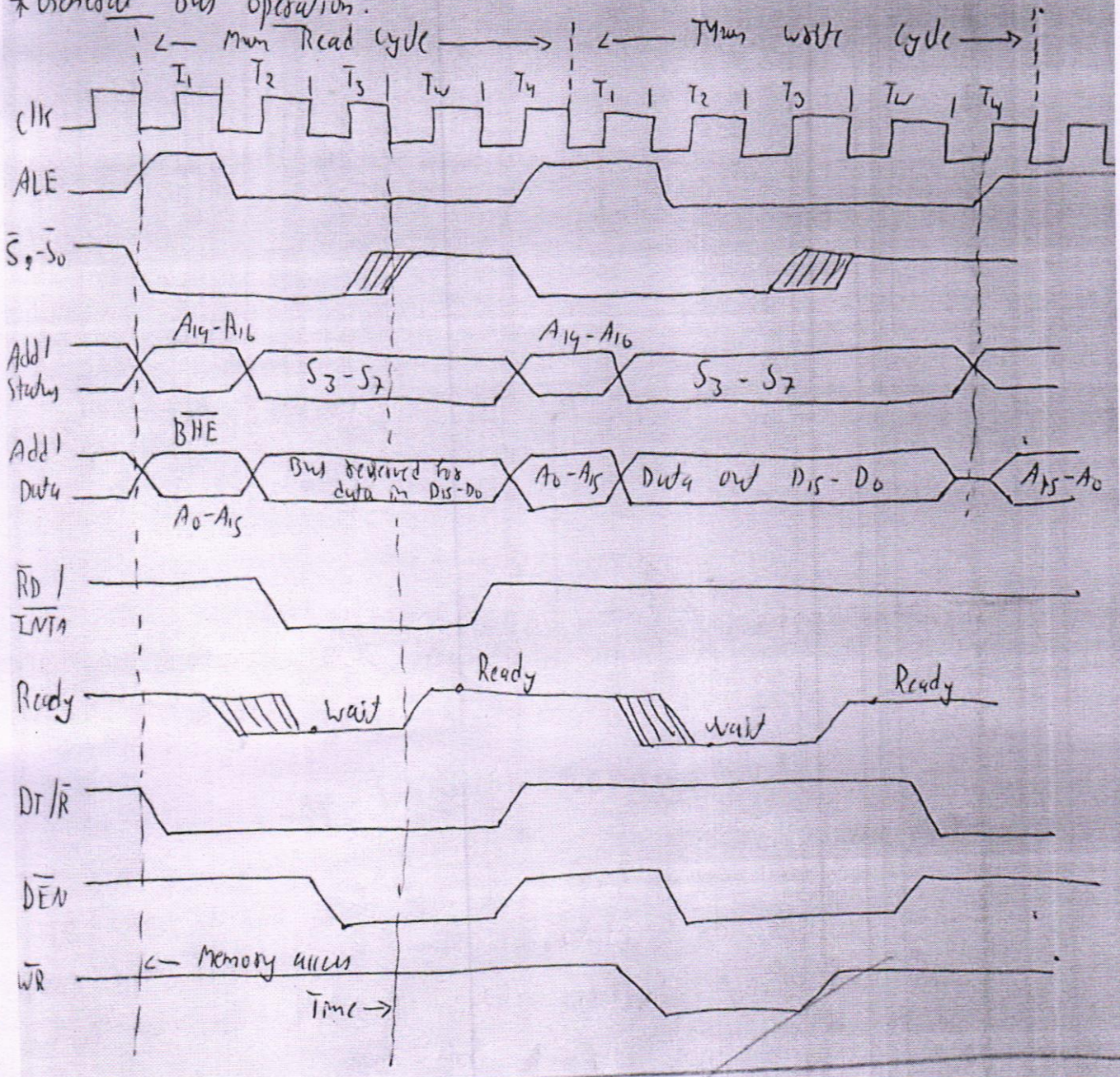
Write the general bus operation of 8086

=> All the processor bus cycles consist of at least four clock cycles. They are referred as T_1, T_2, T_3 and T_4. The Address is transmitted by the processor during T_1. It is present on the bus only for one cycle. During T_2 the next cycle, the bus is isolated for changing the direction of bus for the following data read cycle. The data transfer takes place T_3 and T_4.

PRINCIPAL
SIET., TUMAKURU.

In case, an addressed device is slow and thows 'NOT READY' status, wait states (T_w) are inserted between T_3 and T_4.

* General Bus operation:



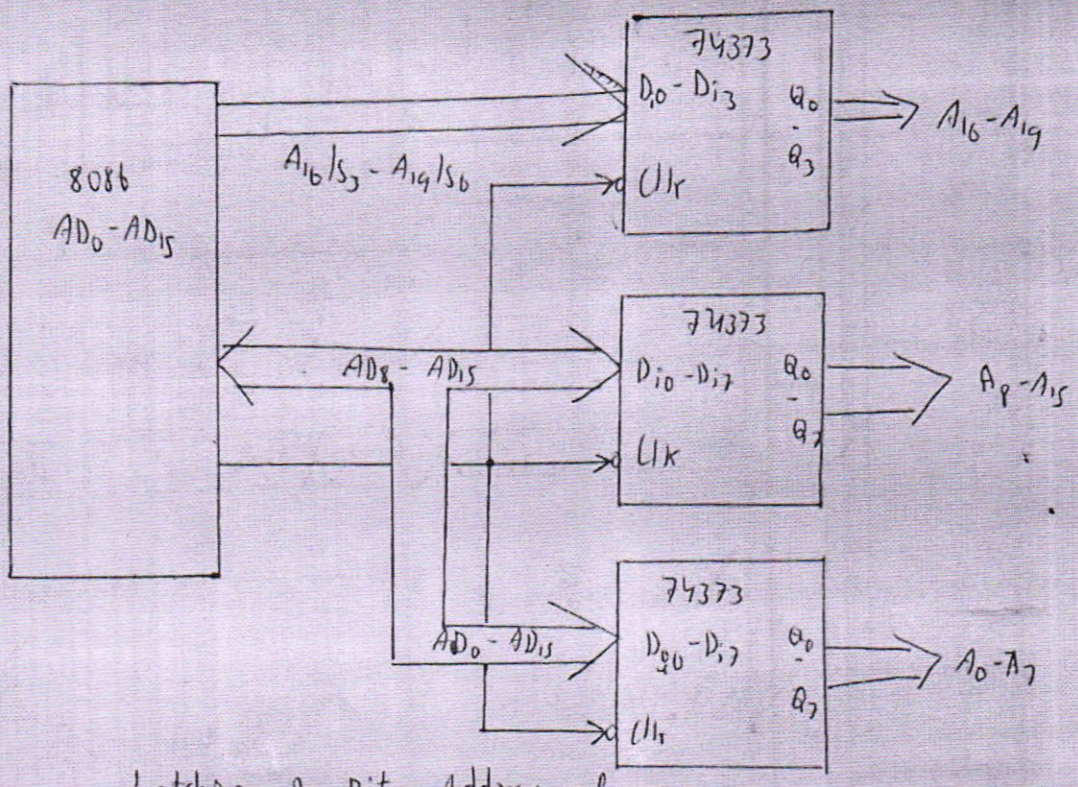
How the address bus and data Bus are derived from system bus

⇒ Deriving System Bus:

Address Bus: The 8086 has a multiplexed 16-bit address/data bus (AD_0 - AD_{15}) and a multiplexed 4 bit address/status bus (A_{16}/S_3 - A_{19}/S_6). The Address can be latched using ALE. For de-mu

PRINCIPAL
SIET., TUMAKURU.

- Multiplexing 20 address lines it requires three latches. Here 74373 out of three latches one uses only for flipflops and the other two uses all eight D_i indicates data i/ps. to latches and Q_i the o/ps.

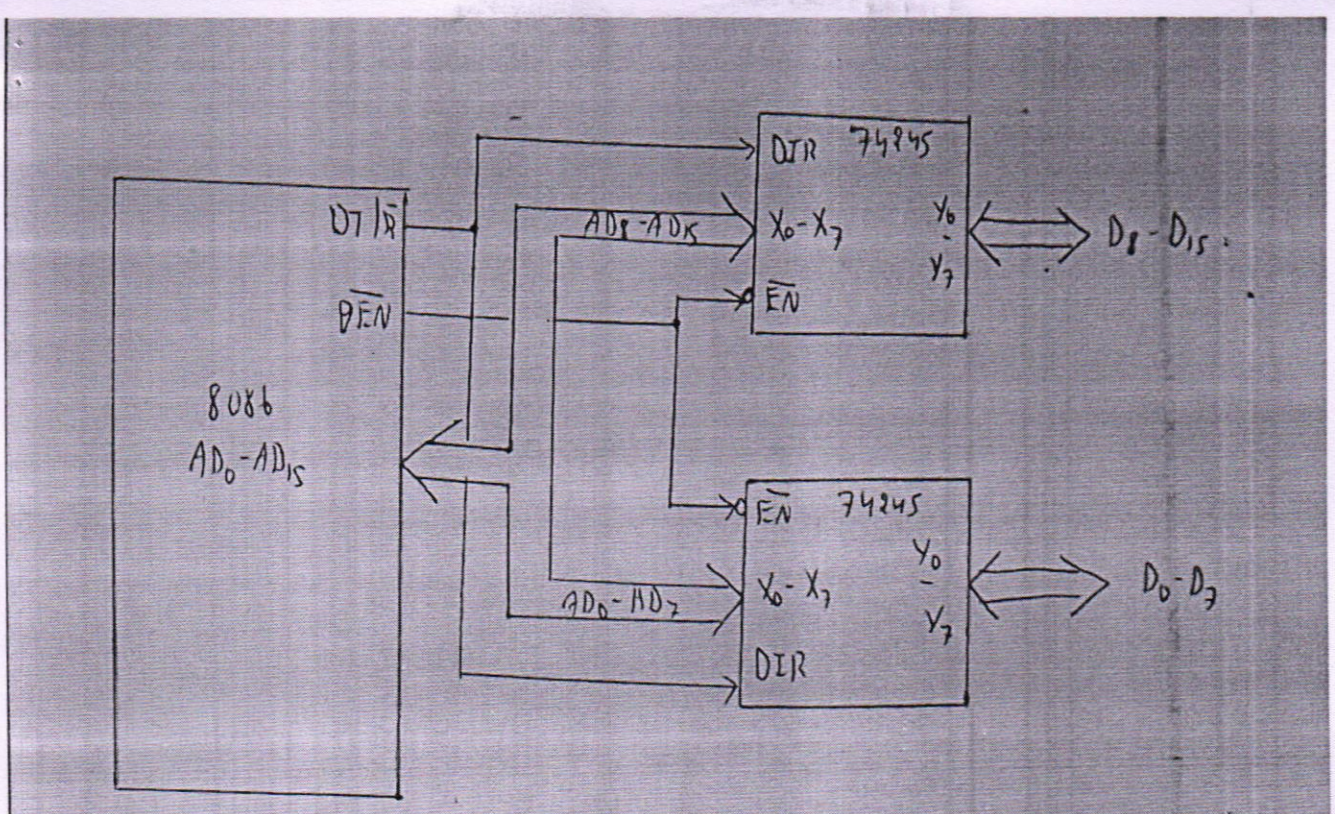


Latching 20-Bit Address of 8086

Data Bus:

8086 has multiplexed 16 bit data bus in the form of AD_0-AD_{15} The data can be separated from the address and buffered using two bidirectional buffers 74245. The data can be either transfrom from MP to memory or from memory to MP in case of write and read operations. hence bidirectional.

PRINCIPAL
SIET., TUMAKURU.



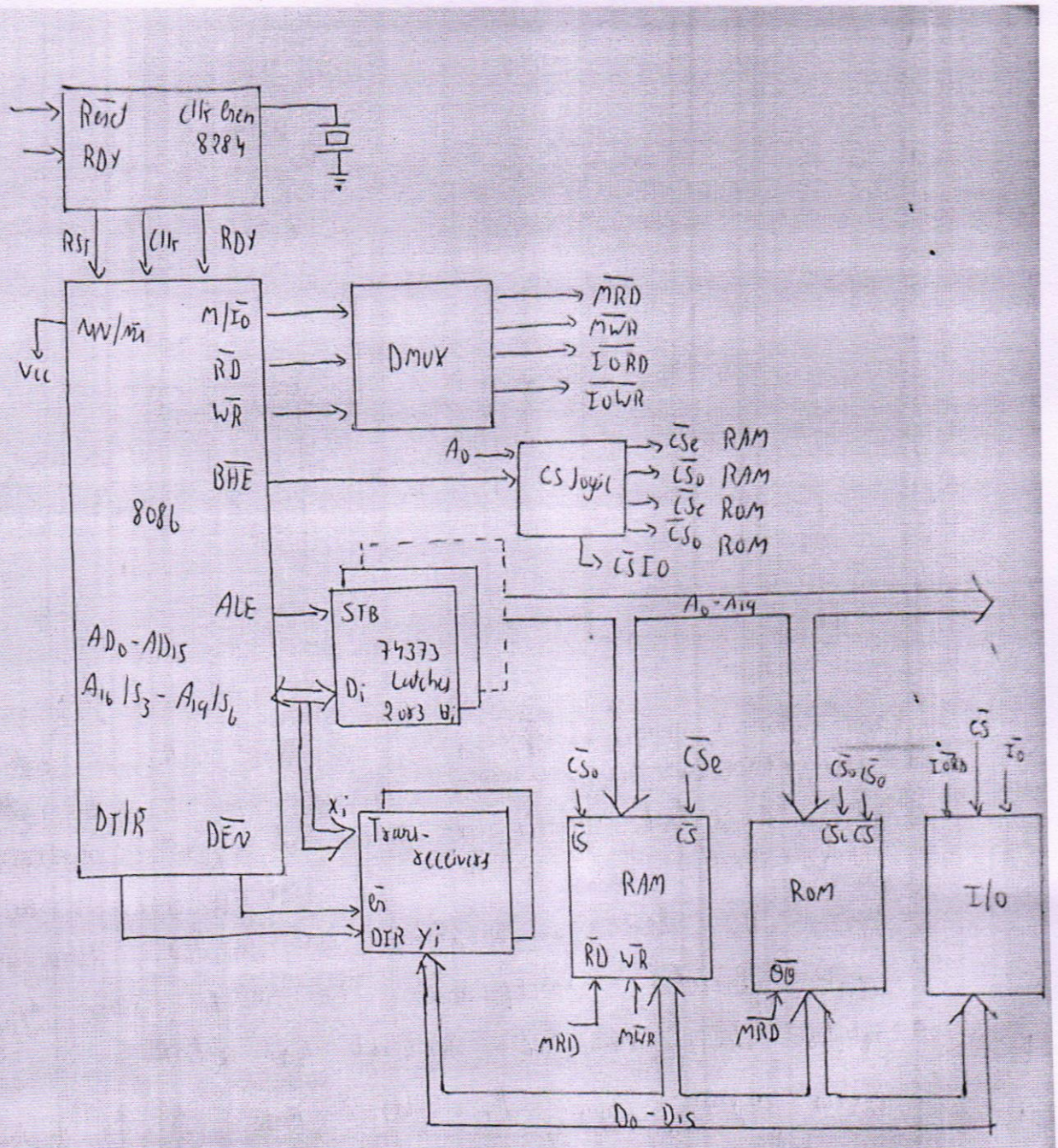
Explain the minimum mode working of 8086 with a neat diagram

=> In Minimum mode MN/$\overline{MX}$ pin is logic 1. In this mode all the control signals are given out by the microprocessor chip itself. There is a single microprocessor in minimum mode system. The remaining components are latches, transceivers, clock generator, memory and I/O devices. Some type of chip selection logic may be required for selecting memory or I/O devices depending on the address map of the system.

The latches are buffered output D-type flip flops like 74LS-373 or 8282. They are used for separating the valid address from the multiplexed address/data signals and are control-ed by the ALE signal by 8086

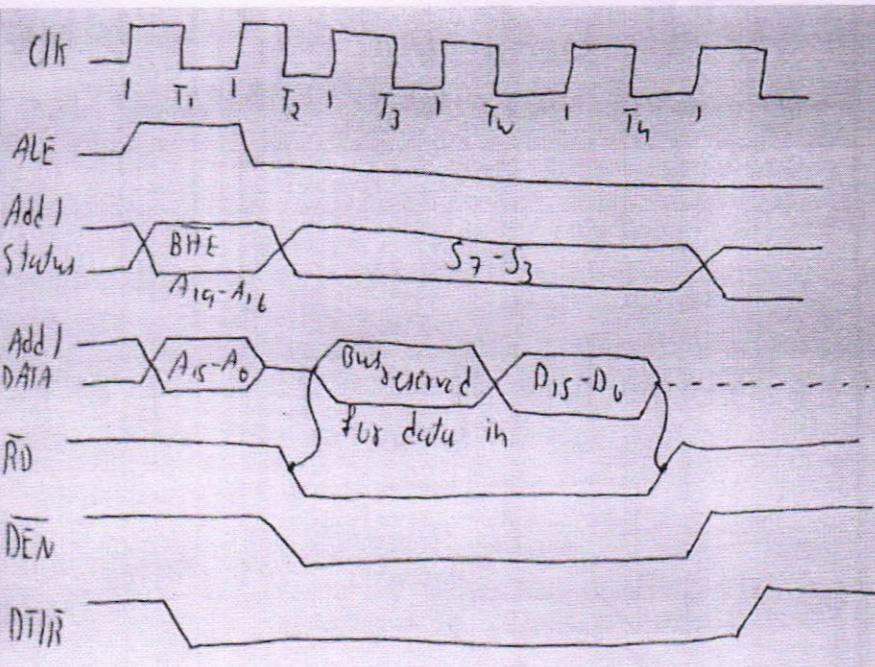
M/$\overline{IO}$	$\overline{RD}$	$\overline{DEN}$	Transfer Type
0	0	1	I/O Read
0	1	0	I/O-write
1	0	0	Memory Read
1	1	0	Memory write

PRINCIPAL
SIET., TUMAKURU.

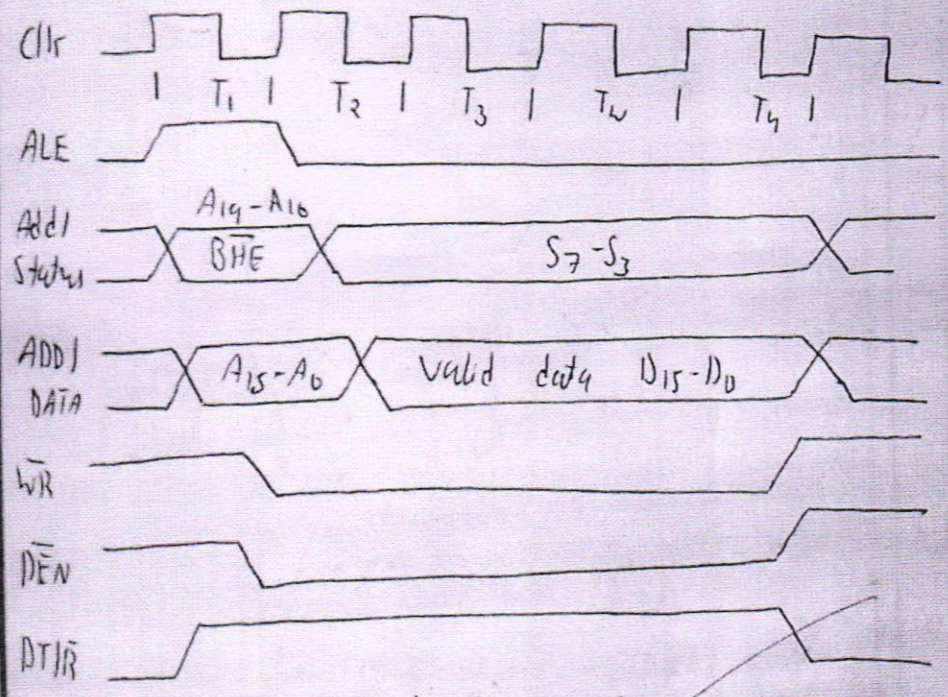


6. With a neat timing diagram explain the read cycle and write cycle in minimum mode of 8086

=> The Read cycle begins in T_1 with ALE signal and M/IO signal During negative Going edge of this signal the valid address is latched on the local bus. The BHE and A_0 signals address low, high or both bytes. From T_1 to T_4, the M/IO signal indicates a memory or I/O operation.



Read cycle Timing Diagram for minimum mode



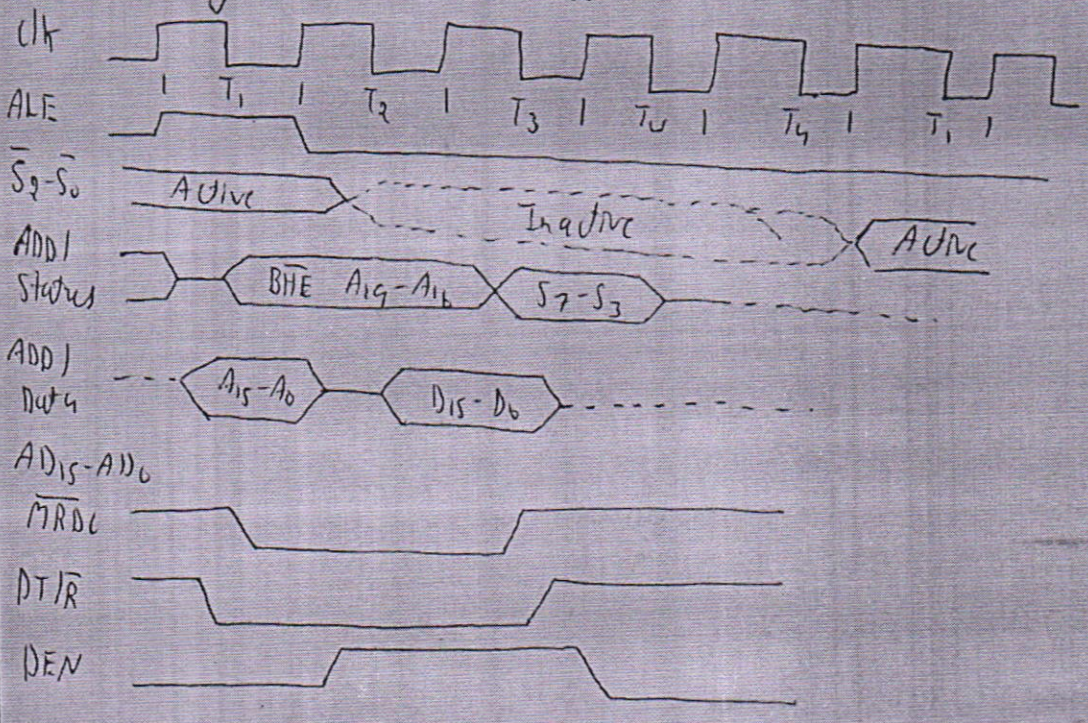
Write cycle Timing diagram for Minimum mode

7. With a neat timing diagram explain the read cycle and write cycle in maximum mode of 8086

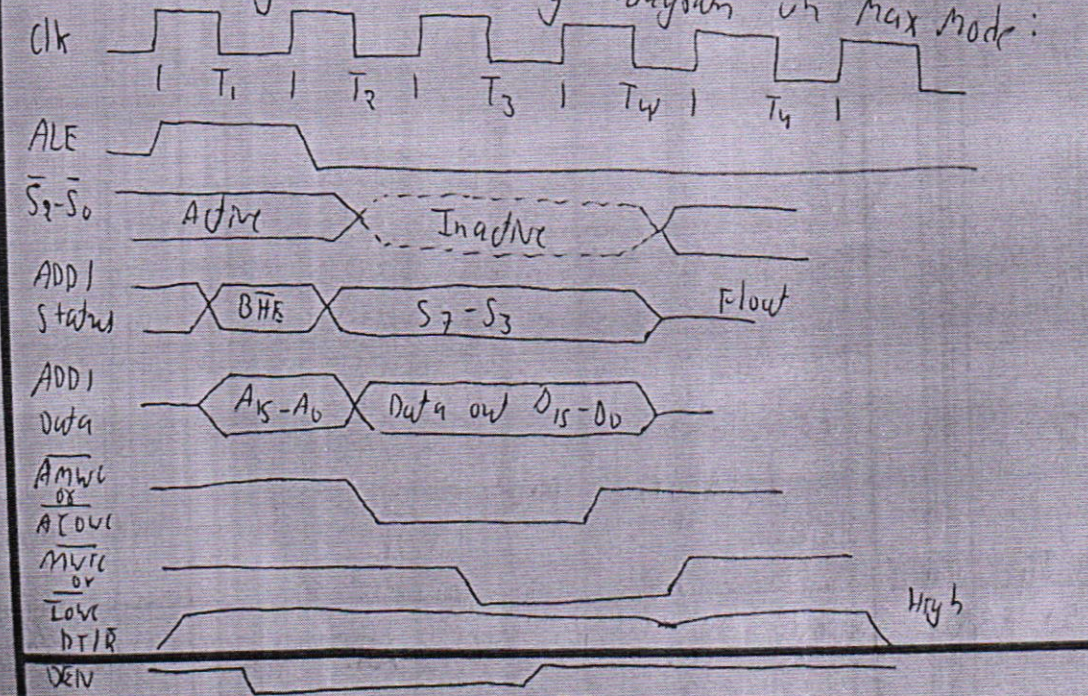
=> In maximum mode 8086 is operated by strapping the MN/MX Pin to ground. In this mode processor derives the status

Signals $\overline{S_2}$, $\overline{S_1}$ and $\overline{S_0}$ Another chip called bus controller decodes the control signals using this status Information In the maximum mode there may be more than one microprocessor in the system configuration.

* Read Timing in Maximum mode:



* Write memory write Timing diagram in max mode:

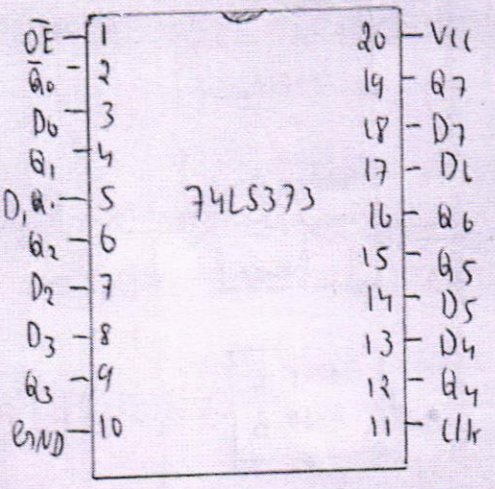


PRINCIPAL
SIET., TUMAKURU.

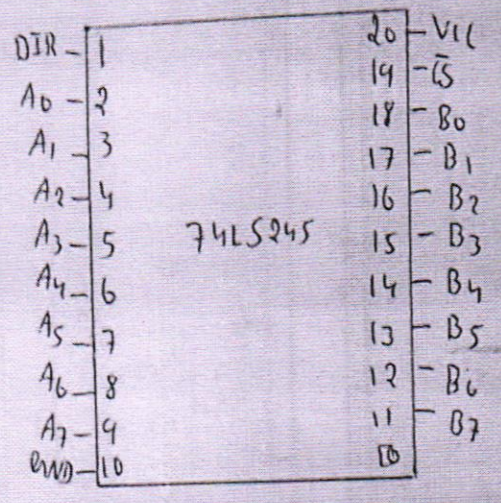
8. Explain the pin diagram of 74LS373 and 74LS245

⇒ Input /output ports are the devices through which the microprocessor communicates with other devices or external data sources / destinations

$\overline{IORD}$ operation is related with reading data from an input device and not an output device and not an output device and $\overline{IOWR}$ operation is related with writing data to an o/p device. The control word and status word may be written



Latch o/p port



Buffer (I/O port)

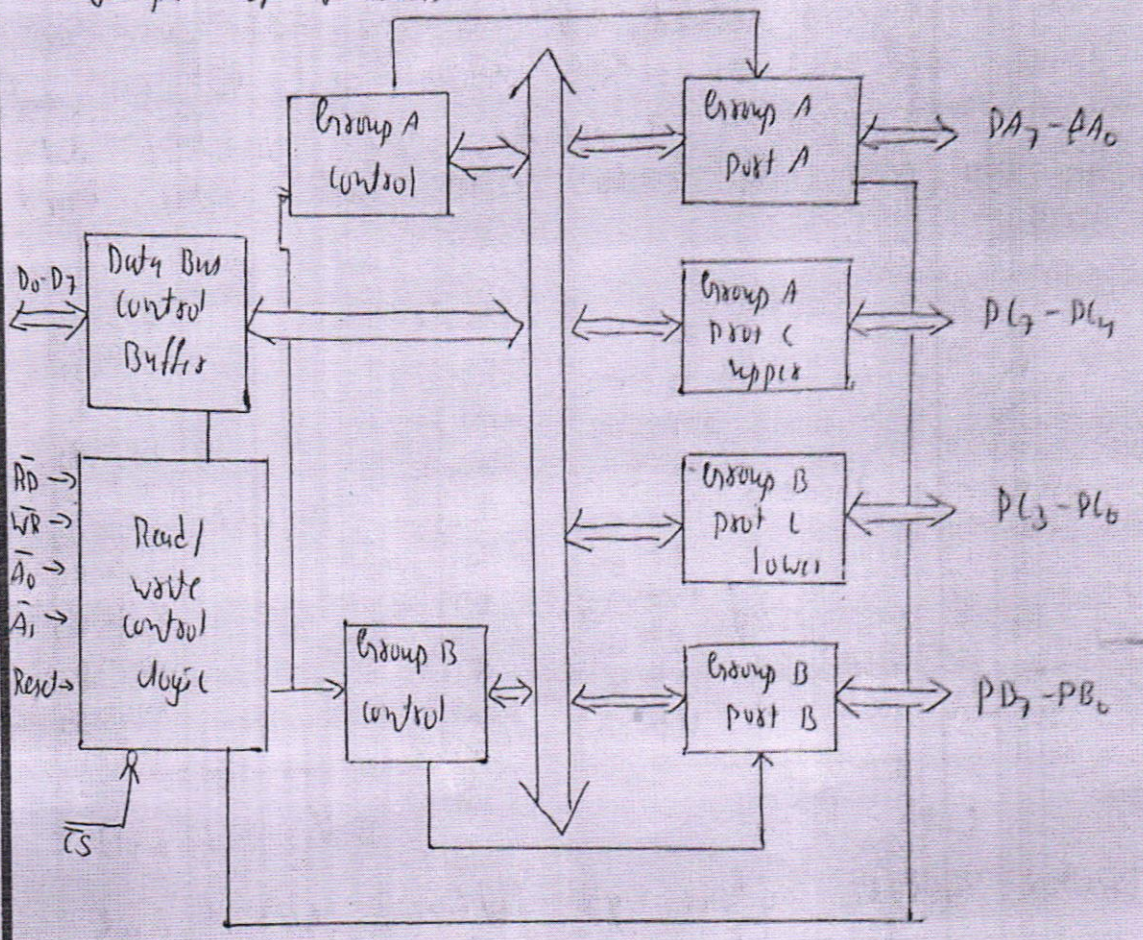
The chip 74LS373, contains eight buffered latches and can be used as an 8 bit o/p port. The chip 74LS245 contains eight buffers and may be used as an 8-bit i/p port. The DIR pin is used for selecting the direction of data flow. The $\overline{OE}$ and $\overline{CS}$ are the chip selects of 74LS373 and 74LS245 Ds and Qs are the latch I/p's and o/p's

9. Explain with a neat block diagram 8255 PIO

⇒ PIO 8255 [Programmable Input output Port]

The parallel input-output port chip 8255 is also known as programmable peripheral I/p - o/p port. The Intel's 8255 is

designed for use with Intel's 8-bit, 16-bit and higher capability microprocessors. It has 24 I/O lines which may be individually programmed in two groups of twelve lines each, or 3 groups of 8 lines.



10. Explain the modes of operation of 8255

⇒ Modes of operation of 8255

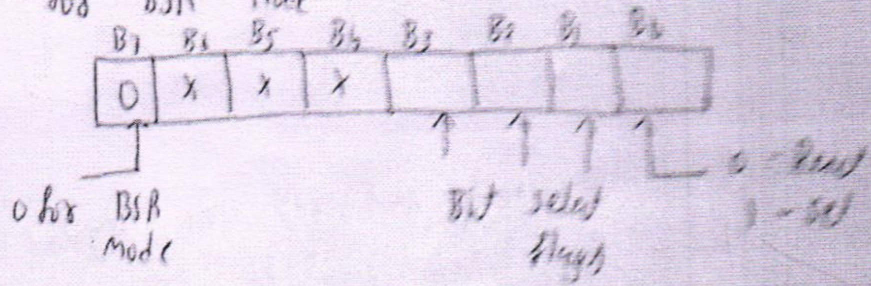
* There are two basic modes of operation of 8255 - I/O mode and Bit set - Reset mode (BSR)
* In IO mode 8255 ports work as programmable I/O ports. In BSR mode only port C ($PC_0 - PC_7$) can be used to set or reset
* Under I/O mode of operation there are three modes of operation of 8255 mode 0, mode 1 and mode 2.

BSR mode:

B_3	B_2	B_1	Selected Bits of Port C
0	0	0	B_0
0	0	1	B_1
0	1	0	B_2
0	1	1	B_3
1	0	0	B_4
1	0	1	B_5
1	1	0	B_6
1	1	1	B_7

* In this mode any of the 8-bits of port C can be set or reset depending on B_0 of the control word. The bit to be set or reset is selected by bit select flags B_3, B_2 and B_1 of the CWR

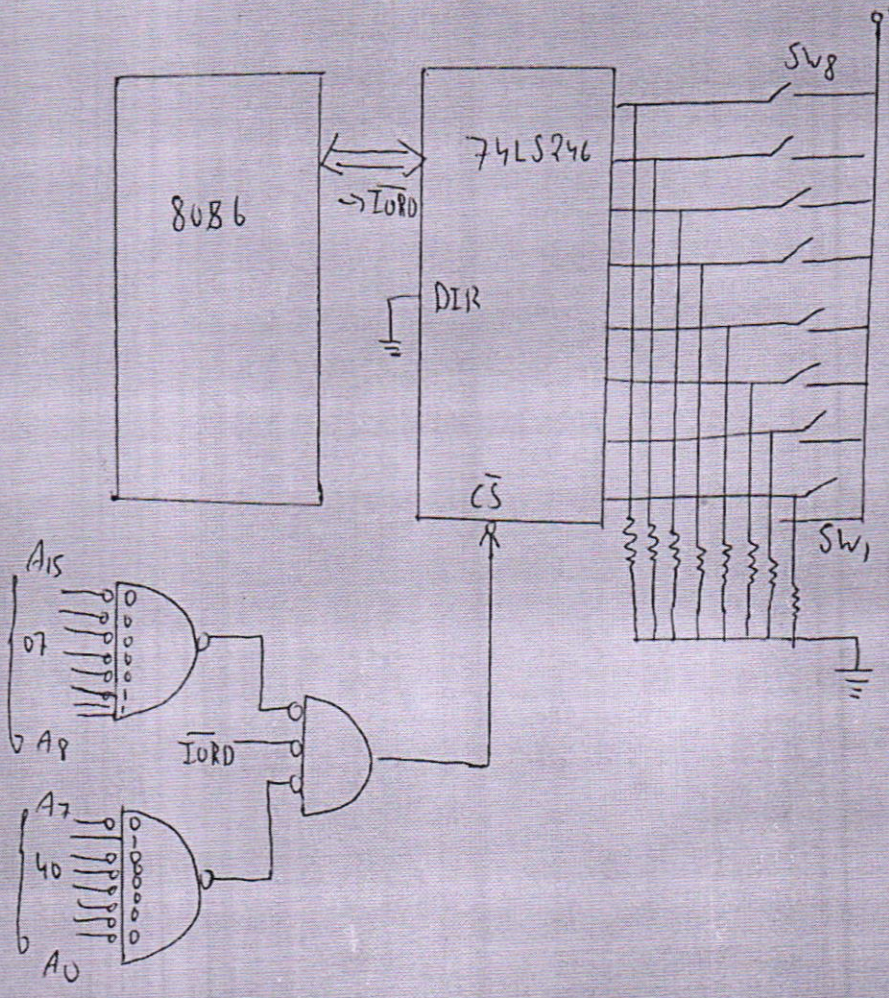
* CWR for BSR Mode



⟹ B_3, B_2, B_1 are from 000 to 111 for bits PC_0 to PC_7

11. Interface an I/O port 74LS245 to read the status of switches Sw_1 to Sw_8. The switches when shorted give the i/p as one(1) else give the i/p as zero(0) to the microprocessor system. Store the status in register BL. The address of port is [illegible]

```
=>  MOV  BL, 00H
    MOV  DX, 0740H
    IN   AL, DX
    MOV  BL, AL      HLT
```

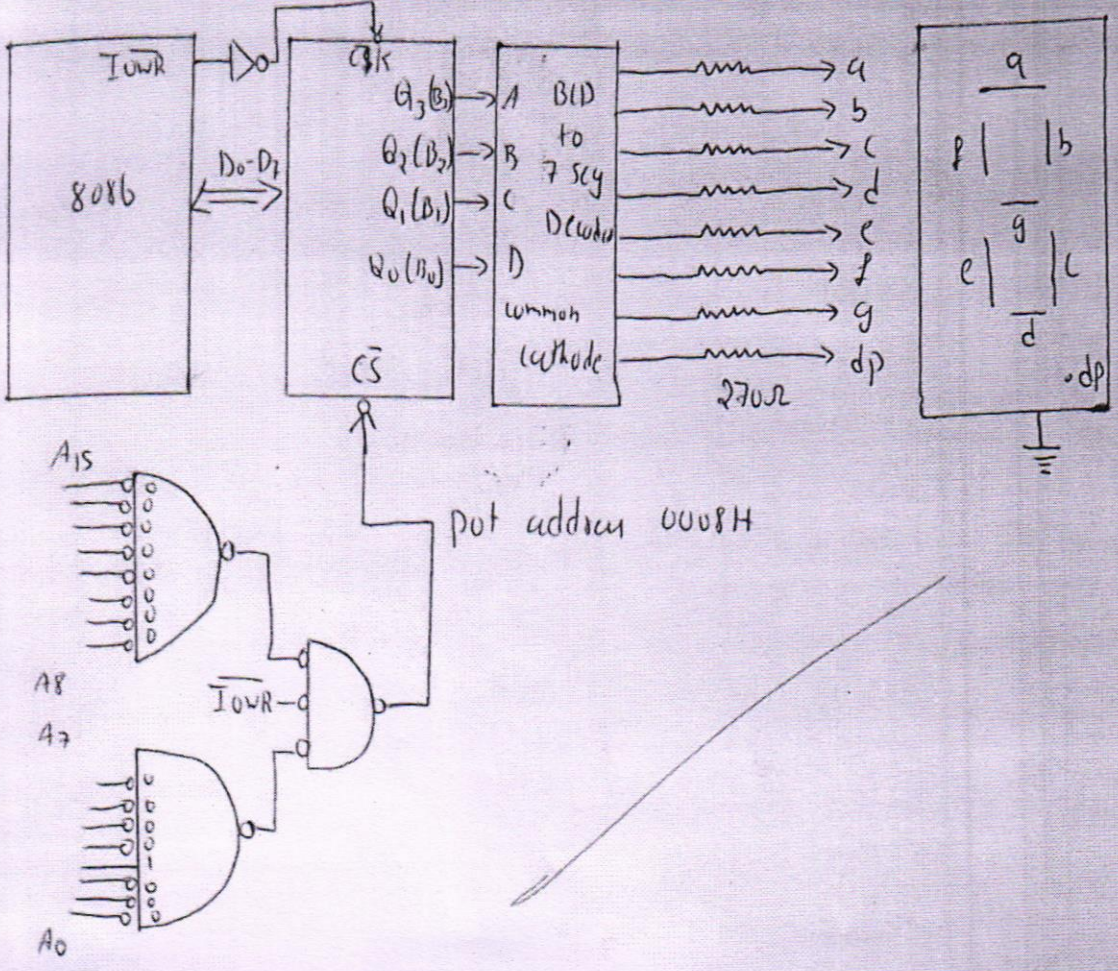


12. Using 74LS373 o/p ports and (7) Seven segment Display Design Seconds Counter that counts zero to Nine. Draw the suitable Hardware Schimmetic and write an ALP for this problem Assume that a Delay of one second is available as a subroutine. Select the port address as 0008H

=>

```
XX: MOV AL, 00H
YY: XOR AL, AL , Clear AL
    OUT 08H, AL ; Display o/p
    Call Delay
    INC AL
    CMP AL, 0AH ; compare with o/p
    JZ XX
    JMP YY
```


PRINCIPAL
SIET., TUMAKURU.



port address 0008H

PRINCIPAL
SIET., TUMAKURU.

Microprocessor

Assignment - 03

1. Write an 8086 ALP to rotate the stepper motor in clockwise direction by 360° and then in anticlockwise direction by 180° Assume 1.8° step and proper Delay.

2. Explain the following Int 21H DOS Fn calls
   i) 01H ii) 02H iii) 09H iv) 0AH

3. Differentiate b/w CISC and RISC the A Von Neumann and Harvard Architecture

4. Explain the Significance of control word register format of 8254

5. Write a program to generate triangular wave using DAC 0800

⇒

1. ALP to rotate the stepper motor in clockwise direction by 360° and then in anticlockwise direction by 180°

⇒

```
Assume CS: code segment
.model small
.code
mov AL, 33h
mov CX, 200
agun1: out port A, AL
       call delay
       ROR Delay
       ROR AL, 01
       Loop agun1
       mov AL, 33h
       Mov CX, 100
agun2: out port A, AL
       Call delay
       ROL AL, 01
       Loop agun2
       JV 03H
       end start
```


PRINCIPAL
SIET., TUMAKURU.

2. 01H * Int 21H DOS function calls

⇒ * function 01H: (character input with echo)

The function 01H takes the input character with echo or Read the character

* function 02H: (Display the character)

The function 02H Displays the character on the user screen

* function 09H: (Display the message or string)

The function 09H Display the message or string on the user screen

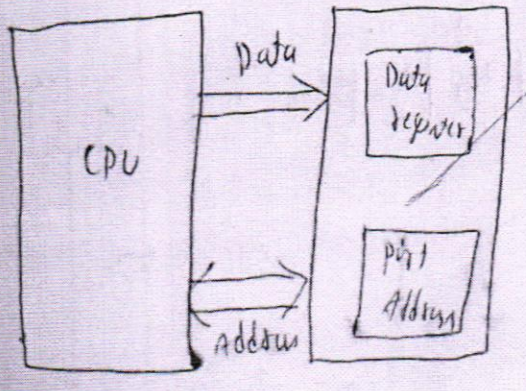
* function 0AH: (Buffered input)

The function 0AH is taken the Buffered input or Reads the Buffered input

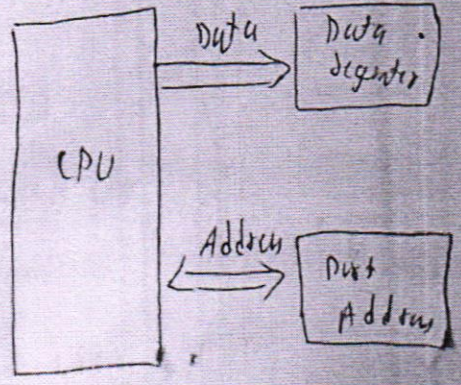
3. Von-Neumann and hasdvasd Architecture of CPU

* Von-Neumann Architecture:

The passing of the Data and Address to the CPU is taken at a single time and is called Von-Neumann Architecture.



Von-Neumann Architecture



hasdvasd Architecture

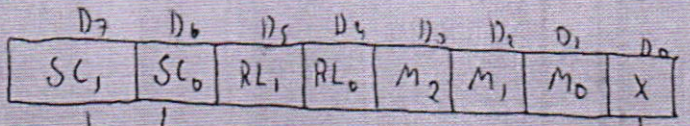
* Harvard Architecture: In the harvard Architecture the Data and Address is passed to the CPU on is different path is called harvard Architecture

=>

RISC	CISC
* Simple Address mode are used	* Complex Address mode are used
* Instruction takes one or two cycles	* Instruction takes multiple cycles
* Few Instructions	* complex Instruction set
* Most of them have multiple Register banks	* Single Register Bank
* Fixed format Instructions	* variable format Instructions
* Instruction is executed by hardware	* Instruction is executed by Micro program.

4. 8254

* control word Register:



* SC → select counter * RL → Read and load * Mode selection

The control word Register having the 8 bit of square that is named as shown and the D_0 is BCB or hexa Decimal

PRINCIPAL
SIET., TUMAKURU.

count whenever we are using BCD count we select D_0 as 1 or D_0 as 0 will count hexadecimal value.

BCD	1
Hex	0

* Mode Selection:

M_2	M_1	M_0	operation
0	0	0	mode 0
0	0	1	mode 1
X	1	0	mode 2
X	1	1	mode 3
1	0	0	mode 4
1	0	1	mode 5

* Read and load the bytes:-

RL_1	RL_0	operation
0	0	latch counter
0	1	Read and load LSB ~~first~~ only
1	0	Read and load MSB ~~first~~ only
1	1	Read and load LSB first and then MSB

Where * LSB → Least significant byte
* MSB → Most significant byte

* Select counter:

SL_1	SL_0	Operation
0	0	counter 0
0	1	counter 1
1	0	counter 2
1	1	illegal

5.

⇒

~~.model small~~
~~Start 10H~~
~~data~~
~~portA~~

```
        Assume CS : code
        code  Segment
start : Mov   AL, 80H
        out   CWR , AL
        MOV   AL , 00H
Back  : out   portA , AL
        INC   AL
        CMP   AL, FFH
        JB    Back
Back1 : out   portA , AL
        DEC   AL
        CMP   AL, 00
        JA    Back1
        JMP   Back
        code  Ends
        End   Start
```

1	0	0	0	0	0	0	0

= 80H

PRINCIPAL
SIET., TUMAKURU.

Assignment-04

1. Explain the operation of i) PUSH and POP instruction
   ii) CALL and RET instruction
2. Draw the interrupt vector table and write the sequence of operation that use perform when interrupt is occurred
3. Differentiate b/w Procedures and Macros.
4. Write A program to generate a Delay of 10ms using 8086 microprocessor operating at 10MHz Frequency show the calculation for Delay
5. Explain the stack structure of 8086
6. What are Methods that can be used to pass the parameter to the procedure explain any two of them
7. Write ALP to find factorial of given number.

⇒

1. *PUSH and POP instruction:

⇒ In the stack it contain a set of sequentially arranged data bytes with the last byte is appearing on the top of stack

When the PUSH instruction is done the element of the given address is pushed to the stack and whenever the POP instruction is called the elements present in the stack is return to the main program it is done as First in Last out.

SP = 5000 10x SP = 5000x10
SS = 2500 CS + 50000 / 2500 / 52500

PRINCIPAL
SIET., TUMAKURU.

* Call and RET instruction :-

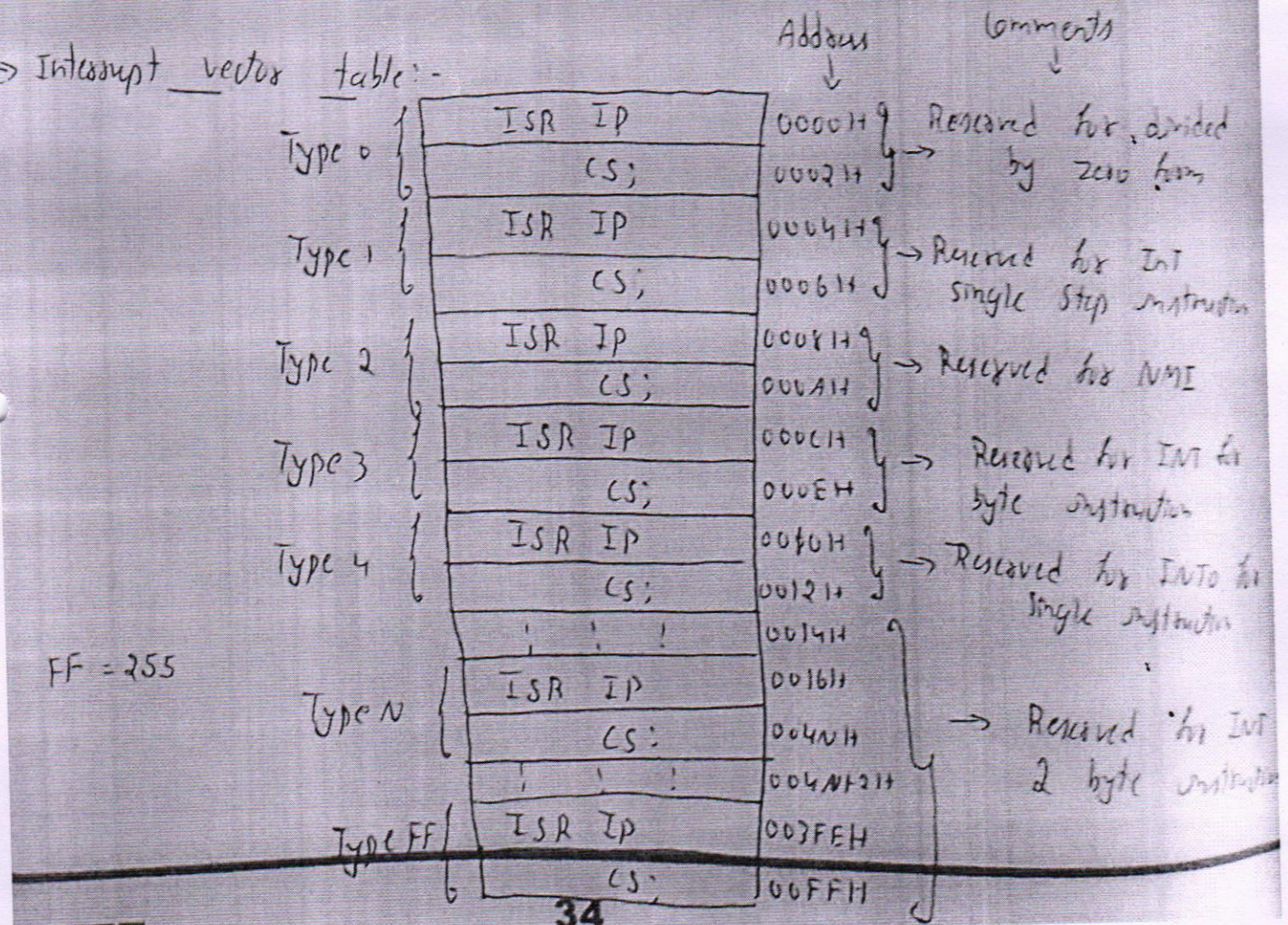
⇒ The call is instruction that calls the subroutine to the main program. when ever call is executed the subroutine of the call program is executed after that during RET instruction it come back to the main program.

⇒ RET : The RET instruction that perform after every call instruction is completed and RET is executed to return back to the main program.

There are 4 types of RET instruction:

i) RET with in segment
ii) RET with segment and multipled by 16-bit
iii) RET with intersegment
iv) RET with intersegment and multipled by 16-bit

⇒ Interrupt vector table :-



PRINCIPAL
SIET., TUMAKURU.

* Type 0 interrupt:- when ever the divided by zero form is obtained in the program Type 0 interrupt is performed and Type 0 interrupt having higher priority than all other interrupts

* Type 1 interrupt:- when for single time delay is required then the INT interrupt instruction

* Type 2 interrupt:- when the NMI in is performed it having separate PIN for in the 8086 it also have a higher priority than all other interrupts expect the Type 0 (divided by zero) instruction

* Type 3 and Type 4 interrupts:-
When the INT and $\overline{INTO}$ is activated mean $\overline{INTO}$ is Low and INT is high the interrupt is performed.

3.

Procedure	MACROS
* Accessed by CALL and RET instructed	* Accessed by the Assemble program in processor
* Machine code is generate only once in procedure	* Machine code is generate for every time when MACROS is called.
* Procedure have less memory space	* MACROS have more memory space
* use stack of memory is cannot define locally	* stack of memory is defined locally

PRINCIPAL
SIET., TUMAKURU.

4 | 10msec 10MHz

⇒ The required delay T_d = 10msec

Instruction Selection	State No execution
MOV CX, count	4
DEC CX	2
NOP	3
JNZ label	16

No of cycles for execution in 21316 = 121 21

Count $N = \frac{\text{required delay } (T_d)}{n \times T} = \frac{10 \times 10^{-3}}{21 \times 0.1M} = 4.761 \times 10^3$

$= 4762$

Count (N) = 4762 = 129AH

```
Pdoc Delay label
Assume CS: code P
Mov CX, 129AH
Wait DEC CX
     NOP
     JNZ wait
     RET
Delay ENDP
```

Delay = 4 X 0.1M + 2 X 129A X 0.1M + 3X 129A X0.1M + 16 X 129A X0.1M

Delay = 10m Sec

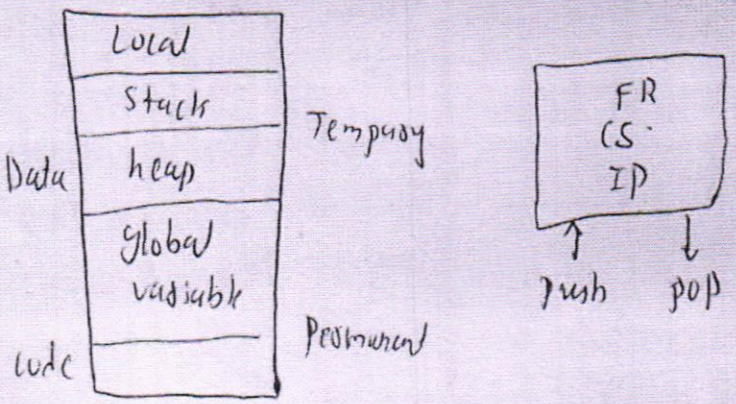
5. ⇒

PRINCIPAL
SIET., TUMAKURU.

Assignment - 05

⇒ Stack Structure of 8086:

The stack of the CPU is portion of the moody memory is present blw the Temporary and permanent memory in the CPU.



The Stack is a section of memory set aside to store return address, save content of registers for the calling program while a procedure execute

In 8086 we can set aside upto 64kb memory as stack. Register. SS is used to store address of stack segment and SP BP are the two default part of the stack.

⇒ The methods of passing the parameter to the proledure are:

1) Stack
2) Registers
3) Memory

Stack is a set of sequential data byte and it is consist of normal Address and physical Address as well and stack segment (SS) and Stack point (SP) are also present. where storing capacity of SS stack segment is

PRINCIPAL
SIET., TUMAKURU.

64 bit and stack pointer is 16 bits

Registers - their main parts in the string the data.

global declaration:

```
Assume cs: code , ds: data
    Data Segment
    Mov ds, 75H
    Data Ends
    code segment
    start: Mov Ax, @data
           Mov ds, Ax
           Mov Ax, 75H
           Mov DX, Ax
           Mov AH, 04ch
           Int 21H
           end start
```

7 =>

```
.model small
.data
MSG1 DB 'ENTER THE NUMBER : $'
MSG2 DB 'THE FACTORIAL : $'
FACT DW ?
.code
Mov ax, @data
Mov ds, ax
disp MSG1
Mov AH, 01H
INT 21H
SUB AL, 30H
Mov BL, AL
```

PRINCIPAL
SIET., TUMAKURU.

```
        MOV  BH, 00H
        MOV  AX, 0001H
        CMP  BX, 0000H
        JE   LAST
UP:     MUL  BX
        DEC  BX
        CMP  BX, 0001H
        JE   LAST
        JMP  UP
LAST:   MOV  FACT, AX
        DISP MSG2
        MOV  AX, FACT
        MOV  CL, 04H
        AND  AX, 0F000H
        ROL  AX, CL
        CALL ASCII
        CALL PRINT
        MOV  AX, FACT
        AND  AX, 00F0H
        ROR  AX, CL
        CALL ASCII
        CALL PRINT
        MOV  AX, FACT
```

PRINCIPAL
SIET. TUMAKURU.

```
AND  AX, 000FH
CALL ASCII
CALL PRINT
MOV  AH, 4CH
INT  21H
ASCII PROC  NEAR
      CMP  AX, 09H
      JE   DOWN
      JC   DOWN
      ADD  AL, 07H
DOWN: ADD  AL, 30H
      RET
ASCII ENDP
PRINT PROC NEAR
      MOV DL, AL
      MOV AH, 02H
      INT 21H
      RET
PRINT ENDP
END
```