20CS401

Hall Ticket Number:

II/IV B.Tech(Regular) DEGREE EXAMINATION

August,2022 — **Computer Science Engineering**

Fourth Semester — **Microprocessors & Microcontrollers**

Time: Three Hours — **Maximum:70 Marks**

Answer question 1 compulsory. **(14X1 = 14 Marks)**

Answer one question from each unit. **(4X14=56 Marks)**

1. a) What is the need of queue in 8086 architecture?
   b) What is assembler directive and list any four assembler directives
   c) List six conditional flags of 8086.
   d) What is the used of data segment in 8086 memory segmentation
   e) Exemplify IF-THEN program in assembly code.
   f) Compare procedure and macro.
   g) Define Reentrant procedure
   h) Explain type-3 interrupt in 8086
   i) What is the operation of INTR and INTA signals
   j) Draw the neat circuit diagram to interface LCD with 8086
   k) List any four arithmetic instructions of 8051
   l) What are the flags affected by ANL instruction
   m) Write the special function of port 3 pins
   n) Compare microprocessors and microcontrollers

Unit –I

2.	a)	Describe all flowchart symbols used to represent program operations	CO1	7M
	b)	Explain briefly about Bus interface unit(BIU) of 8086 Microprocessor.	CO1	7M

(OR)

3.		Describe various assembly language program development tools in detail.	CO1	14M

Unit –II

4.	a)	Write an assembly language program for 8086 to illustrate macro.	CO2	7M
	b)	How do you pass parameters to macro and what is nested macro.	CO2	7M

(OR)

5.		Explain addressing modes of 8086 in detail with sufficient examples.	CO2	14M

Unit –III

6.	a)	Explain the operation of 8259 with neat block diagram	CO3	7M
	b)	Design and explain the microcomputer system using 8086.	CO3	7M

(OR)

7.	a)	Draw neat timing diagram for memory read and write operations and explain in detail.	CO3	7M
	b)	Describe interfacing LED displays to microprocessors	CO3	7M

Unit –IV

8.	a)	Draw and explain the internal architecture of 8051 microcontroller.	CO4	7M
	b)	Draw and explain the Pin diagram of 8051 microcontroller.	CO4	7M

(OR)

9.	a)	Compare different microcontroller members of 8051 family	CO4	7M
	b)	Draw the neat circuit diagram of keyboard interfacing with 8051 and explain the programming steps.	CO4	7M

SCHEME

20CS401

Hall Ticket Number:

II/IV B.Tech(Regular) DEGREE EXAMINATION

August,2022 — **Computer Science Engineering**

Fourth Semester — **Microprocessors & Microcontrollers**

Time: Three Hours — **Maximum:70 Marks**

Answer question 1 compulsory. **(14X1 = 14 Marks)**

Answer one question from each unit. **(4X14=56 Marks)**

1. **a) What is the need of queue in 8086 architecture?**

When EU is busy in decoding and executing an instruction, the BIU fetches up to six instruction bytes for the next instructions. These bytes are called as the pre-fetched bytes and they are stored in a first in first out (FIFO) register set, which is called as a queue. This speed up operations of the processor by **helping to reduce fetches latency.**

b) What is assembler directive and list any four assembler directives

Instructions to the Assembler regarding the program being executed. Control the generation of machine codes and organization of the program; but no machine codes are generated for assembler directives.

Used to:

› specify the start and end of a program

› attach value to variables

› allocate storage locations to input/ output data

› define start and end of segments, procedures, macros etc..

DB,DW,SEGMENT,ENDS,ASSUME,ORG,END,EVEN,EQU,PROC,FAR,NEAR,ENDP,SHORT, MARO, ENDM

c) List six conditional flags of 8086.

- The carry flag
- The parity flag
- The Auxiliary Carry Flag
- The Zero Flag
- The sign Flag
- The overflow Flag

d) What is the used of data segment in 8086 memory segmentation

Operands for most instructions are fetched from this segment.

e) Exemplify IF-THEN program in assembly code.

- **Structure:** IF condition THEN

  action

  action

- IF the stated condition is true, then the series of actions following THEN will be executed.
- If the condition is false then, skip the actions followed by THEN and continue with next statement.

f) Compare procedure and macro.

Difference between PROCEDURE & MACRO

Procedure	Macro
Procedures are used for large group of instructions to be repeated.	Macros are used for small group of instructions to be repeated.
Object code is generated only once in memory.	Object code is generated every time the macro is called.
CALL & RET instructions are used to call procedure and return from procedure.	Macro can be called just by writing its name.
Length of the object file is less	Object file becomes lengthy.
Directives PROC & ENDP are used for defining procedure.	Directives MACRO and ENDM are used for defining MACRO
More time is required for it's execution	**Less time is required for it's execution**
Procedure can be defined as **Procedure_name PROC** ---- ------ **Procedure_name ENDP**	**Macro can be defined as** MACRO-name MACRO [ARGUMENT ,..... ARGUMENT N] ------ ------- **ENDM**
For Example **Addition PROC near** ------ **Addition ENDP**	**For Example** **Display MACRO** --------- **ENDM**

g) Define Reentrant procedure

This is called a re-entrant procedure because a procedure is re-entering into itself form another procedure which is also present inside its own body.

- The re-entrant procedure occurs in the following three conditions:
  - when the procedure is undergoing recursion,
  - when multi-threading is being implemented inside a program or
  - when some interruption is being generated.

h) Explain type-3 interrupt in 8086

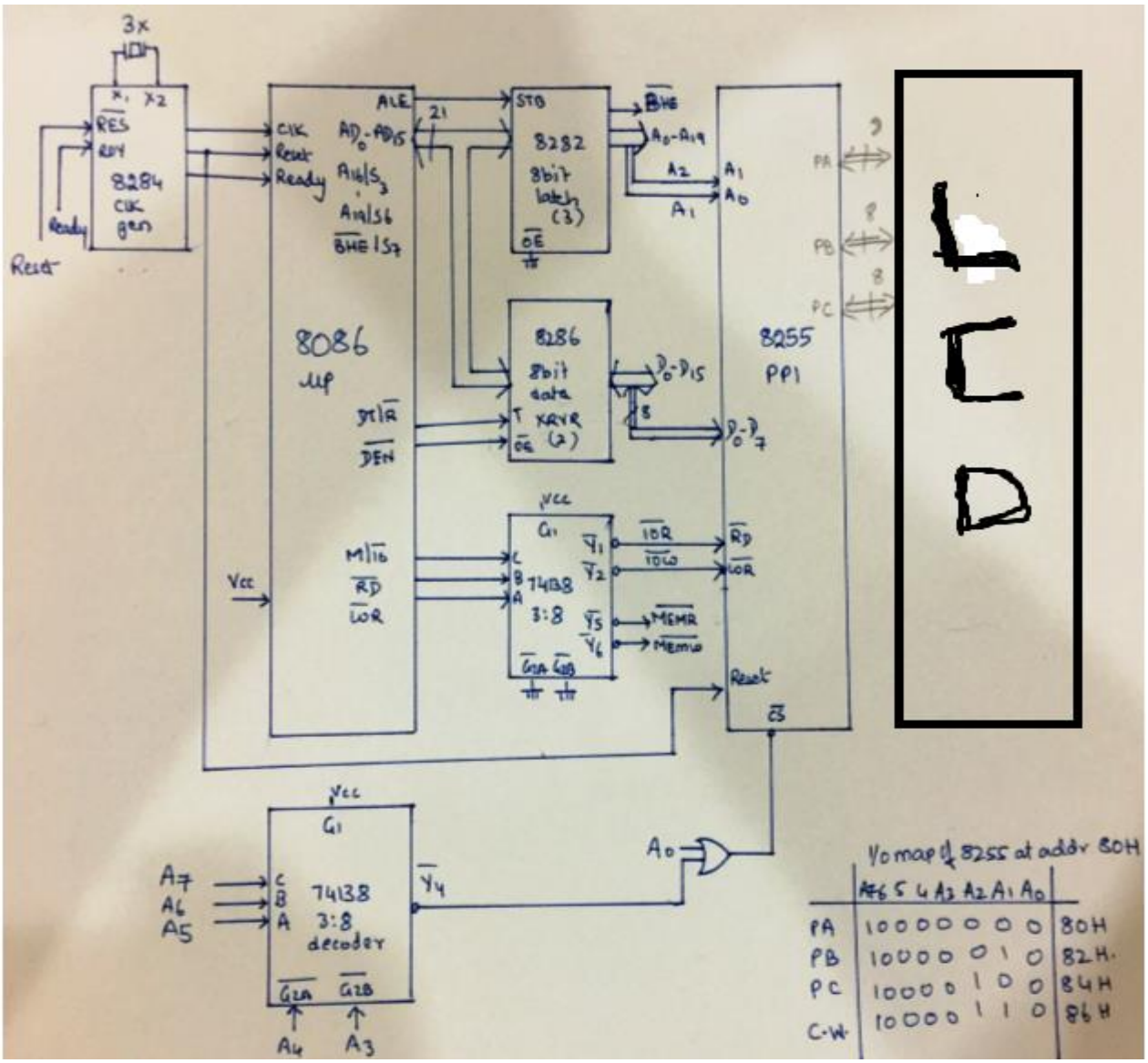
- **INT 3 (Breakpoint Interrupt)-Type 3**
  - A break point is used to examine the CPU and memory after the execution of a group of Instructions.
  - This interrupt is used to cause breakpoints in the program.
  - It is caused by writing the instruction INT 03H or simply INT.
  - It is useful in debugging large programs where single stepping is efficient.
  - Its ISR is used to display the contents of all registers on the screen.
  - Its ISR address is stored at location 3 x 4 = 0000CH in the IVT.

i) What is the operation of INTR and INTA signals

INTR:- It provides a single interrupt request and is activated by the I/O port. This interrupt can be masked or delayed. It is a level-triggered interrupt. It can receive any interrupt type, so the value of IP and CS will change on the interrupt type received.

INTA:- Interrupt acknowledge. It is active low (0) during T2, T3 and Tw of each interrupt acknowledge cycle.

j) Draw the neat circuit diagram to interface LCD with 8086



k) List any four arithmetic instructions of 8051

- Addition
- Multiplication
- Subtraction
- Division
- Increment
- Decrement

l) What are the flags affected by ANL instruction

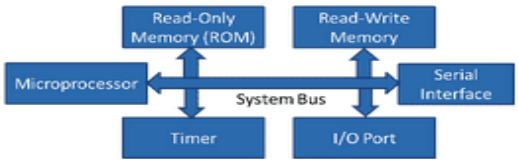
Logic instructions do NOT affect the flags in PSW.

m) Write the special function of port 3 pins

It is 8-bit I/O port. In an alternating function each pins can be used as a special function I/O pin.

Port Pin	Alternate Function
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{\text{INT0}}$ (external interrupt 0)
P3.3	$\overline{\text{INT1}}$ (external interrupt 1)
P3.4	T0 (Timer 0 external input)
P3.5	T1 (Timer 1 external input)
P3.6	$\overline{\text{WR}}$ (external data memory write strobe)
P3.7	$\overline{\text{RD}}$ (external data memory read strobe)

n) Compare microprocessors and microcontrollers

Microprocessor	Micro Controller
	
Microprocessor is heart of Computer system.	Micro Controller is a heart of embedded system.
It is just a processor. Memory and I/O components have to be connected externally	Micro controller has external processor along with internal memory and i/O components
Since memory and I/O has to be connected externally, the circuit becomes large.	Since memory and I/O are present internally, the circuit is small.
Cannot be used in compact systems and hence inefficient	Can be used in compact systems and hence it is an efficient technique
Cost of the entire system increases	Cost of the entire system is low
Due to external components, the entire power consumption is high. Hence it is not suitable to used with devices running on stored power like batteries.	Since external components are low, total power consumption is less and can be used with devices running on stored power like batteries.
Most of the microprocessors do not have power saving features.	Most of the micro controllers have power saving modes like idle mode and power saving mode. This helps to reduce power consumption even further.
Since memory and I/O components are all external, each instruction will need external operation, hence it is relatively slower.	Since components are internal, most of the operations are internal instruction, hence speed is fast.
Microprocessor have less number of registers, hence more operations are memory based.	Micro controller have more number of registers, hence the programs are easier to write.
Microprocessors are based on von Neumann model/architecture where program and data are stored in same memory module	Micro controllers are based on Harvard architecture where program memory and Data memory are separate
Mainly used in personal computers	Used mainly in washing machine, MP3 players

Unit –I

2. a) Describe all flowchart symbols used to represent program operations CO1 7M

- Formula or sequence of operations used to solve a programming problem is called as the algorithm.
- **There are two ways of representing algorithms:-**
  - Flowchart
  - Structured programming and pseudo code.

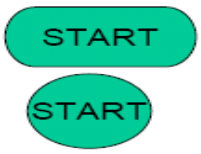
Flowchart:-It uses graphical shapes to represent different types of programming operations.

Flow chart symbols are given below:-

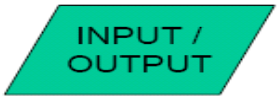
PROGRAM DEVELOPMENT STEPS

FLOWCHARTS:

Specific operation desired is written in the graphic symbol



A *racetrack- or circular-* shaped symbol is labeled *START* is used to indicate the *beginning* of the program.



A *parallelogram* is used to represent an *input* or an *output* operation.

A rectangular box symbol is used to represent simple operations other than input and output operations.

PROGRAM DEVELOPMENT STEPS

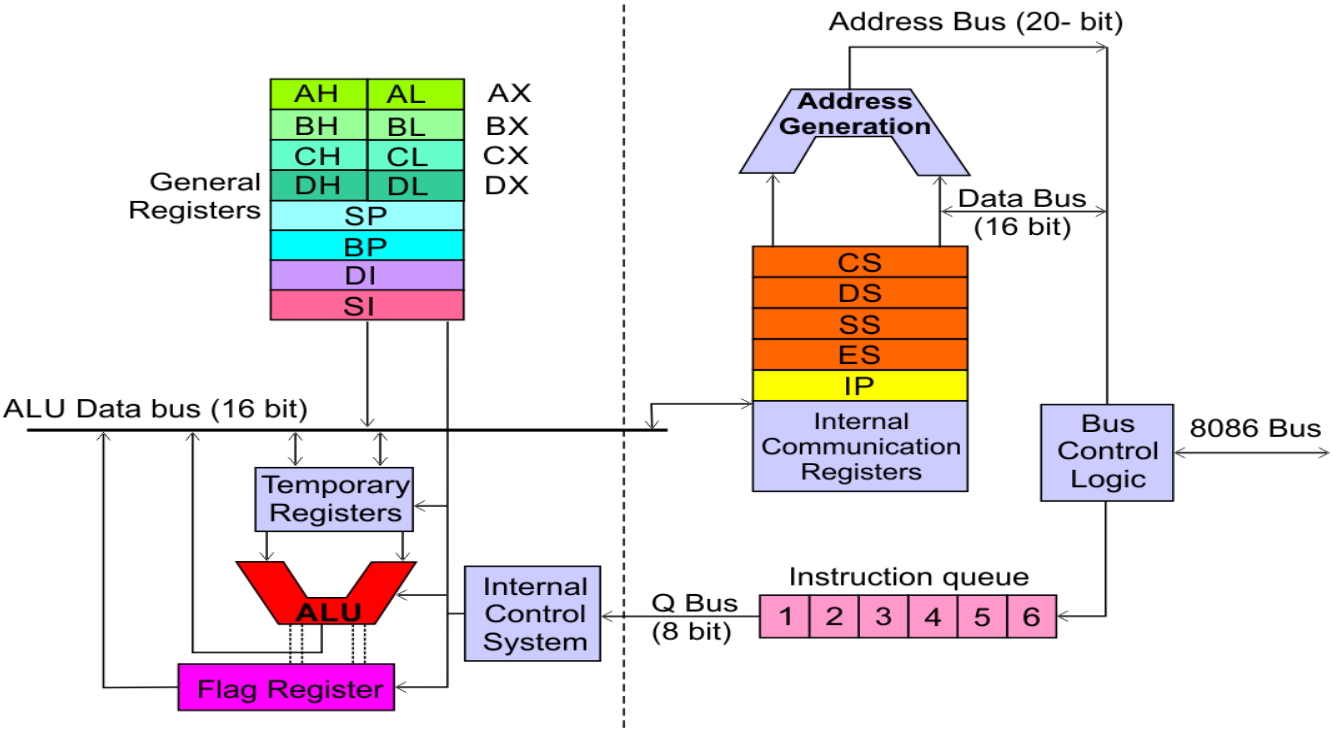
A rectangular box with double lines at each end is often used to represent a *subroutine* or *procedure*.

A *diamond* is used to represent an *decision* point or crossroad.
Usually it indicates that some condition is to be checked at this point.

A Connector symbol is used to represent a break in the column and it will be used in the next column for representing continuation.

A five-sided Off-Page-Connector is used to represent a break in the page and it will be used in the next page for representing continuation.

b) Explain briefly about Bus interface unit (BIU) of 8086 Microprocessor. CO1 7M



Execution Unit (EU)
EU executes instructions that have already been fetched by the BIU. BIU and EU functions separately.

Bus Interface Unit (BIU)
BIU fetches instructions, reads data from memory and I/O ports, writes data to memory and I/ O ports.

Bus Interface Unit (BIU)

Dedicated Adder to generate 20 bit address

Four 16-bit segment registers

Code Segment (CS)

Data Segment (DS)

Stack Segment (SS)

Extra Segment (ES)

Code Segment Register

- 16-bit
- CS contains the base or start of the current code segment; IP contains the distance or offset from this address to the next instruction byte to be fetched.

- BIU computes the 20-bit physical address by logically shifting the contents of CS 4-bits to the left and then adding the 16-bit contents of IP.
- That is, all instructions of a program are relative to the contents of the CS register multiplied by 16 and then offset is added provided by the IP.

Data Segment Register

- 16-bit
- Points to the current data segment; operands for most instructions are fetched from this segment.
- The 16-bit contents of the Source Index (SI) or Destination Index (DI) or a 16-bit displacement are used as offset for computing the 20-bit physical address.

Stack Segment Register

- 16-bit
- Points to the current stack.
- The 20-bit physical stack address is calculated from the Stack Segment (SS) and the Stack Pointer (SP) for stack instructions such as PUSH and POP.
- In based addressing mode, the 20-bit physical stack address is calculated from the Stack segment (SS) and the Base Pointer (BP).

Extra Segment Register

- 16-bit
- Points to the extra segment in which data (in excess of 64K pointed to by the DS) is stored.
- String instructions use the ES and DI to determine the 20-bit physical address for the destination.

Instruction Pointer

- 16-bit
- Always points to the next instruction to be executed within the currently executing code segment.
- So, this register contains the 16-bit offset address pointing to the next instruction code within the 64Kb of the code segment area.
- Its content is automatically incremented as the execution of the next instruction takes place.

Instruction queue

- A group of First-In-First-Out (FIFO) in which up to 6 bytes of instruction code are pre fetched from the memory ahead of time.
- This is done in order to speed up the execution by overlapping instruction fetch with execution.
- This mechanism is known as pipelining.

(OR)

3. Describe various assembly language program development tools in detail. CO1 14M

Assembly language program development tools are:-

1. Editor
2. Assembler
3. Linker
4. Locator
5. Debugger
6. Emulator

Editor:-

- An Editor is a program which allows you to create an ALP program.
- The Editor stores the ASCII codes for letters and numbers in successive RAM locations.
- After you type the program, save the file with extension ".asm". Ex: add.asm
- This file is called source file.
- Next the source file is processed with assembler like TASM/MASM.(Turbo/Microsoft Marco Assemblers).

Assembler:-

- Assembler Translates ALP into binary Codes.
- The Processing of the Assembler is as follows:
  - Determines the displacements of named data items, offset of labels, and puts this information in symbol table.
  - Assembler produces binary code for each instruction and inserts the offset during the first process.
- The Assembler generates two output files.
- The first one is Object file which has extension ".obj "
- This file contains binary codes and information about the address of instructions.
- The Contents of this file will be loaded in memory and run.
- The 2nd file generated is assembler list file and the extension is ".LST ".
- The list file contains
- 1) Assembly Language Statements
- 2) Binary Codes
- 3) Offsets for each Instruction
- This file is used for printing and testing the program.
- Note: Assembler finds only Syntax errors.
- It will not test the logic of the program.

Linker:-

- Linker is a program which joins several object files into one large object file.
- Linker Produces Link File and Link Map file.
- Link file contains all the binary codes for all combined modules.
- Map file contains the address information about the linked files.
- Linker does not assign absolute addresses but relative addresses starting from zero.

Locator:-

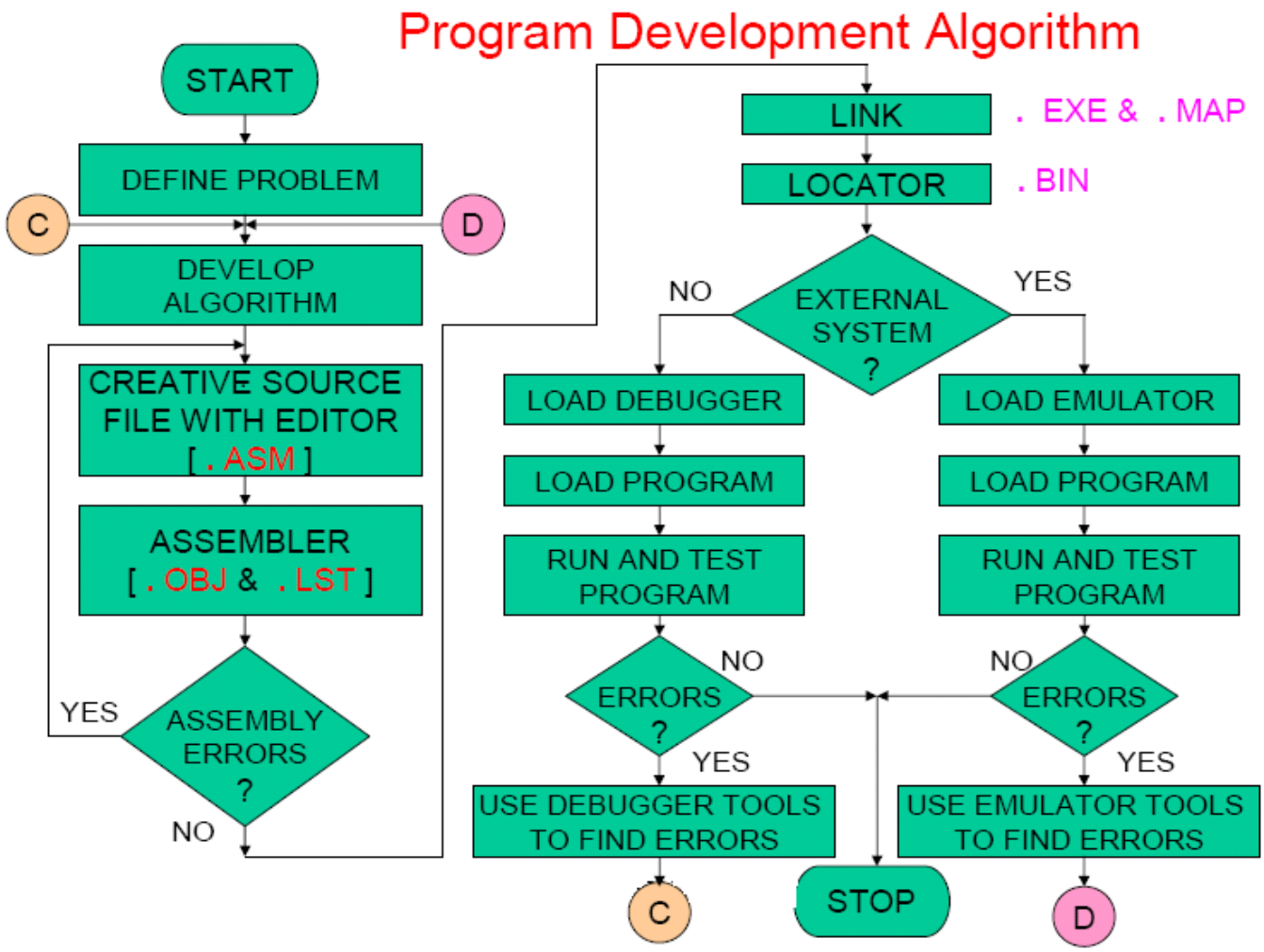
- Locator is a program which is used to assign the specific addresses of where the segments of object code are to be loaded in memory.

Debugger:-

- Debugger is a program which loads the object code into system memory.
- It is used for run and debug the program.
- We can change the contents memory and registers during debugging.

Emulator:-

- Emulator is used is used to debug the hardware and software of external system.
- It is used for run and debug the program.
- We can change the contents memory and registers during debugging.
- We can take the snapshot of the contents of registers, activity on the address and data bus, and state of the flags as each instruction is executes.



Unit –II

4. a) Write an assembly language program for 8086 to illustrate macro. CO2 7M

- A MACRO is group of small instructions that usually performs one task.
- It is a reusable section of a software program.
- A macro can be defined anywhere in a program using directive MACRO &ENDM.
- The Label prior to MACRO is the macro name which should be used in the actual program.
- The ENDM directive marks the end of the instructions.
- **Macro syntax:**

name MACRO [parameters,...]

<instructions>

ENDM

Example:- Defining and calling a Macro without parameters

```
PUSH-ALL  MACRO
          PUSHF
          PUSH AX
          PUSH BX
          PUSH CX
          PUSH DX
          PUSH BP
          PUSH SI
          PUSH DI
          PUSH DS
          PUSH ES
          PUSH SS
ENDM
```

b) How do you pass parameters to macro and what is nested macro. CO2 7M

- A MACRO is group of small instructions that usually performs one task.
- It is a reusable section of a software program.
- A macro can be defined anywhere in a program using directive MACRO &ENDM.
- The Label prior to MACRO is the macro name which should be used in the actual program.
- The ENDM directive marks the end of the instructions.
- **Macro syntax:**

Name MACRO [parameters...]

<Instructions>

ENDM

Example: Passing parameters to Macros

```
MyMacro MACRO p1, p2, p3
MOV AX, p1
MOV BX, p2
MOV CX, p3
ENDM
ORG 100h
MyMacro 1, 2, 3
MyMacro 4, 5, DX
RET
```

Nested macro :- A macro calling another macro is called a nested macro. TASM and MASM can contain a macro that calls a previously defined macros. All macros should be defined before segment definition though their order can be anything.

```
DISPLAY_CHAR MACRO CHAR
```

```
        PUSH  AX
        PUSH  DX
        MOV AH, 02H
        MOV DL, CHAR
        INT 21H
        POP DX
        POP AX
ENDM
CRLF  MACRO
        DIPLAY_CHAR  0DH
        DISPLAY_CHAR  0AH
ENDM
```

(OR)

5. Explain addressing modes of 8086 in detail with sufficient examples. CO2 14M

- Every instruction of a program has to operate on a data.
- The different ways in which a source operand is denoted in an instruction are known as **addressing modes.**

1. Register Addressing
2. Immediate Addressing
3. Direct Addressing
4. Register Indirect Addressing
5. Based Addressing
6. Indexed Addressing
7. Based Index Addressing
8. String Addressing
9. Direct I/O port Addressing
10. Indirect I/O port Addressing
11. Relative Addressing
12. Implied Addressing

Register mode – In this type of addressing mode both the operands are registers.
Example:

```
MOV AX, BX

XOR AX, DX

ADD AL, BL
```

Immediate mode – In this type of addressing mode the source operand is a 8 bit or 16 bit data. Destination operand can never be immediate data.
Example:

```
MOV AX, 2000

MOV CL, 0A

ADD AL, 45

AND AX, 0000
```

Note that to initialize the value of segment register an register is required.

```
MOV AX, 2000

MOV CS, AX
```

Displacement or direct mode – In this type of addressing mode the effective address is directly given in the instruction as displacement.
Example:
MOV AX, [DISP]

MOV AX, [0500]

Register indirect mode – In this addressing mode the effective address is in SI, DI or BX.
Example: Physical Address = Segment Address + Effective Address
MOV AX, [DI]

ADD AL, [BX]

MOV AX, [SI]

Based indexed mode – In this the effective address is sum of base register and index register.
Base register: BX, BP

Index register: SI, DI

The physical memory address is calculated according to the base register.
Example:

MOV AL, [BP+SI]

MOV AX, [BX+DI]

Indexed mode – In this type of addressing mode the effective address is sum of index register and displacement.
Example:
MOV AX, [SI+2000]

MOV AL, [DI+3000]

Based mode – In this the effective address is the sum of base register and displacement.
Example:
MOV AL, [BP+ 0100]

String mode – This addressing mode is related to string instructions. In this the value of SI and DI are auto incremented and decremented depending upon the value of directional flag.
Example:
MOVS B

MOVS W

Direct port addressing mode-- an 8-bit port address is directly specified in the instruction.

Example: IN AL, [09H]

Operations: $PORT_{addr} = 09_H$

(AL) ← (PORT)

Content of port with address 09_H is moved to AL register

Indirect port addressing mode-- The instruction will specify the name of the register which holds the port address. In 8086, the 16-bit port address is stored in the DX register.

Example: OUT [DX], AX

Operations: $PORT_{addr}$ = (DX)

(PORT) <-- (AX)

Content of AX is moved to port whose address is specified by DX register.

Relative mode-- In this the effective address is calculated with reference to instruction pointer.
Example:
JNZ 8 bit address

IP=IP+8 bit address

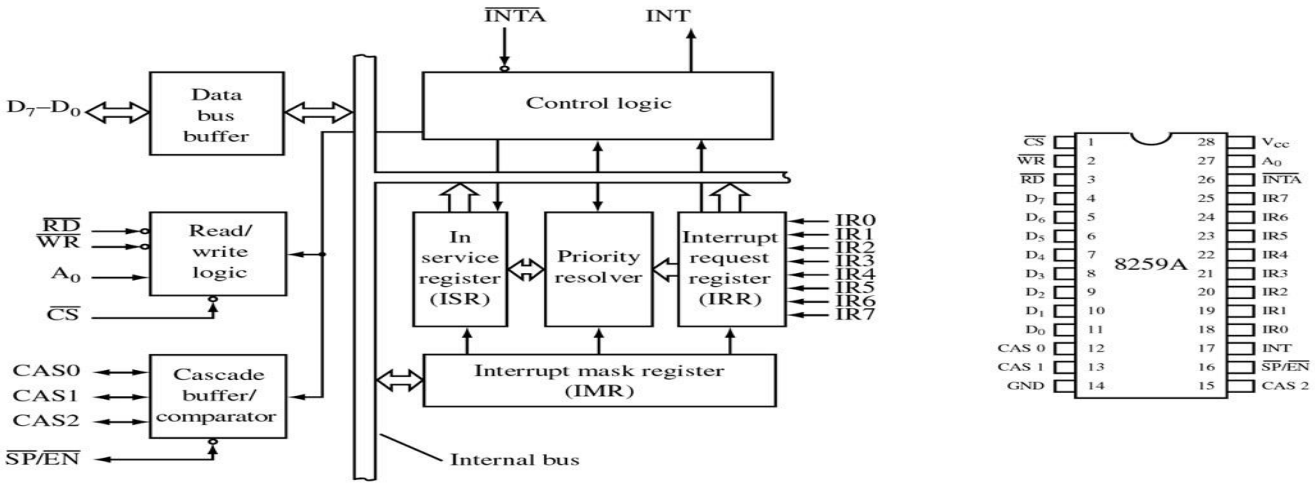
Implied mode-- Instructions using this mode have no operands. The instruction itself will specify the data to be operated by the instruction.

Example: CLC

This clears the carry flag to zero.

Unit –III

6. a) Explain the operation of 8259 with neat block diagram CO3 7M



The Programmable Interrupt Controller (PlC) functions as an overall manager in an Interrupt-Driven system.

1) When 2 interrupts come at a time to 8259 Interrupt inputs (IR0-IR7), the highest priority will be for IR0 and lowest priority will be for IR7.
2) Priority of Interrupts is based on the 4 blocks in 8259. They are
3) **Interrupt Request Register:- (IRR**): Keeps track of which interrupts are asking for service.
4) **Interrupt Mask Register:- (IMR):** Disable or enable individual interrupt inputs.
5) **In-Service Register:- (ISR):** Keeps track of which interrupts are being serviced.
6) **Priority Resolver:-** Determines when an interrupt request on one of the IR input gets serviced.

8259 Functional Description

7) CS' for enabling the device.
8) IR0-IR7 are Interrupt input pins.
9) WR' for accepting command words from 8086.
10) RD' for releasing status onto the data bus.

11) Cas2-cas0 are used to connect multiple 8259 devices.
12) INT is used to interrupt 8086 when an interrupt is generated in 8259.
13) INTA' is used to enable the interrupt pointer onto the data bus.
14) A0 is connected to 8086 Address line A1.
15) SP'/EN' is used for Master Slave or Buffer Enable.

b) Design and explain the microcomputer system using 8086. CO3 7M

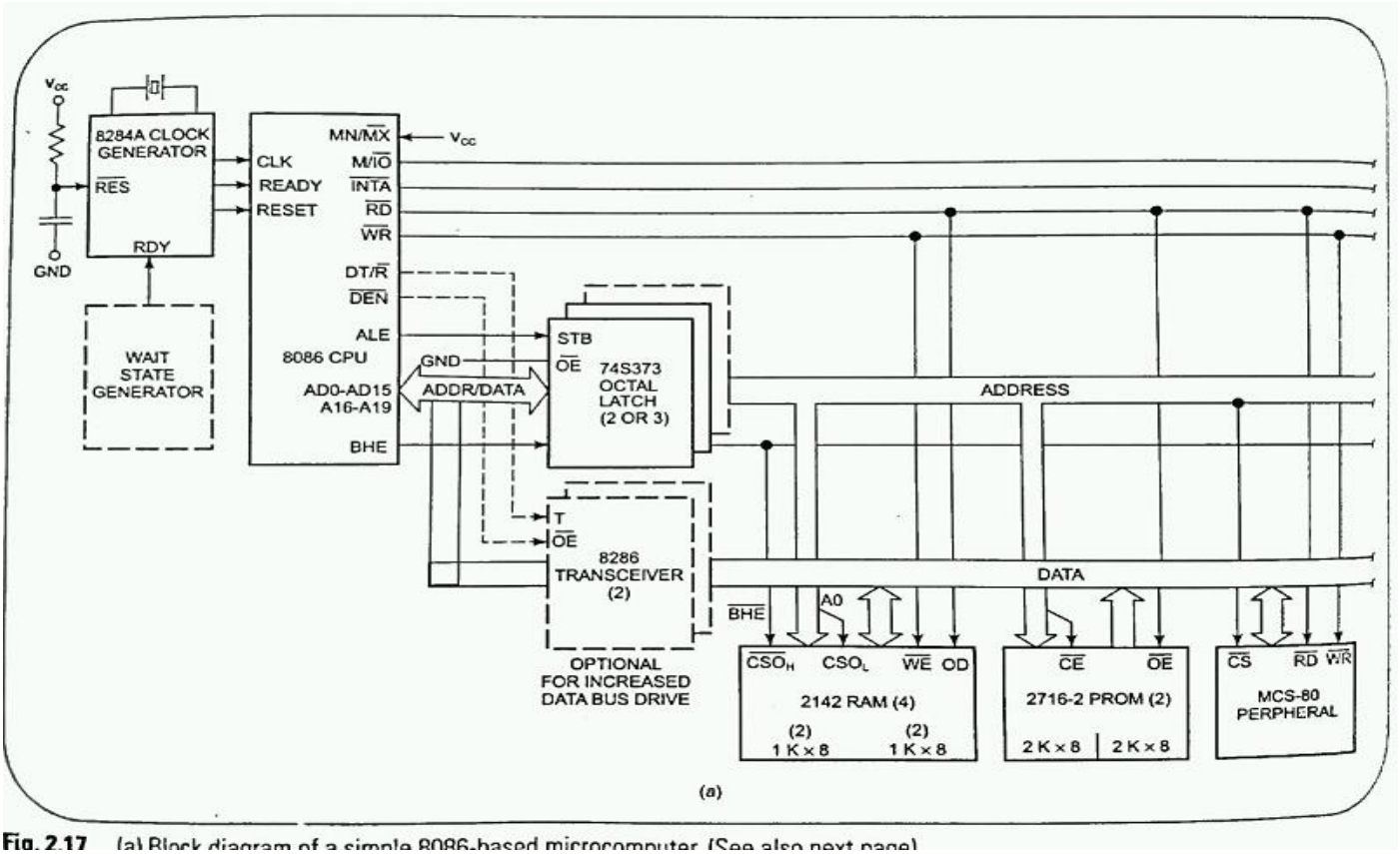


Fig. 2.17 (a) Block diagram of a simple 8086-based microcomputer. (See also next page)

Block diagram of a simple 8086 based computer. There are wide varieties of ports available. Some examples are parallel port devices such as the 8255A, serial port devices, special port devices in which interface with CRT's, port devices which interface with keyboards, and port devices which interface with floppy disks. If the 8086 is doing a read from the memory or from a port, the RD signal will be asserted. If the 8086 doing a write to memory or to a port, the WR signal will be asserted.

The basic control bus consists of the signals labeled M/IO, RD, WR:-

If the 8086 is doing a read from memory (or) from port the RD- signal will be asserted.
the 8086 is doing a write to memory (or) write to port the WR- signal will be asserted.
The first step in any operation where it access memory or port, The 8086 sends out the lower 16-bits of the address on the data bus. External latches such as 74LS373 octal device are used to "grap". This address and hold it during the rest of the operation.
To strobe this latches at the proper time , the 8086 outputs a signal called address enable latch(ALE).
Once the address is stored on the output of the latches, the 8086 removes the address from the address/data bus and uses the bus for reading or writing the data.

The 8286 transceiver block represents bidirectional three state buffers. Buffers used on the data bus must be bidirectional.

The 8284A Clock generator device uses a crystal to produce stable frequency clock signal which steps the 8086 through execution of its instructions in an orderly manner. 8284A also synchronizes the RESET signal and READY signal with clock so that these signals are applied to 8086 properly.

(OR)

7. **a) Draw neat timing diagram for memory read and write operations CO3 7M and explain in detail.**

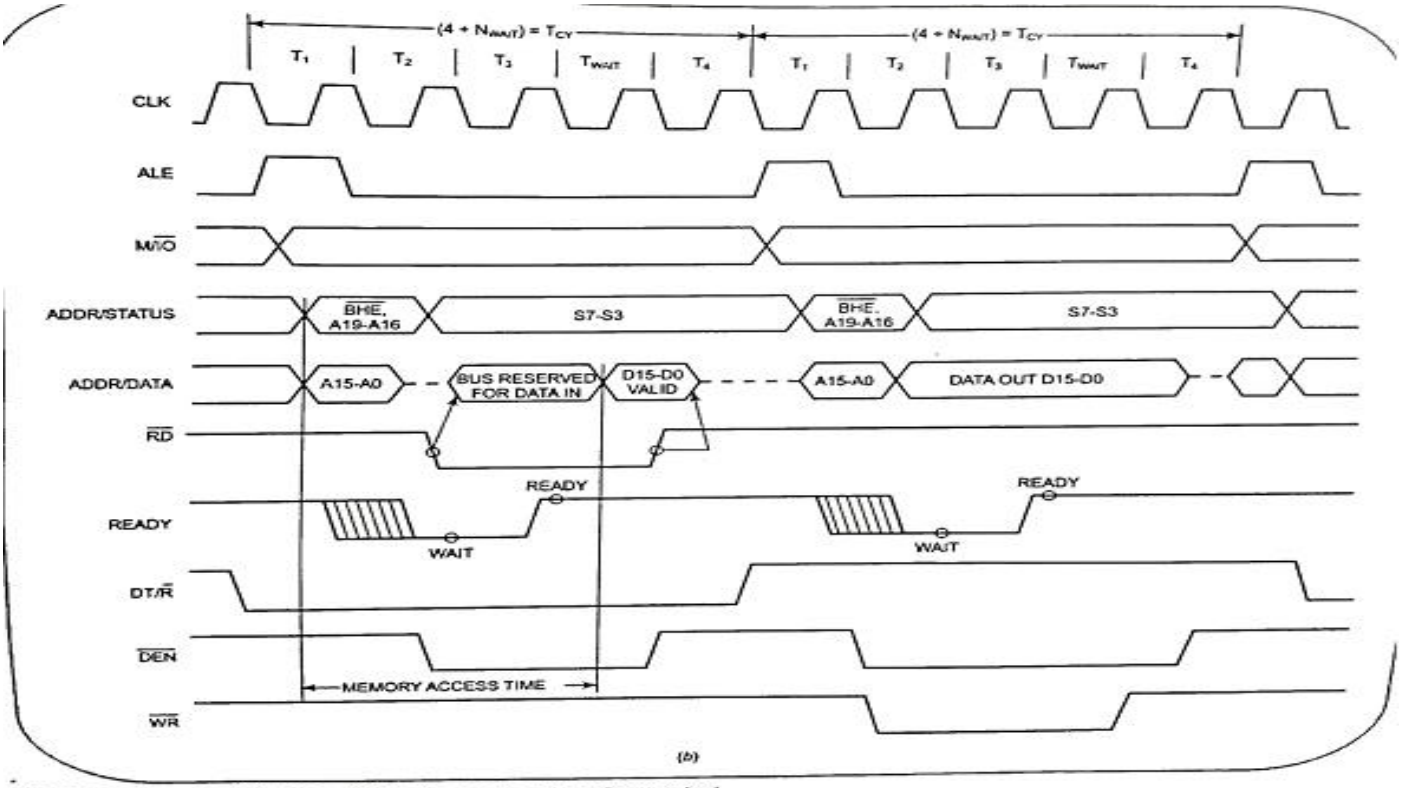


Fig. 2.17 (continued) (b) Basic 8086 system timing. (Intel Corporation)

- Here 8086 is operated as a Minimum mode..
- The time require for a microcomputer to fetch and execute an entire instruction is called instruction cycle.
- An instruction cycle consists of one or more machine cycles.
- An instruction cycle is made up of machine cycles, and machine cycle is made up of states.
- The 8086 asserts M/IO' high if the read is to be from memory and asserts M/IO' low if the read is going to be from a port.

READ CYCLE:-

- At T_1 state ALE =1, this indicates that a valid address is latched on the address bus and also
- M / IO'= 1, which indicates the memory operation is in progress.
- In T_2, the address is removed from the local bus and is sent to the addressed device. Then the bus is tristated.
- When RD' = 0, the valid data is present on the data bus.
- During T_2 DEN' =0, which enables transceivers and DT/R' = 0, which indicates that the data is received.
- During T_3, data is put on the data bus and the processor reads it.
- The output device makes the READY line high. This means the output device has performed the data transfer process. When the processor makes the read signal to 1, then the output device will again tristate its bus drivers.

WRITE CYCLE:-

- At T_1 state ALE =1, this indicates that a valid address is latched on the address bus and also M / IO'= 1, which indicates the memory operation is in progress.
- In T_2, the processor sends the data to be written to the addressed location.
- The data is buffered on the bus until the middle of T_4 state.
- The WR'=0 becomes at the beginning of T_2.
- The BHE' and A0 signals are used to select the byte or bytes of memory or I/O word.
- During T_2 DEN' =0, which enables, transceivers and DT/R' = 1, which indicates that the data is transferred by the processor to the addressed device.

b) Describe interfacing LED displays to microprocessors CO3 7M

- **Alphanumeric LED displays are available in three common formats.**
- For displaying only number and hexadecimal letters, **simple 7-segment displays** such as that as shown in fig are used.
- To display numbers and the entire alphabet, **18 segment displays** and **5 by 7 dot-matrix displays** as shown in fig can be used.
- The 7-segment type is the least expensive, most commonly used and easiest to interface with, so we will concentrate first on how to interface with this type.
- **Directly Driving LED Displays:-** Figure shows a circuit that you might connect to a parallel port on a microcomputer to drive a single 7-segment , common anode display. For a common-anode display, a segment is tuned on by applying a logic low to it.
- The 7447 converts a BCD code applied to its inputs to the pattern of lows required to display the number represented by the BCD code.
- This circuit connection is referred to as a *static display* because current is being passed through the display at all times.
- Each segment requires a current of between 5 and 30mA to light. Let's assume you want a current of 20mA. The voltage drop across the LED when it is lit is about 1.5V.
- The output low voltage for the 7447 is a maximum of 0.4V at 40mA. So assume that it is about 0.2V at 20mA.
- Subtracting these two voltage drop from the supply voltage of 5V leaves 3.3V across the current limiting resistor.
- Dividing 3.3V by 20mA gives a value of 168Ω for the current-limiting resistor.
- The voltage drops across the LED and the output of 7447 are not exactly predictable and exact current through the LED is not critical as long as we don't exceed its maximum rating.

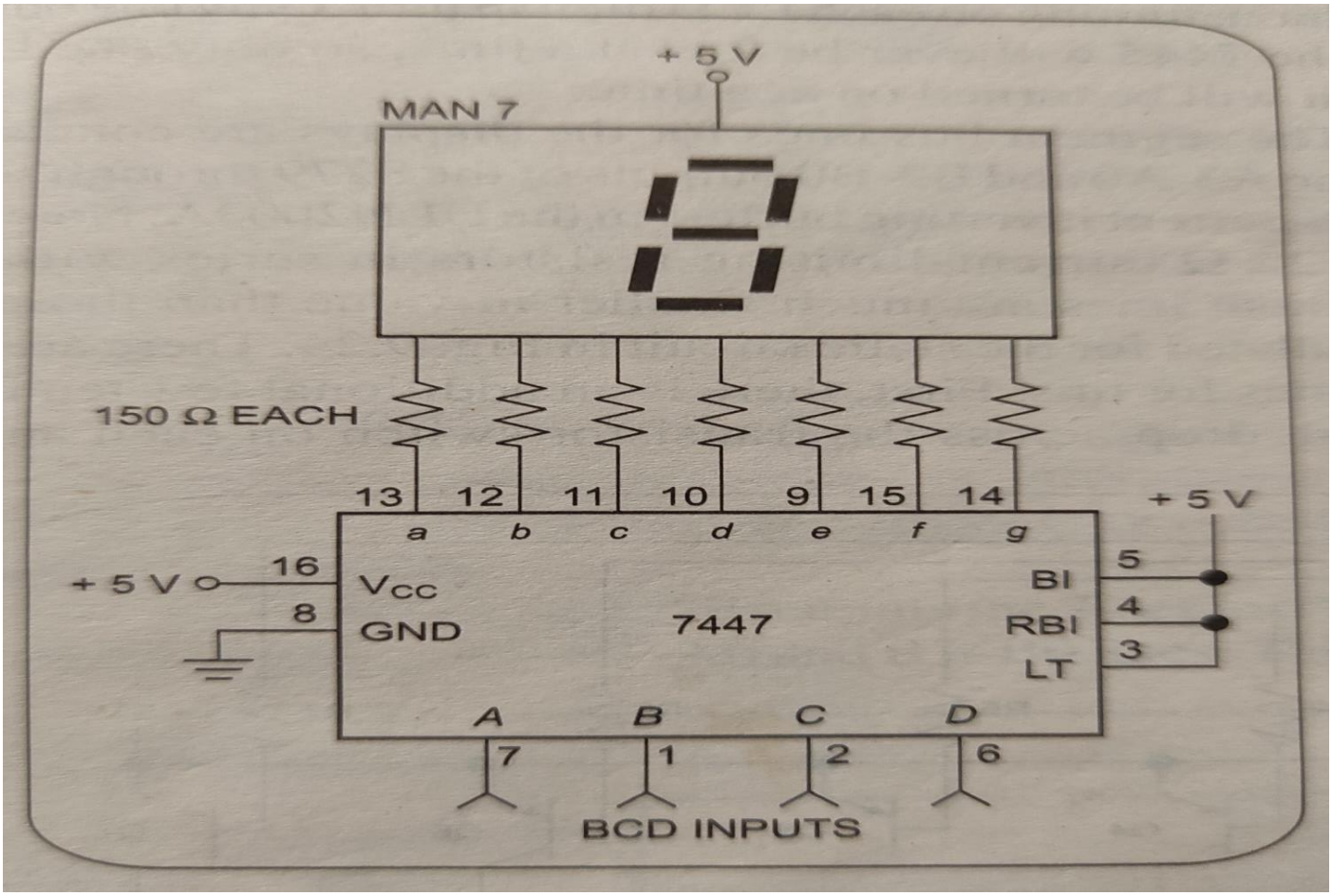


Fig:- Circuit for single 7-sigment LED display with 7447

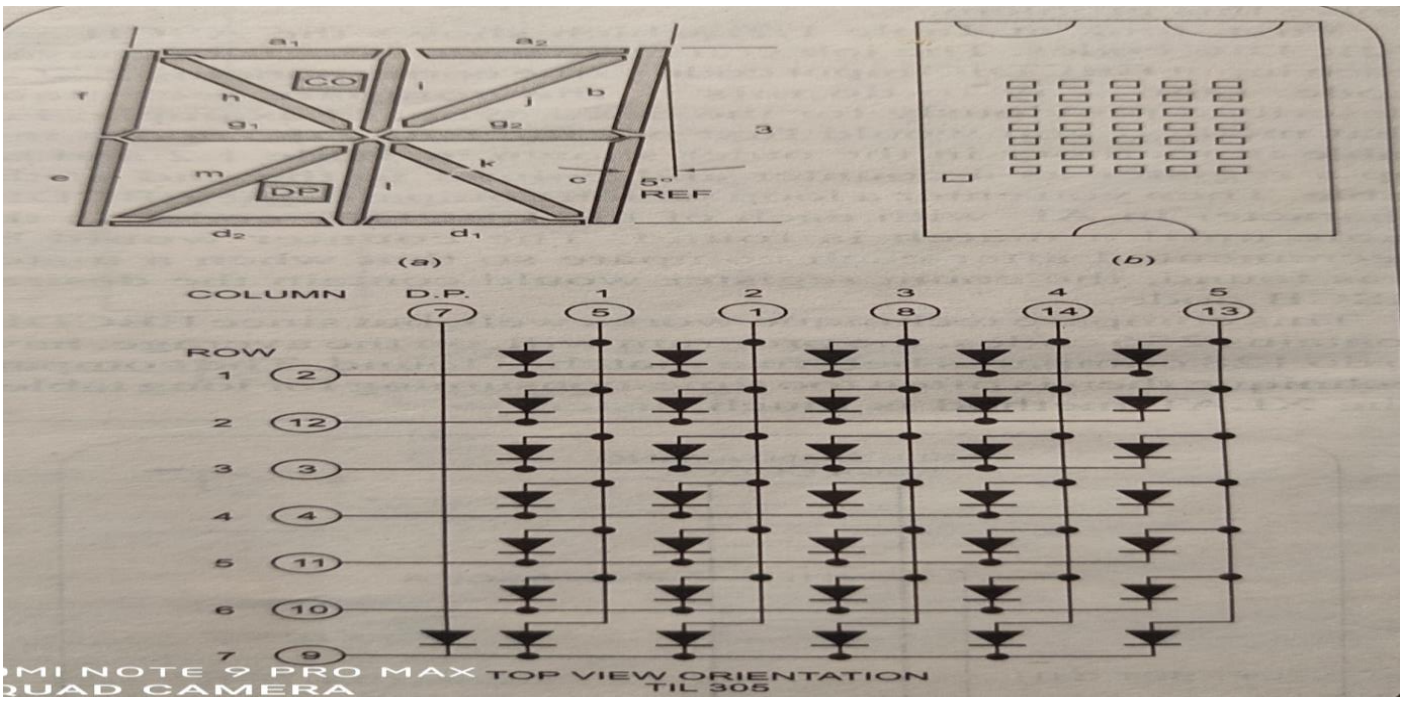


Fig (a):- 18 segment device,
Fig (b):- dot-matrix LED device (5x7),
Fig(c):- 5x7 dot matrix connections (Common anode configuration)

Software-Multiplexed LED Display:-

- The circuit in fig works for driving just one or two LED digits with a parallel output port. However, this scheme has several problem if you want to drive, eight digits.
- The first problem is power consumption. For worst-case calculations, assume that all 8 digits are displaying the digit 8, so all 7 segments are all lit. Seven segment time 20mA per segment gives a current of **140mA** per digit. Multiplying this by 8 digits gives a total current of **1120mA** or 1.12A for 8 digits.
- A second problem of the **static approach** is that each display digit requires a separate 7447 decoder, each of which uses of another **13mA**.
- The current required by the decoders and the LED displays might be several times the current required by the reset of the circuitry in the instrument.
- To solve the problem of the static display approach, we use a *multiplex method*, example for an explanation of the multiplexing.

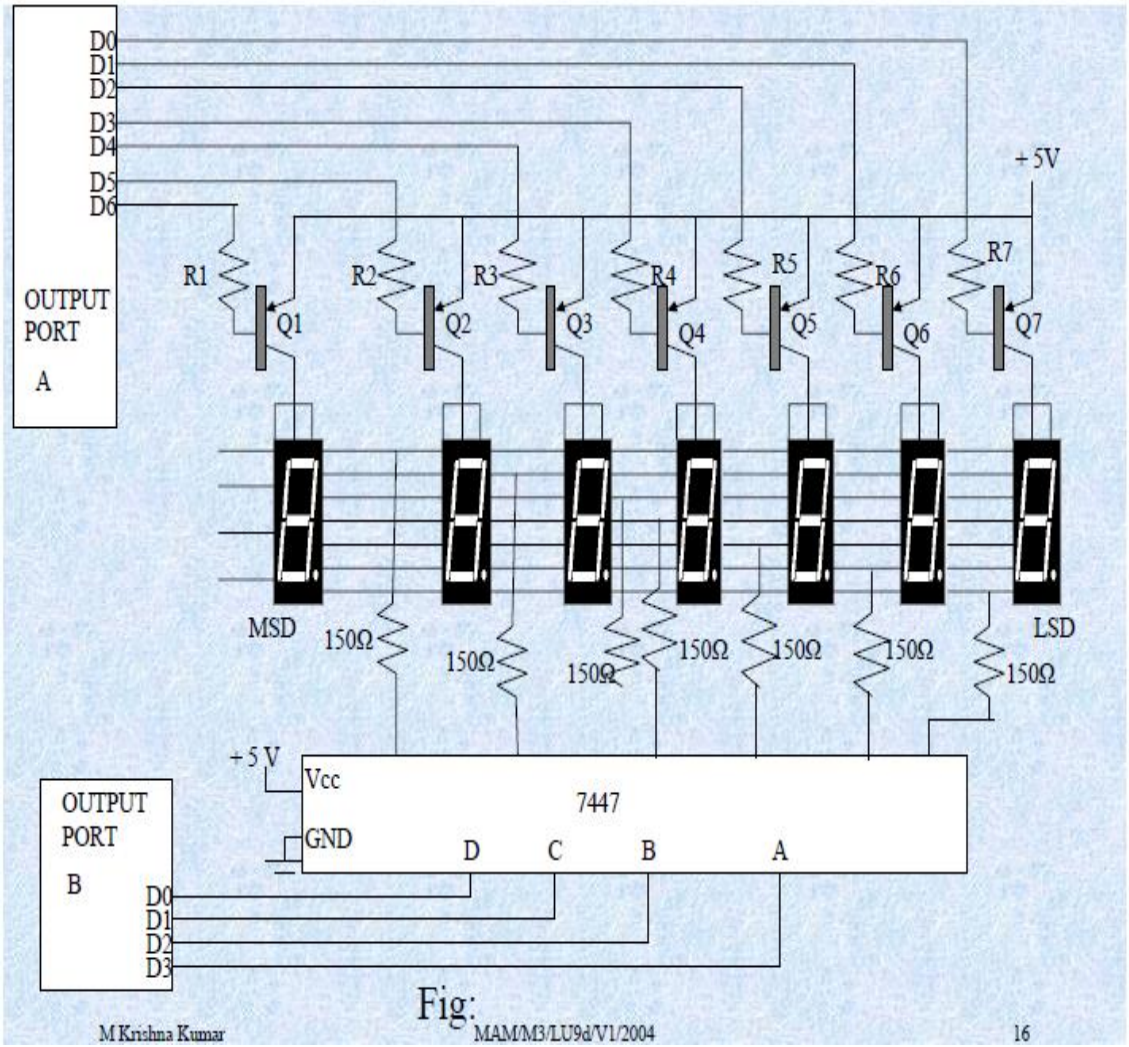
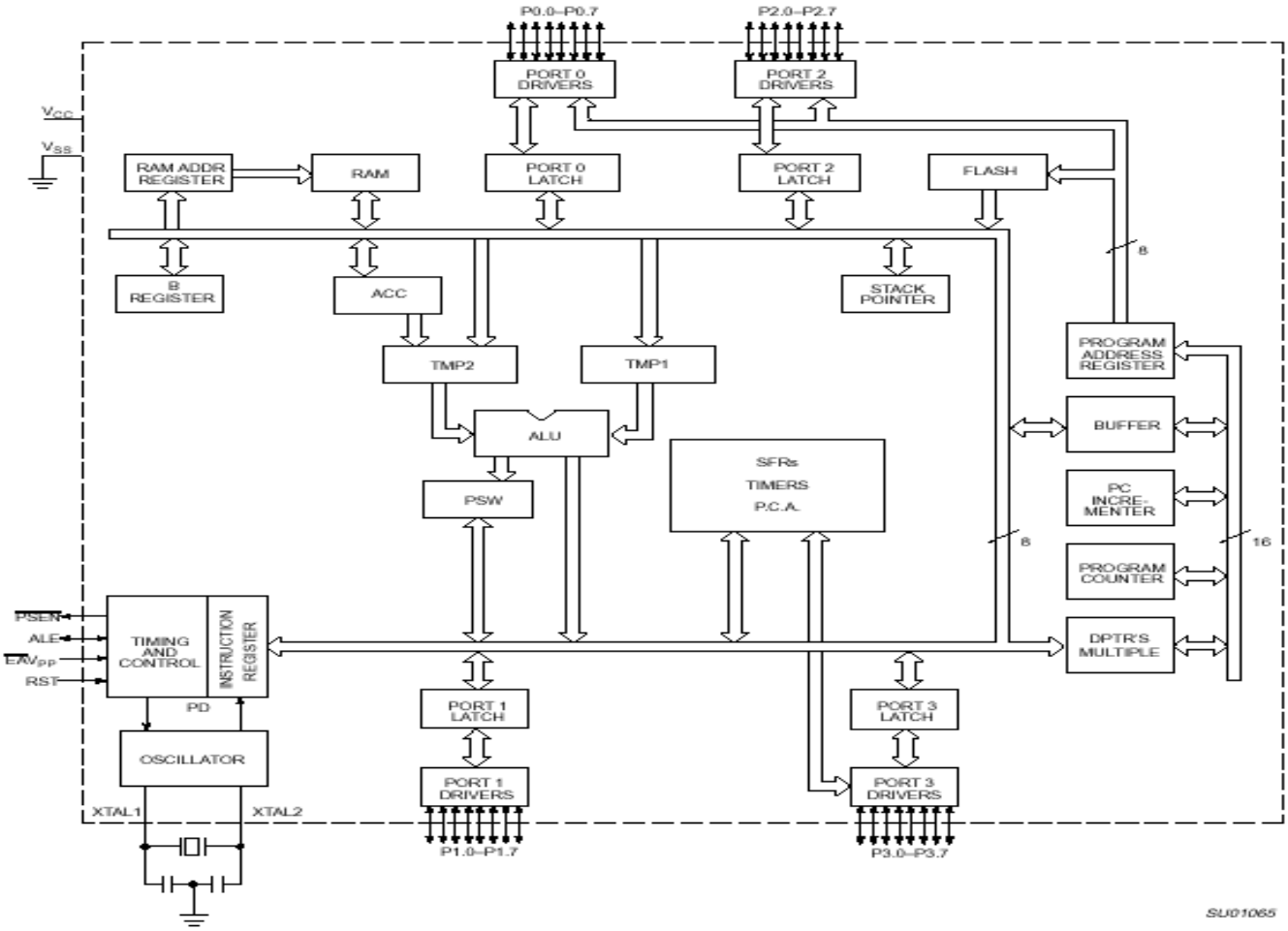


Fig:- Software-Multiplexed LED Display

Unit –IV

8. a) Draw and explain the internal architecture of 8051 microcontroller. CO4 7M



8051 microcontroller is designed by Intel in 1981. It is an 8-bit microcontroller. It is built with 40 pins

- 4K bytes internal ROM
- 128 bytes internal RAM
- Four 8-bit I/O ports (P0 - P3).
- Two 16-bit timers/counters
- One serial interface
- 64k external memory for code
- 64k external memory for data
- only 1 On chip oscillator (external crystal)
- 6 interrupt sources (2 external , 3 internal, Reset)
- 64K external code (program) memory(only read)PSEN
- 64K external data memory(can be read and write) by RD,WR
- Code memory is selectable by EA (internal or external)
- We may have External memory as data and code

b) Draw and explain the Pin diagram of 8051 microcontroller. CO4 7M

8051 microcontroller is a 40 pin Dual Inline Package (DIP). These 40 pins serve different functions like read, write, I/O operations, interrupts etc. 8051 has four I/O ports wherein each port has 8 pins which can be configured as input or output depending upon the logic state of the pins. Therefore, 32 out of these 40 pins are dedicated to I/O ports. The rest of the pins are dedicated to VCC, GND, XTAL1, XTAL2, RST, ALE, EA' and PSEN'.

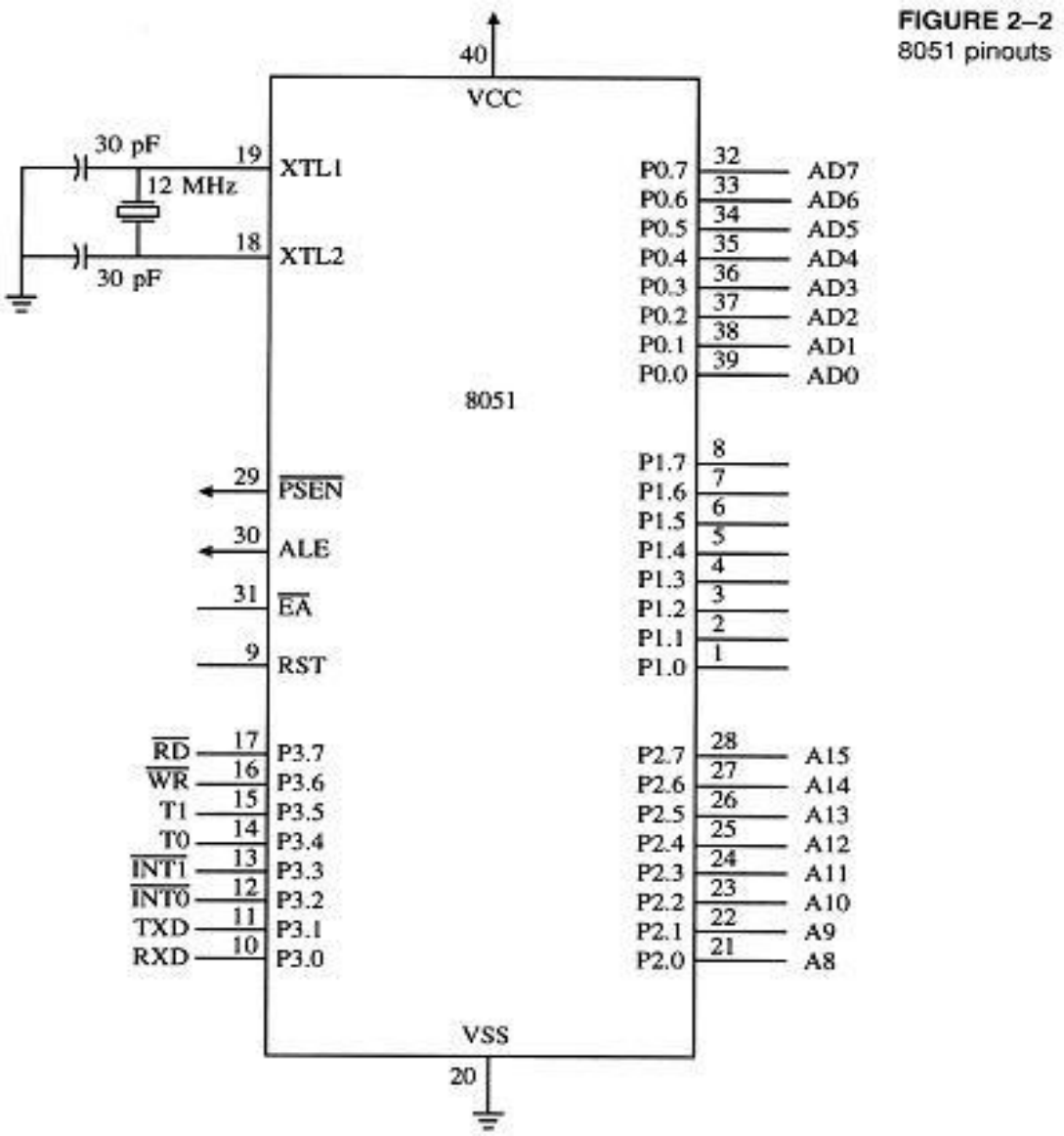


FIGURE 2–2
8051 pinouts

Description of the Pins:-

- **Pin 1 to Pin 8 (Port 1) –**
  Pin 1 to Pin 8 are assigned to Port 1 for simple I/O operations. They can be configured as input or output pins depending on the logic control i.e. if logic zero (0) is applied to the I/O port it will act as an output pin and if logic one (1) is applied the pin will act as an input pin. These pins are also referred to as P1.0 to P1.7 (where P1 indicates that it is a pin in port 1 and the number after '.' tells the pin number i.e. 0 indicates first pin of the port. So, P1.0 means first pin of port 1, P1.1 means second pin of the port 1 and so on). These pins are bidirectional pins.

- **Pin 9 (RST) –**
  Reset pin. It is an active-high, input pin. Therefore if the RST pin is high for a minimum of 2 machine cycles, the microcontroller will reset i.e. it will close and terminate all activities. It is often referred as "power-on-reset" pin because it is used to reset the microcontroller to it's initial values when power is on (high).

- **Pin 10 to Pin 17 (Port 3) –**
  Pin 10 to pin 17 are port 3 pins which are also referred to as P3.0 to P3.7. These pins are similar to port 1 and can be used as universal input or output pins. These pins are bidirectional pins.
  These pins also have some additional functions which are as follows:
  - **P3.0 (RXD) :**
    10th pin is RXD (serial data receive pin) which is for serial input. Through this input signal microcontroller receives data for serial communication.
  - **P3.1 (TXD) :**
    11th pin is TXD (serial data transmit pin) which is serial output pin. Through this output signal microcontroller transmits data for serial communication.
  - **P3.2 and P3.3 (INT0′, INT1′) :**
    12th and 13th pins are for External Hardware Interrupt 0 and Interrupt 1 respectively. When this interrupt is activated(i.e. when it is low), 8051 gets interrupted in whatever it is doing and jumps to the vector value of the interrupt (0003H for INT0 and 0013H for INT1) and starts performing Interrupt Service Routine (ISR) from that vector location.
  - **P3.4 and P3.5 (T0 and T1) :**
    14th and 15th pin are for Timer 0 and Timer 1 external input. They can be connected with 16 bit timer/counter.
  - **P3.6 (WR') :**
    16th pin is for external memory write i.e. writing data to the external memory.
  - **P3.7 (RD') :**
    17th pin is for external memory read i.e. reading data from external memory.

- **Pin 18 and Pin 19 (XTAL2 And XTAL1) –**
  These pins are connected to an external oscillator which is generally a quartz crystal oscillator. They are used to provide an external clock frequency of 4MHz to 30MHz.

- **Pin 20 (GND) –**
  This pin is connected to the ground. It has to be provided with 0V power supply. Hence it is connected to the negative terminal of the power supply.

- **Pin 21 to Pin 28 (Port 2) –**
  Pin 21 to pin 28 are port 2 pins also referred to as P2.0 to P2.7. When additional external memory is interfaced with the 8051 microcontroller, pins of port 2 act as higher-order address bytes. These pins are bidirectional.

- **Pin 29 (PSEN) –**
  PSEN stands for Program Store Enable. It is output, active-low pin. This is used to read external memory. In 8031 based system where external ROM holds the program code, this pin is connected to the OE pin of the ROM.

- **Pin 30 (ALE/ PROG) –**
  ALE stands for Address Latch Enable. It is input, active-high pin. This pin is used to distinguish between memory chips when multiple memory chips are used. It is also used to de-multiplex the multiplexed address and data signals available at port 0.
  During flash programming i.e. Programming of EPROM, this pin acts as program pulse input (PROG).

- **Pin 31 (EA/ VPP) –**
  EA stands for External Access input. It is used to enable/disable external memory interfacing. In 8051, EA is connected to Vcc as it comes with on-chip ROM to store programs. For other family members such as 8031 and 8032 in which there is no on-chip ROM, the EA pin is connected to the GND.

- **Pin 32 to Pin 39 (Port 0) –**
  Pin 32 to pin 39 are port 0 pins also referred to as P0.0 to P0.7. They are bidirectional input/output pins. They don't have any internal pull-ups. Hence, 10 K? pull-up registers are used as external pull-ups. Port 0 is also designated as AD0-AD7 because 8051 multiplexes address and data through port 0 to save pins.

- **Pin 40 (VCC) –**
  This pin provides power supply voltage i.e. +5 Volts to the circuit.

(OR)

9. a) Compare different microcontroller members of 8051 family **CO4** **7M**

Overview of the 8051 Family :

8051 microcontroller was initially designed by Intel Corporation in 1981. Features of 8051 made it extremely popular in market. Because of its popularity and high demand Intel allowed other manufacturers to fabricate and market different variants of 8051 with a condition that all these variants should be code compatible with 8051.

	ROM	RAM	Timer
8051	4k	128	2
8031	-	128	2
8751	4k eprom	128	2
8052	8krom	256	3
8032	-	256	3
8752	8k eprom	256	3

b) Draw the neat circuit diagram of keyboard interfacing with 8051 and explain the programming steps. **CO4** **7M**

Keyboards and LCDs are the most widely used input/output devices of the 8051.

Interfacing the keyboard to the 8051

At the lowest level, keyboards arc organized in a matrix of rows and columns. The CPU accesses both rows and columns through ports; therefore, with two 8-bit ports, an 8 x 8 matrix of keys can be connected to a microprocessor. When a key is pressed, a row and a column make a contact; otherwise, there is no connection between rows and columns.

Scanning and identifying the key:-

Figure 12-6 shows a 4 x 4 matrix connected to two ports. The rows are connected to an output port and the columns are connected to an input port. If no key has been pressed, reading the input port will yield Is for all columns since they are all connected to high (Vee)' If all the rows arc grounded and a key is pressed, one of the columns wi11have 0 since the key pressed provides the path to ground. It is the function of the microcontroller to scan the keyboard continuously to detect and identify the key pressed. How it is done is explained next.

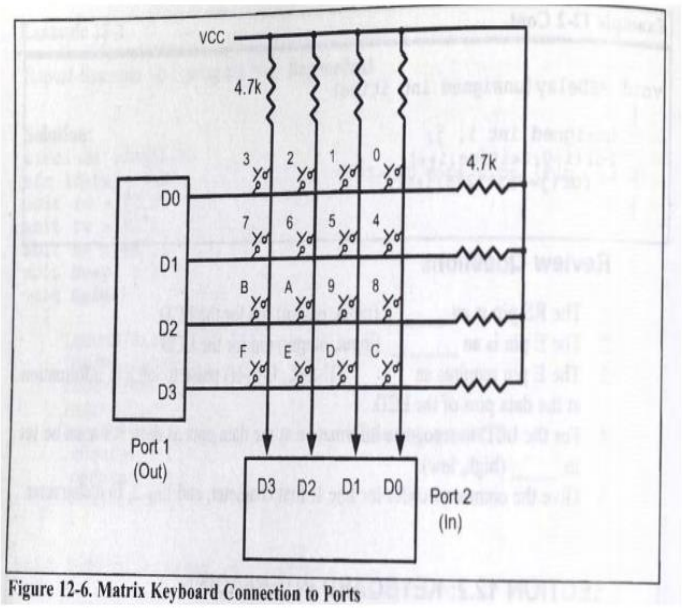


Figure 12-6. Matrix Keyboard Connection to Ports

Figure 12-6. Matrix Keyboard Connection to Ports

Grounding rows and reading the columns:-

To detect a pressed key, the microcontroller grounds all rows by providing o to the output latch, then it reads the columns. If the data read from the columns is 03 - DO = 1111, no key has been pressed and the process continues until a key press is detected, However, if one of the column bits has a zero, this means that a key press has occurred. For example, if 03 - DO = 110 I, this means that a key in the 0 I column has been pressed. After a key press is detected, the microcontroller will go through the process of identifying the key. Starting with the top row, the microcontroller grounds it by providing a low to row DO only; then it reads the columns. If the data read is all 1's, no key in that row is activated and the process is moved to the next row. It grounds the next row, reads the columns, and checks for any zero. This process continues until the row is identified. After identification

Example 12-3

From Figure 12-6, identify the row and column of the pressed key for each of the following,

(a) D3 - DO = 1110 for the row, D3 - DO = 1011 for the column

(b) D3 - DO = 1101 for the row, D3 - DO = O111 for the column

Solution: From Figure 12-6 the row and column can be used to identify the key

(a) The row belongs to DO and the column belongs to D2' therefore key number 2 was pressed.

(b) The row belongs to D1 and the column belongs to D3, therefore key number 7 was pressed.

Scheme prepared by

Dept. Of CSE

Signature of HOD

Dept. Of CSE

SIGNATURE OF EVALUATORS