MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION
(Autonomous)
(ISO/IEC - 27001 - 2005 Certified)

MODEL ANSWER

SUMMER– 17 EXAMINATION

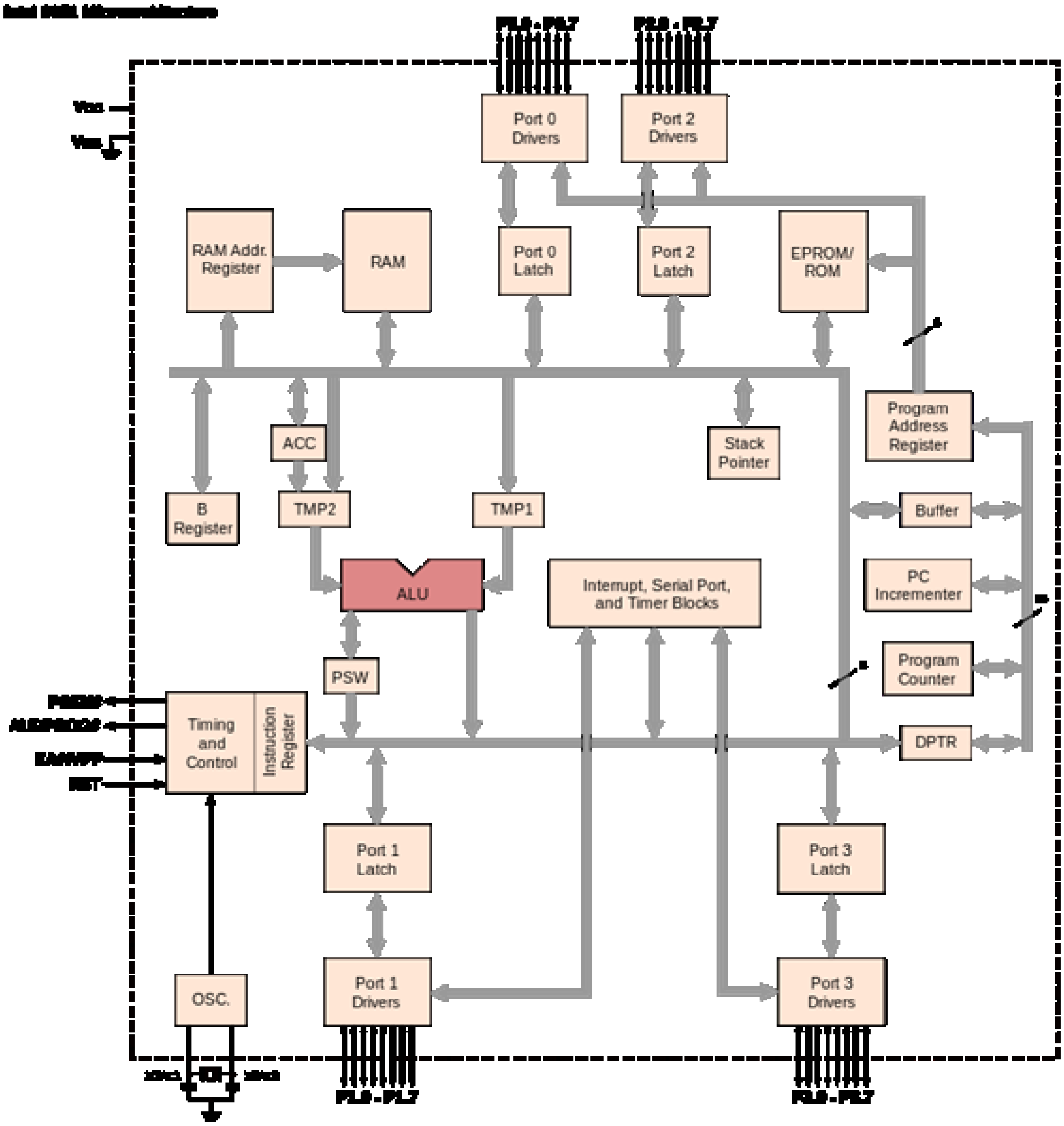
Subject Title: EMBEDDED SYSTEM **Subject Code:** 17626

Important Instructions to examiners:

1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling errors should not be given more Importance (Not applicable for subject English and Communication Skills.
4) While assessing figures, examiner may give credit for principal components indicated in the figure. The figures drawn by candidate and model answer may vary. The examiner may give credit for any equivalent figure drawn.
5) Credits may be given step wise for numerical problems. In some cases, the assumed constant values may vary and there may be some difference in the candidate's answers and model answer.
6) In case of some questions credit may be given by judgement on part of examiner of relevant answer based on candidate's understanding.
7) For programming language papers, credit may be given to any other program based on equivalent concept.

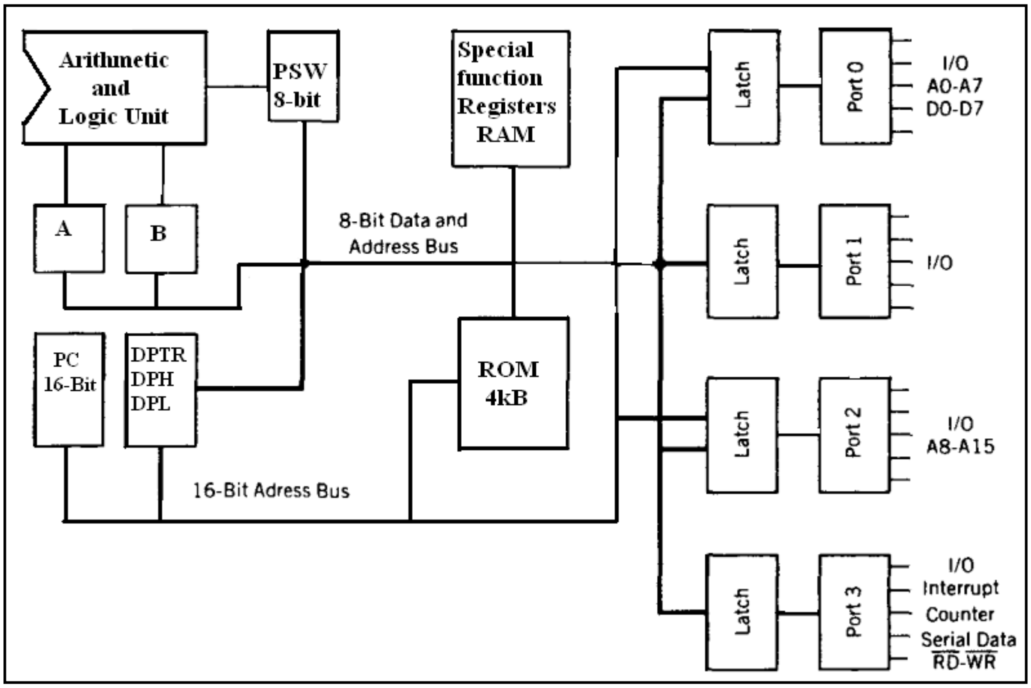
Q. No	Sub Q. N.	Answer	Marking Scheme
1.	**A)**	**Attempt any three of the following:**	**12 Marks**
	i)	**Draw the architecture of 8051.**	**4 M**

Ans:



OR

(Correct Diagram: 4 marks)

		 Arithmetic and Logic Unit PSW 8-bit Special function Registers RAM A B 8-Bit Data and Address Bus PC 16-Bit DPTR DPH DPL ROM 4kB 16-Bit Adress Bus Latch Latch Latch Latch Port 0 Port 1 Port 2 Port 3 I/O A0-A7 D0-D7 I/O I/O A8-A15 I/O Interrupt Counter Serial Data RD-WR	
	ii)	**Identify the addressing mode used in following instruction.** **a) MOV A, #55H b) ADD A, B** **c) MOV @ Ri, 35H d) MOVC A,@ A+ DPTR**	**4 M**
	Ans:	**a) Immediate Mode** **b) Registers Direct Addressing Mode** **c) Register Indirect Address Mode** **d) Indexed Addressing Mode**	**(Each answer: 1 mark)**
	iii)	**Enlist any 8 features of 8051 µC.**	**4 M**
	Ans:	1. 4 KB on chip program memory (ROM or EPROM)). 2. 128 bytes on chip data memory (RAM). 3. 8-bit data bus 4. 16-bit address bus 5. 32 general purpose registers each of 8 bits 6. Two -16 bit timers T_0 and T_1 7. Five Interrupts (3 internal and 2 external).	**(Any 8 features: ½ Mark each)**

8. Four Parallel ports each of 8-bits (PORT0, PORT1, PORT2, PORT3) with a total of 32 I/O lines.
9. One 16-bit program counter and One 16-bit DPTR (data pointer)
10. One 8-bit stack pointer
11. One Microsecond instruction cycle with 12 MHz Crystal.
12. One full duplex serial communication port.

iv) **State difference between microprocessor and microcontroller (4 points).** **4 M**

Ans:

S.No	Microprocessor	Microcontroller
1	A microprocessor is a general purpose device which is called a CPU	A microcontroller is a dedicated chip which is also called single chip computer.
2	A microprocessor do not contain on chip I/O Ports, Timers, Memories etc..	A microcontroller includes RAM, ROM, serial and parallel interface, timers, interrupt circuitry (in addition to CPU) in a single chip.
3	Microprocessors are most commonly used as the CPU in microcomputer systems	Microcontrollers are used in small, minimum component designs performing control-oriented applications.
4	Microprocessor instructions are mainly nibble or byte addressable	Microcontroller instructions are both bit addressable as well as byte addressable.
5	Microprocessor instruction sets are mainly intended for catering to large volumes of data.	Microcontrollers have instruction sets catering to the control of inputs and outputs.
6	Microprocessor based system design is complex and expensive	Microcontroller based system design is rather simple and cost effective
7	The Instruction set of microprocessor is complex with large number of instructions.	The instruction set of a Microcontroller is very simple with less number of instructions. For, ex: PIC microcontrollers have only 35 instructions**.**
8	A microprocessor has zero status flag	A microcontroller has no zero flag.

(Any correct 4 point , each: 1 mark)

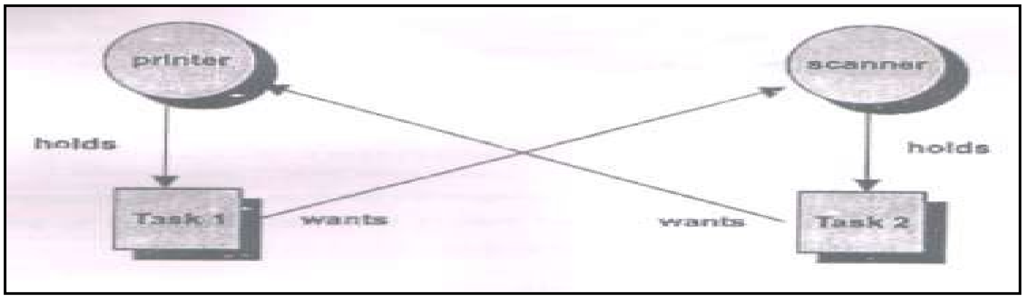
9	Ex 8085,8086	Ex 8051, 8751

B) **Attempt any one of the following:** **6 Marks**

i) **Explain deadlock. How it can be avoided?** **6 M**

Ans:

(Explanation of dead lock : 4 marks , how to avoid : 2 marks)



Deadlock is the situation in which multiple concurrent threads of execution in a system are blocked permanently because of resources requirement that can never be satisfied.

A typical real-time system has multiple types of resources and multiple concurrent threads of execution contending for these resources. Each thread of execution can acquire multiple resources of various types throughout its lifetime.

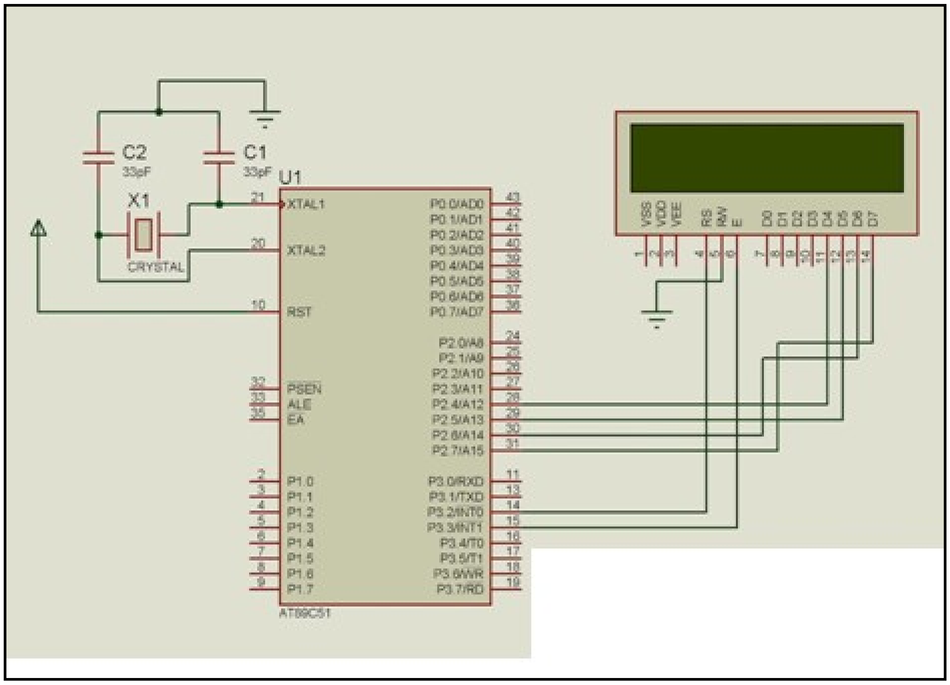
Potential for deadlock exist in a system in which the underlying RTOS permits resources sharing among multiple threads of execution In this example, task #1 wants the scanner while holding the printer. Task #1 cannot proceed until both the printer and the scanner are in its possession.

Task #2 wants the printer while holding the scanner. Task #2 cannot continue until it has the printer and the scanner. Because neither task #1 nor task#2 is willing to give up what it already has, the two tasks are now deadlocked because neither can continue

How to avoid a deadlock is for threads to:

- Acquire all resources before proceeding
- Acquire the resources in the same order
- Release the resource in the revere order

	ii)	**Draw the labeled diagram to interface 16x2 LCD to microcontroller 8051 and state the function of RS, R/W and EN pin of 16x2 LCD.**	**6M**
	Ans:		**(Diagram : 3 marks , Explanation of each pin 1 mark)**



Explanation of RS, R/W and EN RS:

RS: is used to make the selection between data and command register. RS=0, command register is selected RS=1 data register is selected.

RW: R/W gives you the choice between writing and reading. R/W=1, reading is enabled. R/W=0 then writing is enabled.

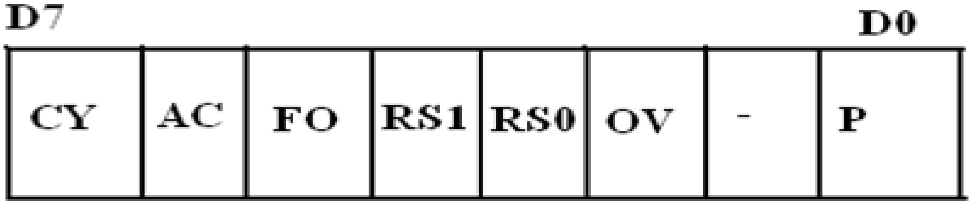
EN: Enable pins is used by the LCD to latch information presented to its data pins. When data is supplied to data pins, a high to low pulse must be applied to this pin in-order for the LCD to latch in the data present at the data pins.

2.		**Attempt any four of the following:**	**16 Marks**
	i)	**Draw PSW of 8051 and state function of each bit.**	**4M**

Ans:

(PSW format :2 marks , Explanation : 2 marks)

PSW format



The meaning of various bits of PSW register is shown below.

CY	PSW.7	Carry Flag
AC	PSW.6	Auxiliary Carry Flag
FO	PSW.5	Flag 0 available for general purpose.
RS1	PSW.4	Register Bank select bit 1
RS0	PSW.3	Register bank select bit 0
OV	PSW.2	Overflow flag
---	PSW.1	User difinable flag
P	PSW.0	Parity flag .set/cleared by hardware.

- The bits PSW3 and PSW4 are denoted as RS0 and RS1 and these bits are used the select the bank registers of the RAM location

The selection of the register Banks and their addresses are given below.

RS1	**RS0**	**Register Bank**	**Address**
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

		• **CY, the carry flag** This flag is set whenever there is a carry out from the D7 bit. This flag bit is affected after an 8-bit addition or subtraction. It can also be set to 1 or 0 directly by an instruction such as “SETB C” and “CLR C” where “SETB C” stands for “set bit carry” and “CLR C” for “clear carry”. • **AC, the auxiliary carry flag** If there is a carry from D3 to D4 during an ADD or SUB operation, this bit is set; otherwise, it is cleared. This flag is used by instructions that perform BCD (binary coded decimal) arithmetic. • **P, the parity flag** The parity flag reflects the number of 1 s in the A (accumulator) register only. If the A register contains an odd number of Is, then P = 1. Therefore, P = 0 if A has an even number of 1s. • **OV, the overflow flag** This flag is set whenever the result of a signed number operation is too large, causing the high-order bit to overflow into the sign bit. In general, the carry flag is used to detect errors in unsigned arithmetic operations. The overflow flag is only used to detect errors in signed arithmetic operations **FO Flag 0** This is PSW.5 bit and which is used for general purpose	
	ii)	**Enlist any four addressing modes with one example each.**	**4M**
	Ans:	1) Immediate addressing mode 2) Direct addressing mode 3) Register direct addressing mode 4) Register indirect addressing mode 5) Indexed addressing mode	**({**Note : any other correct example can write **} List :2 mark ,example ½ Mark each list)**

		Example: 1) Immediate addressing mode MOV A, #6AH Ex 2 MOV R1,#55 H 2) Direct addressing mode MOV A, 04H Ex2 MOV P1 , A 3) Register direct addressing mode MOV A, R4 Ex2 ADD A, R7 4) Register Indirect address mode MOV A, @R0 EX2 MOV B,@R1 5) Indexed addressing mode. MOVC A, @A+DPTR EX2 MOVC A, @A+PC	
	iii)	**State function of pin $\overline{\text{EA}}$, $\overline{\text{PSEN}}$, RESET, $\overline{\text{ALE}}$.**	**4 M**
	Ans:	**Function of $\overline{\text{PSEN}}$:** 1. PSEN stands for ― program store enable. The read strobe for external Program Memory is the signal PSEN (Program Store Enable). In an 8031-based system in which an external ROM holds the program code, this pin is connected to the OE pin of the ROM. 2. In other words, to access external ROM containing program code, the 8031/51 uses the PSEN signal. This read strobe is used for all external program fetches. PSEN is not activated for internal fetches. **Function of $\overline{\text{EA}}$:** 1. EA which stands for external access is pin number 31 in the DIP packages. It is an input pin and must be connected to either Vcc or GND. In other words, it cannot be left unconnected. 2. The lowest 4K (or SK or 16K) bytes of Program Memory can be either in the on-chip ROM or in an external ROM. This selection is made by strapping the EA (External Access) pin to either VCC or Vss.	**(Each 1 mark)**

3. In the 4K byte ROM devices, if the pin is strapped to Vcc, then program fetches to addresses 0000H through OFFFH are directed to the internal ROM. Program fetches to addresses 1000H through FFFFH are directed to external ROM.
4. If the pin is strapped to Vss, then all program fetches are directed to external ROM. The ROM less parts must have this pin externally strapped to VSS to enable them to execute properly.

Function of $\overline{\text{ALE}}$:

1. ALE stands for address latch enable. It is an output pin and is active high for latching the low byte of address during accesses to external memory.
2. The ALE pin is used for demultiplexing the address and data by connecting to the G pin of the 74LS373 chip.

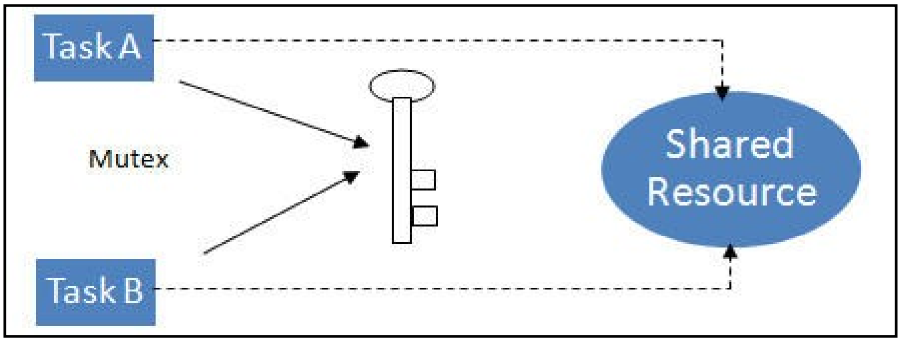
Function of RESET:

1. Pin 9 is the RESET pin. It is an input and is active high (normally low). Upon applying a high pulse to this pin, the microcontroller will reset and terminate all activities.
2. This is often referred to as a power-on reset. Activating a power-on reset will cause all values in the registers to be lost. It will set program counter to all 0s.
3. In order for the RESET input to be effective, it must have a minimum duration of two machine cycles. In other words, the high pulse must be high for a minimum of two machine cycles before it is allowed to go low.

If external RAM & ROM is not accessed, then ALE is activated at constant rate of 1/6 oscillator frequency, which can be used as a clock pulses for driving external devices.

	iv)	**State the function of following 8051 instruction** **a) MOV C A, @A + DPTR** **b) SWAPA** **c) MOV 80H, 90H** **d) MUL AB**	**4 M**
	Ans:	a) **MOVC A, @A+DPTR**: this instruction copies value at the location given by result of addition of content of accumulator and 16 bit register DPTR. **Example**: Suppose Accumulator contain '01H' and DPTR contains value '1000H'	**(Each 1 mark (Note: student can write any**

		then after the execution of instruction 'MOVC A, @A+DPTR the value at address '1001H' will be transferred to accumulator. **b) SWAP A - Swaps nibbles within the accumulator** A: accumulator Description: A nibble refers to a group of 4 bits within one register (bit0-bit3 and bit4-bit7). This instruction interchanges high and low nibbles of the accumulator. **Example: SWAP A** Before execution: A=E1h (11100001)bin. After execution: A=1Eh (00011110)bin. **c) MOV80h,90H**: This instruction copy the content of 90h memory address location at memory address location 80h **Example :** if at 90h location 0Ah data is stored then after execution of this instruction , 0Ah copied at location 80H **d) MUL AB** - Multiplies A and B Description: Instruction multiplies the value in the accumulator with the value in the B register. The low-order byte of the 16-bit result is stored in the accumulator, while the high byte remains in the B register. If the result is larger than 255, the overflow flag is set. The carry flag is not affected. **Example:** **MUL AB** Before execution: A=80 (50h) B=160 (A0h) After execution: A=0 B=32h A·B=80·160=12800 (3200h)	**other example)**
	v)	**Explain Task Synchronization. How it is achieved?**	**4 M**
	Ans:	**Task Synchronization** Synchronization is essential for tasks to share mutually exclusive resources (devices, buffers, etc) and/or allow multiple concurrent tasks to be executed (e.g. Task A needs a result from task B, so task A can only run till task B produces it).	**(Note : Any other example of task synchronization can write)** **(Task synchronization : 2marks ,How**



Achieve : 2 marks)

Task synchronization is achieved using two types of mechanisms:

- **Event Objects:**

Event objects are used when task synchronization is required without resource sharing. They allow one or more tasks to keep waiting for a specified event to occur. Event object can exist either in triggered or non-triggered state. Triggered state indicates resumption of the task.

- **Semaphores:**

A semaphore has an associated resource count and a wait queue. The resource count indicates availability of resource. The wait queue manages the tasks waiting for resources from the semaphore. A semaphore functions like a key that define whether a task has the access to the resource. A task gets an access to the resource when it acquires the semaphore.

There are three types of semaphore:

- Binary Semaphores
- Counting Semaphores
- **Mut**ually **Ex**clusion(Mutex) Semaphores

Semaphore functionality (Mutex) represented pictorially is shown the figure

vi) **Write ‘C’ or assembly language program for 8051 to transfer letter ‘M’ serially at 4800 baud rate.** **4 M**

Ans:

28800 is the maximum baud rate of the 8051 microcontroller

28800/4800= 6

That baud rate ‘6’ is stored in the timers

(Correct program in ALP or C language 4

marks

Assembly language program)

```
ORG 0000H

MOV SCON, #50H          ;serial port mode 1

;configure timer 1 in auto-reload mode for 4800 baud

MOV TMOD, #20H               ;timer mode 2

MOV TH1, #-6                ;reload value for4800 baud

SETB TR1                     ;start timer1

LOOP:   MOV SBUF, #'M'    ;transmit character

WAIT: JNB TI, WAIT ; wait for end of transmission

CLR TI      clear TI

CLR TR1   stop timer1

JMP LOOP      ;re-transmit

END
```

OR   c program

28800 is the maximum baud rate of the 8051 microcontroller

28800/4800= 6

That baud rate ‘6’ is stored in the timers

```
#include<reg51.h>
void main()
{
SCON=0×50; //start the serial communication on port mode1
TMOD=0×20; //selected the timer mode//
TH1=- 6; // load the baud rate
TR1=1; //Timer ON
SBUF='M'; // transmit character
while(TI==0); //wait for end of transmission
TI=0;   // clear  TI
TR1=0; //OFF the timer
while(1);    //continuous loop
}
```

3. **Attempt any four of the following:** **16Marks**

i) **Draw and describe the format of IE SFR of 8051.** **4M**

Ans:

(Format:2 marks, Explanation: 2 marks)

Interrupt Enable register (IE): Byte Address: A8H , bit address A8H to AFH

EA	-	ET2	ES	ET1	EX1	ET0	EX0

EX0 : External interrupt0 ($\overline{INT0}$) enable bit

ET0: Timer-0 interrupt enable bit

EX1 : External interrupt1 ($\overline{INT1}$) enable bit

ET1 : Timer-1 interrupt enable bit

ES : Serial port interrupt enable bit

ET2 : Timer-2 interrupt enable bit, not for 8051, reserved for future use

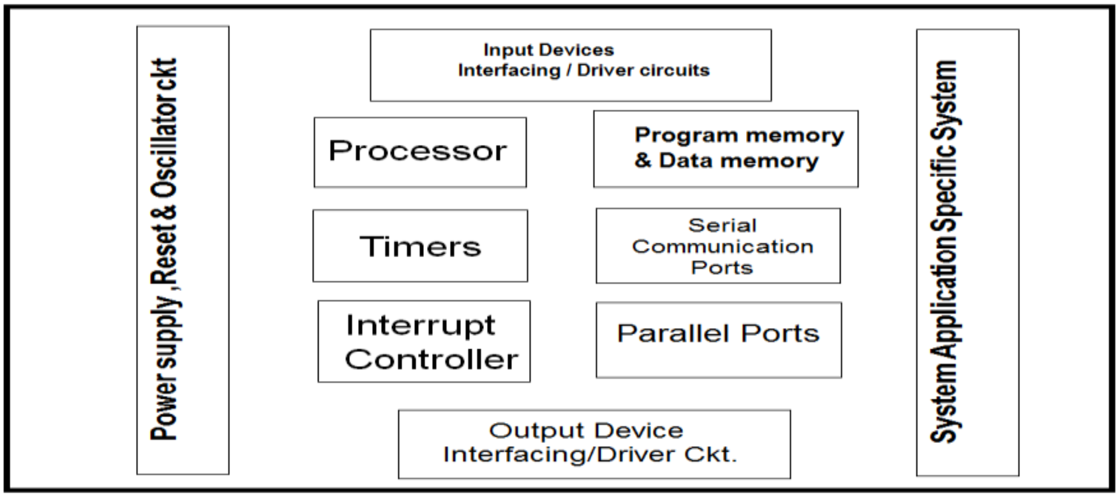
EA : Enable All bit

When EA bit is 0, it is called as global disable i.e. all the maskable interrupts are disabled. And when EA is 1, it enables those interrupts which have their bit set in IE register. i.e. when EA and ET1 is set, and all other bits are reset, this enables the timer1 interrupt.
Bit0 to bit5 i.e. all the bits except EA bit are the local enable/ disable bit. When the bit is zero then the respective interrupt is disabled. And when the bit is set and EA is also set, then the respective interrupt is enabled. But if interrupt bit is set and EA=0, all the interrupts are disabled

ii) Draw the block diagram of embedded system. Explain various hardware units. **4M**

Ans:

(Block diagram: 2 marks, Any Four Hardware units Explanation : 1/2 marks each)



1. Embedded processor:

It is the heart of the embedded system. It has two essential units: control unit and execution unit. Control unit fetches instructions from memory and execution unit includes ALU and circuits to perform execution of the instructions for a program control task

2. Power supply, reset & oscillator circuit:

- Most of the systems have their own power supply. Some embedded system do not have their own power supply. These embedded systems are powered by external power supply e.g. USB based embedded system, network interface card, Graphics Accelerator etc. are powered by PC power supply.
- Reset means that processor begins processing of instructions from starting address set by default in program counter on power up.
- The clock circuit controls execution time of instructions, CPU machine cycles.

3.Timers:
Timer circuit is suitably configured as system clock or RTC (Real time clock). To schedule various tasks and for real time programming an RTC (Real Time Clock), or system clock is needed.

4. Program & data memory:
In embedded system, secondary memory like disk is avoided. Most of the embedded processors have internal memory such as ROM, RAM, flash/EEPROM, EPROM/PROM for storing program and data.

5. Interrupt controller:
It is an interrupt handling mechanism which must exist in embedded system to handle interrupts from various processes and for handling multiple interrupts simultaneously pending for service.

6. I/O ports:
I/O ports are used to interface external devices like sensors, key buttons, transducers, LEDs, LCD actuators, alarms, motors, values, printer etc. There are two types of ports ,parallel and serial port. The parallel ports are used in short distance communication while serial ports are used in long distance communication.

7.Input& output device interfacing/driver circuits:
Some I/O devices like motors, actuators, valves, sensors are not compatible with the processor. Hence the I/O interface circuits are designed to drive such input and output devices interfaced to the embedded processor

8.System Application specific circuits
These are the circuits that can control specific target circuits. They consist of ADC,DAC, relays, sensors etc.

iii) **List the interrupts used in 8051. Give their priorities and vector address.** **4 M**

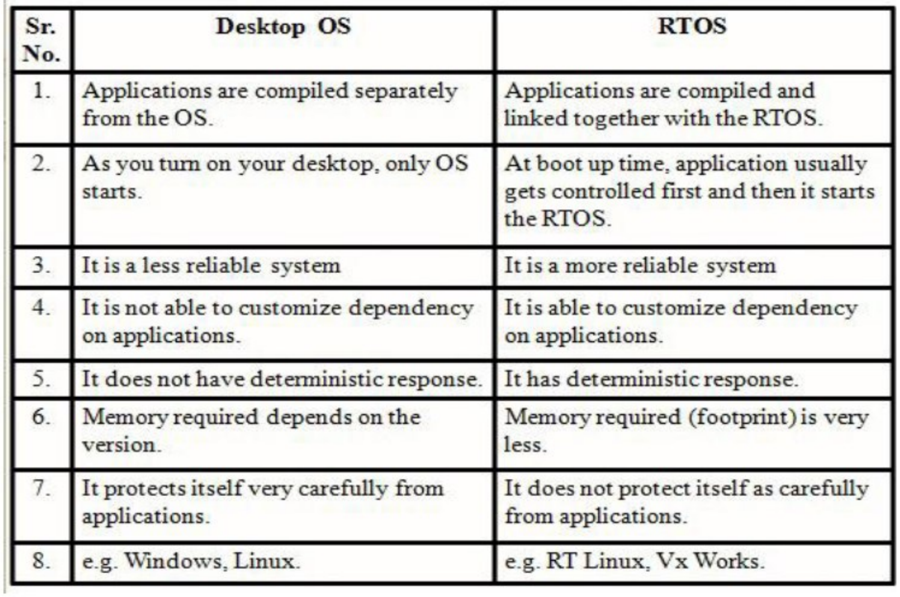
Ans:

Source	Priority	Vector Location
IE0	1 (Highest)	0003H
TF0	2	000BH
IE1	3	0013H
TF1	4	001BH
RI+TI	5 (lowest)	0023H

(Listing the interrupts: 1 mark, Priorities: 1 mark, Vector location: 2 marks)

Interrupt	Vector Address	Priority
IE0 / (External interrupt 0 , INT0)	0003H	1
TF0 / (Timer 0 Interrupt)	000BH	2
IE1 / (External interrupt 1 , INT1)	0013H	3
TF1 / (Timer 1 Interrupt)	001BH	4
TI or RI (serial port interrupt)	0023H	5

	iv)	**Explain the features of RTOS. State how it differs from general operating system.**	**4 M**
	Ans:	**Features of RTOS:** **1. Multithreading & pre-emptibility:** The scheduler should be able to preempt any task in the system and allocate the resource to the thread that needs it most even at peak load. **2. Thread Priority:** All tasks are assigned priority level to facilitate the preemption. **3. Inter task communication and synchronization:** Multiple tasks pass information among each other in a timely fashion and ensuring data integrity **4. Priority inheritance:** RTOS should have large number of priority levels and should prevent priority inversion using priority inheritance **5. Short latencies:** The latencies are short and predefined. Task switching latency, interrupt latency, interrupt dispatch latency **6. Control to memory management:** To ensure predictable response to an interrupt, an RTOS should provide way for task to lock its code and data into real memory.	**(Any four Features - 2 marks, Difference any two points - 2 marks)**

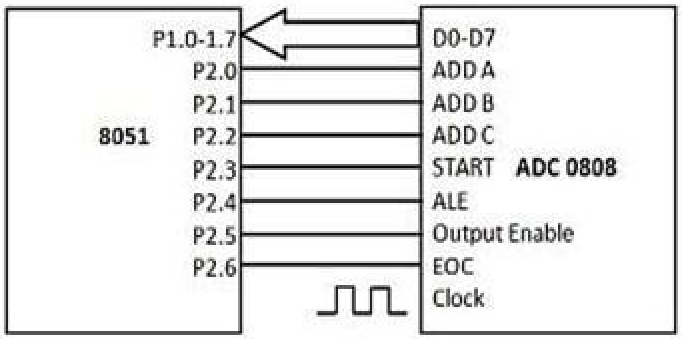


Sr. No.	Desktop OS	RTOS
1.	Applications are compiled separately from the OS.	Applications are compiled and linked together with the RTOS.
2.	As you turn on your desktop, only OS starts.	At boot up time, application usually gets controlled first and then it starts the RTOS.
3.	It is a less reliable system	It is a more reliable system
4.	It is not able to customize dependency on applications.	It is able to customize dependency on applications.
5.	It does not have deterministic response.	It has deterministic response.
6.	Memory required depends on the version.	Memory required (footprint) is very less.
7.	It protects itself very carefully from applications.	It does not protect itself as carefully from applications.
8.	e.g. Windows, Linux.	e.g. RT Linux, Vx Works.

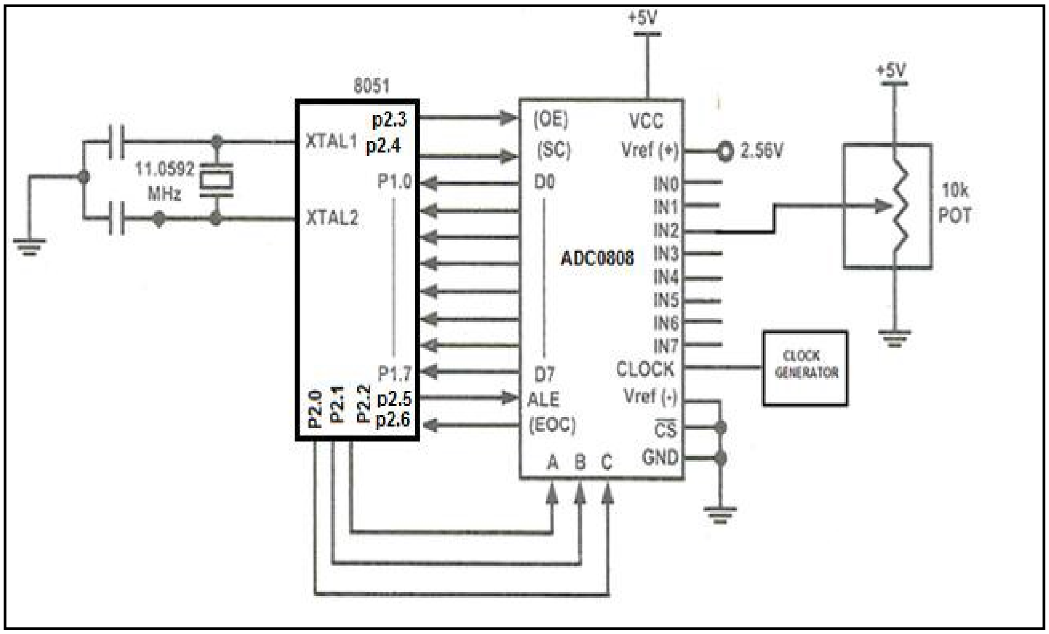
v) **Draw labelled diagram to interface analog to digital converter ADC 0808 to 8051.** **4 M**

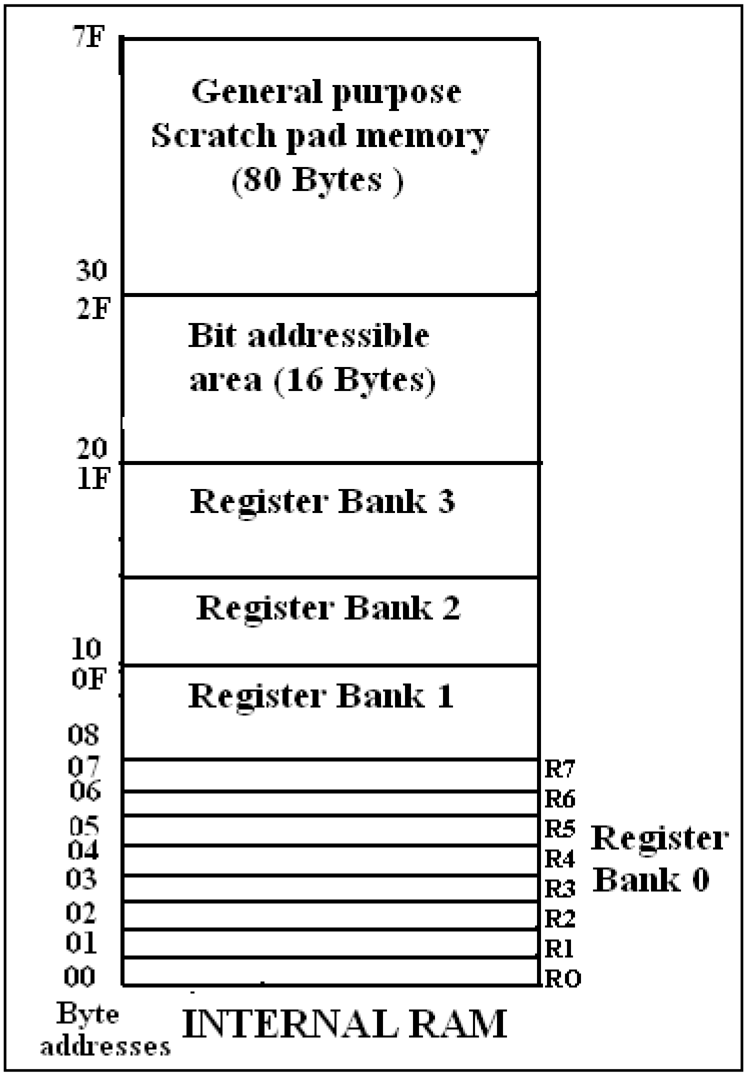
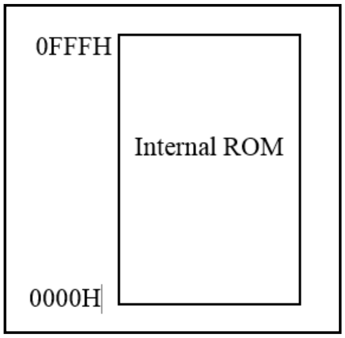
Ans:

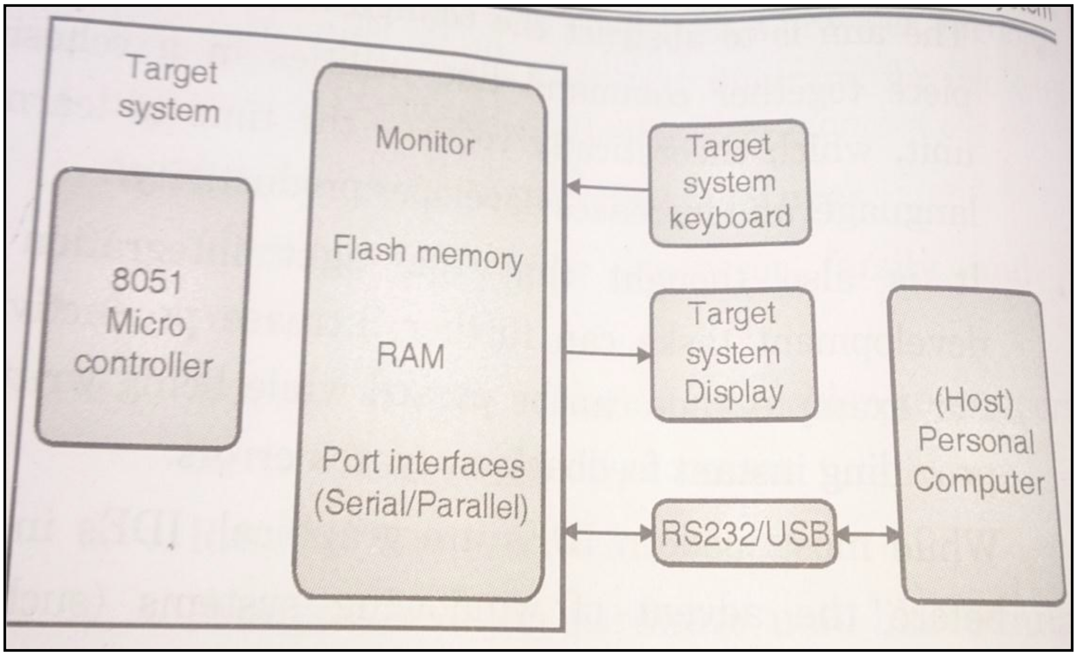
(Neat labelled diagram: 4 marks)

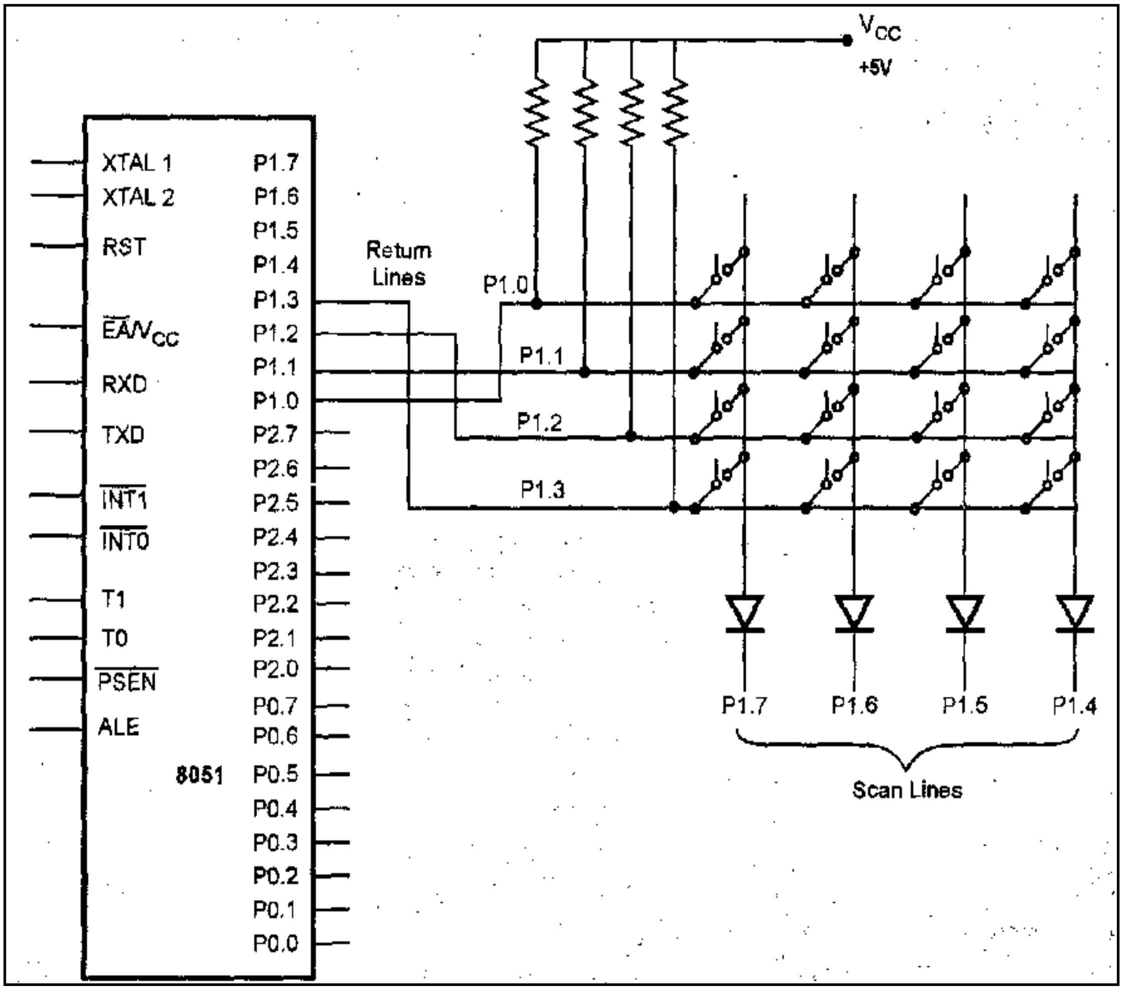


OR

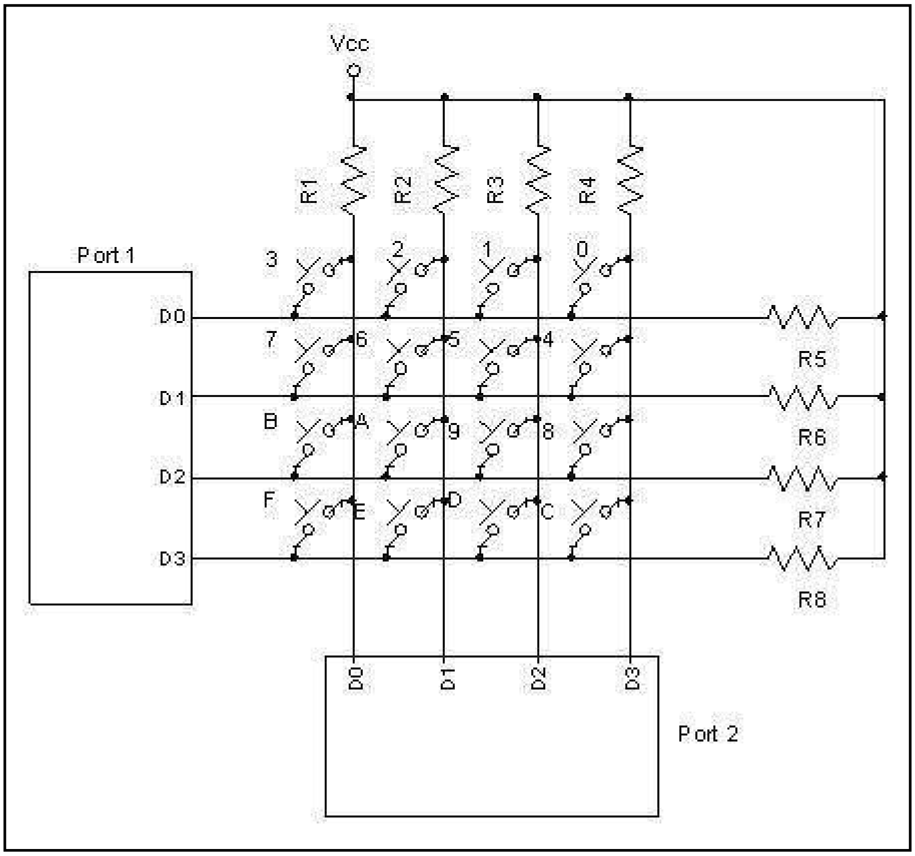
		 +5V 8051 XTAL1 XTAL2 11.0592 MHz p2.3 p2.4 P1.0 P1.7 P2.0 P2.1 P2.2 p2.5 p2.6 (OE) (SC) D0 D7 ALE (EOC) A B C VCC Vref (+) 2.56V ADC0808 IN0 IN1 IN2 IN3 IN4 IN5 IN6 IN7 CLOCK Vref (-) CS GND +5V 10k POT CLOCK GENERATOR	
	vi)	**Explain the starvation with example.**	**4 M**
	Ans:	**Starvation:** Starvation: Starvation is a resource management problem where a process does not get the resources it needs for a long time because the resources are being allocated to other processes. Starvation generally occurs in a Priority based scheduling System. Where High Priority (Lower Number = Higher Priority) requests get processed first. Thus a request with least priority may never be processed. **For example**: Consider priority based scheduling with scheduling criteria as the shortest process first. In a ready queue where all the processes are waiting for CPU, shortest process will be selected first. If there is a continuous supply of shortest process then the longer process may never get scheduled on CPU which leads to its starvation.	**(Starvation:2 marks, Example:2 marks)**

4.	**A)**	**Attempt any three of the following:**	**12 Marks**
	i)	**Draw internal RAM and ROM memory organization of 8051 microcontroller.**	**4M**
	Ans:	**Internal RAM Diagram:**  7F General purpose Scratch pad memory (80 Bytes) 30 2F Bit addressible area (16 Bytes) 20 1F Register Bank 3 Register Bank 2 10 0F Register Bank 1 08 07 06 05 04 03 02 01 00 R7 R6 R5 R4 R3 R2 R1 RO Register Bank 0 Byte addresses INTERNAL RAM  	**(Internal RAM: 2 marks, Internal ROM: 2 marks)**

	ii)	**Explain in brief:** **a) Device programmer** **b) Target board.**	**4M**
	Ans:	**a) Device programmer:** Also called as laboratory programmer, a programming system for a application device such as EPROM/ROM or Flash memory or microcontroller memory, PLA. The device to be programmed is inserted into the socket at the device programmer and burns the code using software at the host. i.e. personal computer through serial port. The device programmer software running on the host uses an input file from the locator software output records reflects the final design which has the bootstrap loader and compressed records which the processor decompresses before the embedded system processor starts the execution. **b) Target board** Target board or machine or system consists of- 1) A microprocessor or microcontroller, 2) ROM-memory of image of embedded system, 3) RAM- memory for implementation of stack, temporary variables and memory buffers Peripheral devices and interfaces such as RS 232,10/100 base ethernet, parallel ports, USB etc. **Example:** A simple sample target system is as shown  Target system 8051 Micro controller Monitor Flash memory RAM Port interfaces (Serial/Parallel) Target system keyboard Target system Display RS232/USB (Host) Personal Computer The target board differs from the final system as it interfaces with personal computers as well as work as a standalone system which requires a repeated downloading of the codes during development phase in the flash memory. Also requires repeated modification, testing, simulation, debugging till it works according to final specifications. Once done with, the code is downloaded in ROM (instead of flash memory) in the target system.	**(Device programmer :2 marks, Target board-2 marks)**

<table>
<tr><td></td><td><b>iii)</b></td><td><b>Draw labelled diagram to interface 4x4 keyboard to 8051.</b></td><td></td></tr>
<tr><td></td><td></td><td>

<b>OR</b></td><td><b>(Correct labeled Diagram: 4 marks)</b></td></tr>
</table>

		 Vcc R1 R2 R3 R4 Port 1 3 2 1 0 D0 7 6 5 4 R5 D1 B A 9 8 R6 D2 F E D C R7 D3 R8 D0 D1 D2 D3 Port 2	
	iv)	**State various software tools available in IDE. Explain any one in brief.**	
		Software tools • Compiler • Cross assembler • Cross compiler • Locators • Loaders • Simulators • Debugger • Integrated development environment (IDE) **Explanation (any one of the following)** **Compiler:** It is a computer program that transforms the source code written in a programming or source language into another computer language i.e. target language i.e. binary code known as object code.	**(State Software tools: 2 marks, Any one Explanation :2 marks)**

Cross assembler:
It is useful to convert object codes for microcontrollers or processor to other codes for another microcontrollers or processor and vice versa.

Cross compiler:
It is used to create executable code other than one on which the compiler is run. They are used to generate executables for embedded systems or multiple platforms.

Linker/Locator:
It is used for relocation process.
It is done during compilation also it can be done at run time by a relocating loader.
It is a program that takes one or more objects generated by compiler and combines them into a single executable program.

Simulators:
A simulator is the s/w that simulates a h/w unit like emulator, peripheral, network and I/O devices on a PC

- It defines a processor or processing device as well as various versions for the target system
- Monitors the detailed information of as source code part with labels and symbols during the execution for each single step.
- Provides the detailed information of the status of memory RAM and simulated ports, simulated peripheral devices of the defined target system

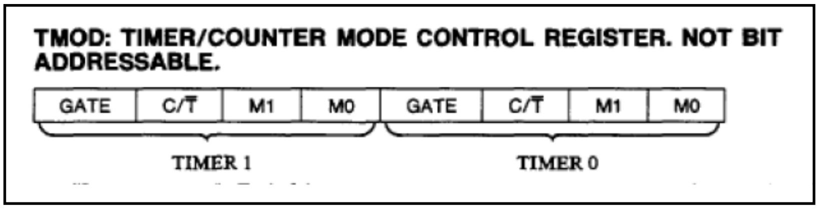
Integrated Development Environment (IDE) :

- It supports for defining a processor family and its version
- Support a user definable assembler to support a new version or a type of processor.
- Provides multiuser environment
- Supports conditional and unconditional break points
- Provide debugger.

B) **Attempt any one of the following:** **6 Marks**

i) **Draw the format of TMOD SFR. Explain Timer modes with diagram.** **6 M**

Ans:


TMOD: TIMER/COUNTER MODE CONTROL REGISTER. NOT BIT ADDRESSABLE.
GATE
C/T̄
M1
M0
GATE
C/T̄
M1
M0
TIMER 1
TIMER 0

(Format of TMOD: 2 marks, explanation of four timer modes with diagram : 1 mark each)

Timer modes:

(Note: if only timer modes are listed, 1 mark can be given)

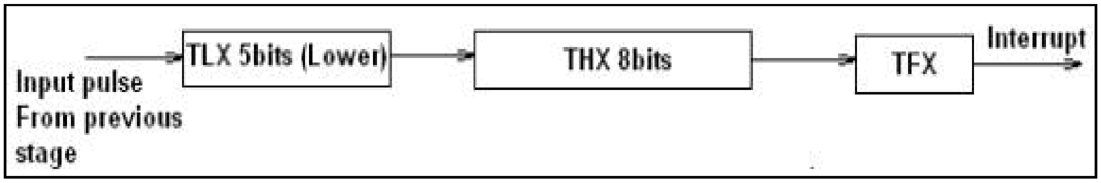
(Note: if only mode diagrams are given: ½ mark for each mode = 2 marks can be given)

M1	M0	Mode	Operating mode
0	0	0	13-bit mode
0	1	1	16-bit mode
1	0	2	8-bit auto-reload
1	1	3	8-bit timers using timer0

Mode 0: 13-bit mode for Timer0 & Timer1
It is **selected by making M1, M0 in TMOD = 00**
This is the **13-bit mode** for both the timers. The 13-bit count is placed in corresponding THx and TLx registers. Mode 0 uses all the 8-bits of THx register and lower 5-bits (bit0 to bit4) of TLx register to hold the 13-bit count. The timer starts working after TRx bit is set in TCON. After starting the timer, the 13-bit count is incremented by one per machine cycle, this is referred as timer operation. If the external clock on pins T0 or T1 is used, the count in registers is incremented by one for every cycle of external clock. Then the timer is working as counter.

The timer is said to overflow when the count in registers TH and TL rolls over to 2000H. This overflow takes after the value 1FFFH (8191d). So maximum count for timer mode 0 is 1FFFH (8191d).

Then the count that should be placed in timer registers
= [maximum count in mode 0] + 1 – desired count
= 1FFFH + 1 – desired count = 2000H – desired count (H)
When the timer overflows the corresponding TF bit in TCON is set. This bit indicates the overflow of the respective timer. And also generates the timer interrupt.



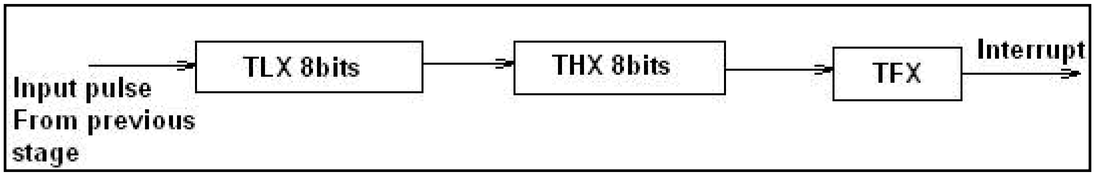
Mode 1: 16-bit mode for Timer0 & Timer1

It is selected by making M1, M0 in TMOD = 01

This is the **16-bit mode** for both the timers. The 16-bit count is placed in corresponding THx and TLx registers. Mode 1 uses all the 8-bits of THx register and TLx register to hold the 16-bit count. The timer starts working after TRx bit is set in TCON. After starting timer the 16-bit count is incremented by one per machine cycle, this is referred as timer operation. If the external clock on pins T0 or T1 is used, the count in registers is incremented by one for every cycle of external clock. Then the timer is working as counter.

The timer is said to overflow when the count in registers TH and TL rolls from FFFFH to 0000H. This overflow takes after the value FFFFH (65535d). So maximum count for timer mode 1 is FFFFH (65535d).

Then the count that should be placed in timer registers
= [maximum count in mode 1] + 1 – desired count
= FFFFH + 1 – desired count = 10000H – desired count (H)
When the timer overflows the corresponding TF bit in TCON is set. This bit indicates the overflow of the respective timer. And also generates the timer interrupt.



Mode 2: 8-bit auto-reload mode for Timer0 and Timer1

It is selected by making M1, M0 in TMOD = 10

This is the **8-bit auto-reload mode** for both the timers. The 8-bit count is placed in corresponding THx and TLx registers. Mode 2 uses all the 8-bits of TLx register to hold the 8-bit count. The timer starts working after TRx bit is set in TCON. This 8-bit count in TLx is incremented by one per machine cycle; this is referred as timer operation. If the external clock on pins T0 or T1 is used, the count in registers is incremented by one for every cycle of external clock. Then the timer is working as counter.

The timer is said to overflow when the count in register TLx rolls from FFH to 00H. This overflow takes after the value FFH (255d). So maximum count for timer mode 2 is FFH (255d).

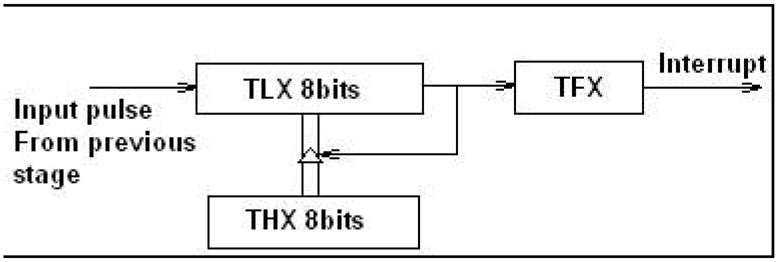
Then the count that should be placed in timer registers

= [maximum count in mode 2] + 1 – desired count

= FFH + 1 – desired count = 100H – desired count (H)

When the timer overflows the corresponding TF bit in TCON is set. This bit indicates the overflow of the respective timer. And also generates the timer interrupt.

Simultaneously the 8051 loads TLx register with the count in THx register. Thus the TLx register is automatically reloaded after the overflow of timer from FFH to 00H. And counting starts again from this value. Again after the overflow, TL is reloaded with the count in TH register. And therefore the Timer mode 2 is called as auto-reload mode, as it reloads the value in TLx from THx, after every overflow. If we want to stop the timer, then reset (clear) the TRx bit in TCON register.



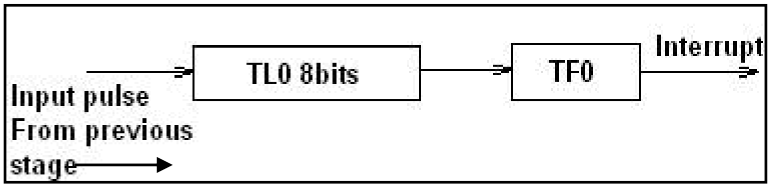
This mode is commonly used for timer1 in serial communication for generating specific baud rate.

Mode 3: 8-bit split timer mode for timer0 only

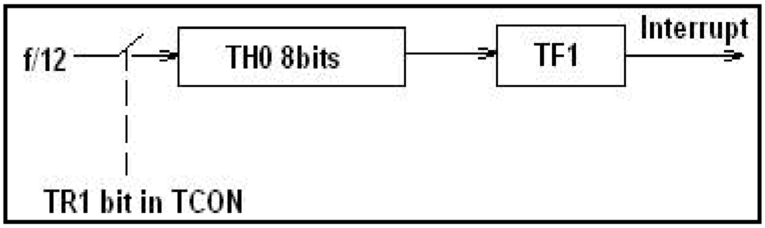
It is selected by making M1, M0 in TMOD = 11

This is the **8-bit split timer mode for timer0** only. The timer1 can be simultaneously used in mode0 or 1 or 2. In mode3, the timer0 is divided into two parts TH0 and TL0. Both these 8-bit registers work **separately as two 8-bit timers.**

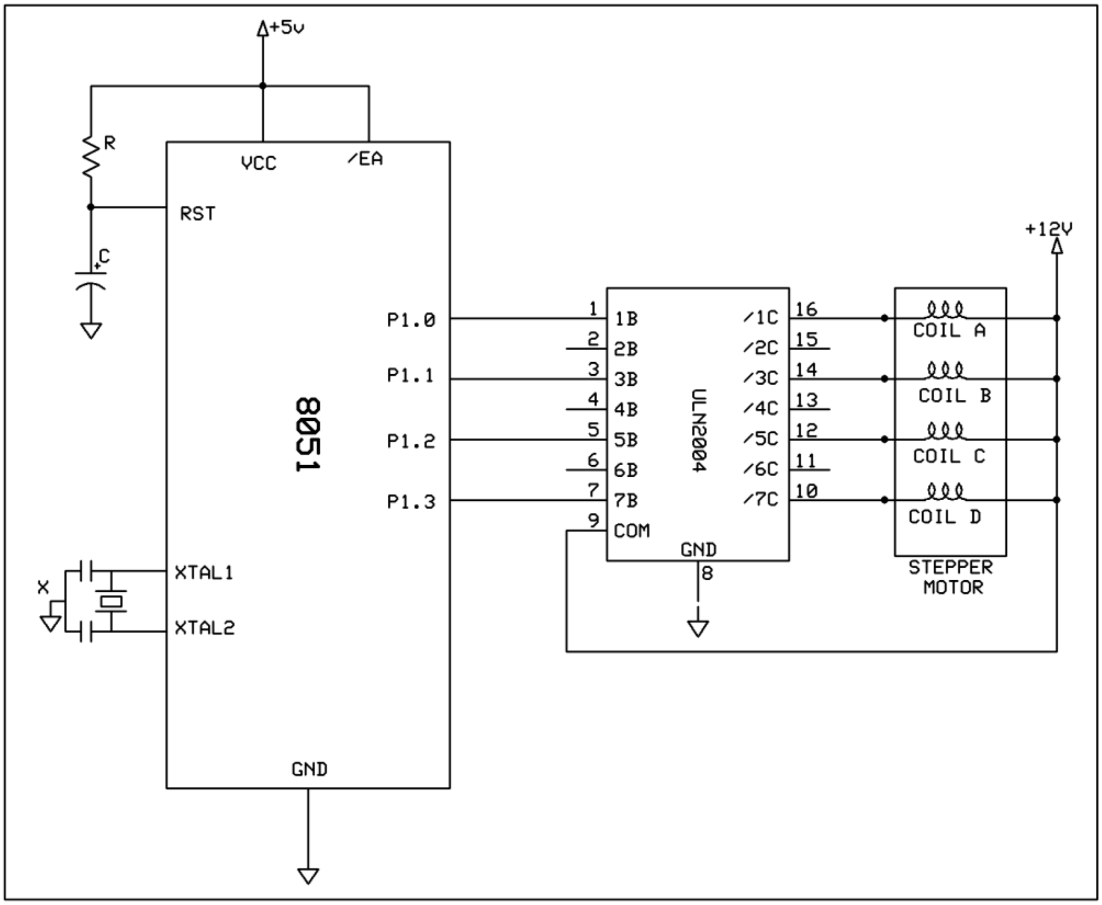
TL0 uses the TR0 and TF0 bits of timer0. And it can be programed as counter (can count external pulses from pin P3.2) or timer (can count internal pulses).



TH0 uses the TR1 and TF1 of timer1. And it can only count internal pulses (working as timer only).



When timer0 is used in mode3, the timer1 can be used in mode0 or mode1 or mode2. Timer1 can be used in these modes if timer1 do not need its own flags TF1 and TR1 (as they will be used by TH0 register). As TF1 is used by TH0 timer register, no interrupt will be generated by timer1. Thus timer1 can be used for baud rate generation or any other application where it does not require the flags TF1 and TR1.

	ii)	**Draw the interfacing of stepper motor with 8051 and write program in assembly or 'C' language to rotate it continuously in clockwise direction.**	**6M**
	Ans:	 +5v R C VCC /EA RST P1.0 P1.1 P1.2 P1.3 8051 X XTAL1 XTAL2 GND 1 1B 2 2B 3 3B 4 4B 5 5B 6 6B 7 7B 9 COM ULN2004 GND 8 /1C 16 /2C 15 /3C 14 /4C 13 /5C 12 /6C 11 /7C 10 +12V COIL A COIL B COIL C COIL D STEPPER MOTOR **Program:** ;PROGRAM : SIMPLE STEPPER MOTOR DRIVER **USING ROTATE INSTRUCTION** ORG 0000H ;START THE PROGRAM MOV A,#66H ;LOAD THE CODE TO DRIVE THE MOTOR UP: MOV P1,A ;SEND CODE TO P1 LCALL DELAY ;DELAY RR A ;ROTATE THE CODE HERE: SJMP HERE ;STOP PROGRAM AFTER 180O ROTATION DELAY: ; DELAY LOOP WITIN LOOP MOV R7,#0FFH UP2: MOV R6,#0FFH UP1: DJNZ R6,UP1 DJNZ R7,UP2 RET	**(Interfacing diagram- 3 Marks, Program - 3 Marks)** **[NOTE: Program may change. Student can also use the other logic. Please check the logic and understanding of students.]**

OR

We are sending 4 pulses as follow

```
; A B C D
; 1 0 0 1 =09 COIL A & D ON
; 1 1 0 0 =0C COIL A & B ON
; 0 1 1 0 =06 COIL B & C ON
; 0 0 1 1 =03 COIL C & D ON
```

;PROGRAM : STEPPER MOTOR DRIVER **USING LOOKUP TABLE**

```
ORG 0000H                     ;START THE PROGRAM
RPT: MOV DPTR,#0400H          ;LOAD THE STARTING ADDRESS OF
LOOKUP
TABLE
MOV R7,#04H                   ;LOAD THE COUNTER
CLR A                         ;CLEAR THE A
H:MOVC A,@A+DPTR              ;TAKE THE CODE FROM LOOKUP TABLE
MOV P1,A                      ;SEND THE CODE TO P0
INC DPTR                      ;INCREMENT DPTR FOR NEXT CODE
LCALL DELAY                   ;DELAY
DJNZ R7,H                     ;DECREMENT THE COUNTER
SJMP RPT                      ;STOP PROGRAM AFTER 180O ROTATION
ORG 0400H                     ;STARTING ADDRESS OF LOOKUP TABLE
DB 09H ;0400H = 09H
DB 0CH ;0401H = 0CH
DB 06H ;0402H = 06H
DB 03H ;0403H = 03H
DELAY:                        ;DELAY LOOP WITIN LOOP
MOV R0,#0FFH
UP2: MOV R1,#0FFH
UP1: DJNZ R1,UP1
DJNZ R0,UP2
RET
```

OR

// **C language program Stepper motor interfacing**

```
#include <Intel\8052.h>
#include <standard.h>
void delay (unsigned int);
/*
COIL A = P1.0
COIL B = P1.1
COIL C = P1.2
COIL D = P1.3
*/
void main ()
{
P1 = 0x00; //MOTOR OFF
While(1)
{
P1 = 0x09; // COIL A & D ON
delay (100);
P1 = 0x0C; // COIL A & B ON
delay (100);
P1 = 0x06; // COIL B & C ON
delay (100);
P1 = 0x03; // COIL C & D ON
delay (100);
}
}
void delay(unsigned int t)
{
Unsigned int x, y;
for(x=0; x<=t; x++)
for(y=0; y<=675; y++);
}
```

[NOTE: Program may change. Student can also use the other logic. Please check the logic and understanding of students.]

5.		**Attempt any four of the following:**	**16Marks**
	i)	**State any four logical instructions of 8051 microcontroller.**	**4M**
	Ans:	**Logic Instructions** Logic instructions perform logic operations upon corresponding bits of two registers. After execution, the result is stored in the first operand.	**(Any Four instructions: 4 marks)**

Mnemonic	**Description**	**Byte**	**Cycle**
ANL A,Rn	AND register to accumulator	1	1
ANL A,direct	AND direct byte to accumulator	2	2

			ANL A,@Ri	AND indirect RAM to accumulator	1	2		
			ANL A,#data	AND immediate data to accumulator	2	2		
			ANL direct,A	AND accumulator to direct byte	2	3		
			ANL direct,#data	AND immediae data to direct register	3	4		
			ORL A,Rn	OR register to accumulator	1	1		
			ORL A,direct	OR direct byte to accumulator	2	2		
			ORL A,@Ri	OR indirect RAM to accumulator	1	2		
			ORL direct,A	OR accumulator to direct byte	2	3		
			ORL direct,#data	OR immediate data to direct byte	3	4		
			XRL A,Rn	Exclusive OR register to accumulator	1	1		
			XRL A,direct	Exclusive OR direct byte to accumulator	2	2		
			XRL A,@Ri	Exclusive OR indirect RAM to accumulator	1	2		
			XRL A,#data	Exclusive OR immediate data to accumulator	2	2		
			XRL direct,A	Exclusive OR accumulator to direct byte	2	3		
			XORL direct,#data	Exclusive OR immediate data to direct byte	3	4		

	CLR A	Clears the accumulator	1	1	
	CPL A	Complements the accumulator (1=0, 0=1)	1	1	
	SWAP A	Swaps nibbles within the accumulator	1	1	
	RL A	Rotates bits in the accumulator left	1	1	
	RLC A	Rotates bits in the accumulator left through carry	1	1	
	RR A	Rotates bits in the accumulator right	1	1	
	RRC A	Rotates bits in the accumulator right through carry	1	1	

ii) List alternate functions of port 3 of 8051 microcontroller. **4M**

Ans:

Pin	**Name**	**Alternate Function**
P3.0	RXD	Serial input line
P3.1	TXD	Serial output line
P3.2	INT0	External interrupt 0
P3.3	INT1	External interrupt 1
P3.4	T0	Timer 0 external input
P3.5	T1	Timer 1 external input
P3.6	WR	External data memory write strobe
P3.7	RD	External data memory read strobe

(½ Marks Each)

iii) Draw the interfacing of DAC with microcontroller 8051. **4M**

Ans:

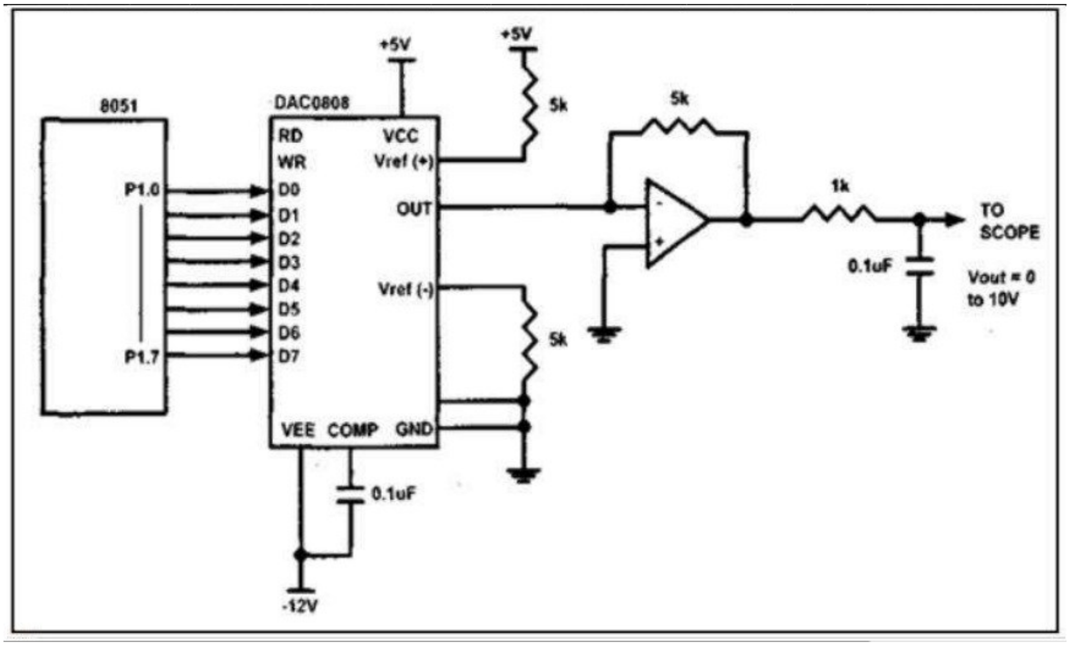
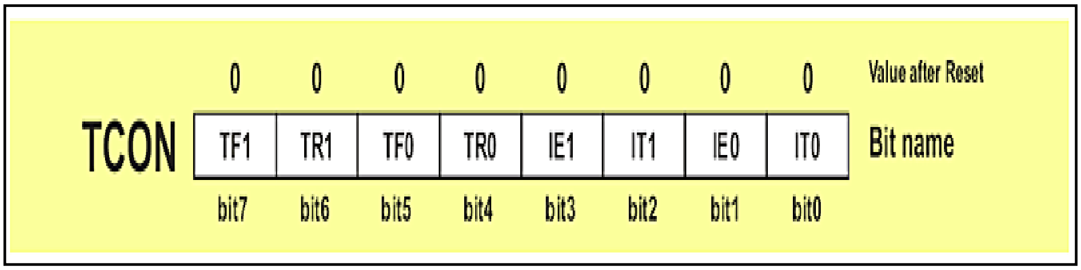


Fig. Circuit Diagram of interfacing DAC0808 with 8051 Microcontroller

(Complete interfacing diagram: 4 marks)

iv) Draw the format TCON and SCON in 8051 microcontroller. **4M**

Ans:

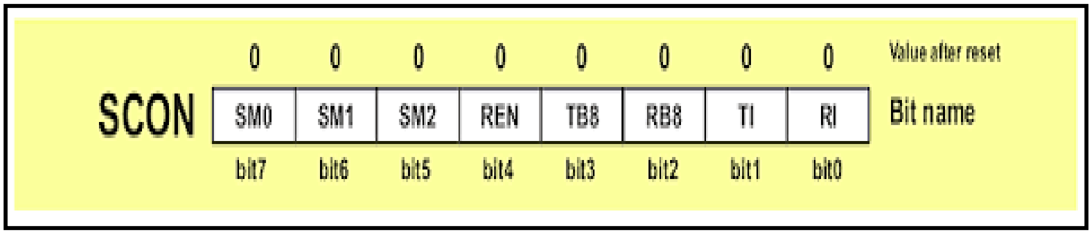


	0	0	0	0	0	0	0	0	Value after Reset
TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	Bit name
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	

(TCON: 2 Marks and SCON: 2 Marks)

Bit	Symbol	TCON Bit Function
7	TF1	Timer 1 Overflow flag. Set when timer rolls from all 1's to 0. Cleared when processor vectors to execute interrupt service routine located at program address 001Bh.
6	TR1	Timer 1 run control bit. Set to 1 by program to enable timer to count; cleared to 0 by program to halt timer.
5	TF0	Timer 0 Overflow flag. Set when timer rolls from all 1's to 0. Cleared when processor vectors to execute interrupt service routine located at program address 000Bh.
4	TR0	Timer 0 run control bit. Set to 1 by program to enable timer to count; cleared to 0 by program to halt timer.
3	IE1	External interrupt 1 Edge flag. Set to 1 when a high-to-low edge signal is received on port 3.3 (~~INT1~~). Cleared when processor vectors to interrupt service routine at program address 0013h. Not related to timer operations.
2	IT1	External interrupt 1 signal type control bit. Set to 1 by program to enable external interrupt 1 to be triggered by a falling edge signal. Set to 0 by program to enable a low-level signal on external interrupt 1 to generate an interrupt.
1	IE0	External interrupt 0 Edge flag. Set to 1 when a high-to-low edge signal is received on port 3.2 (~~INT0~~). Cleared when processor vectors to interrupt service routine at program address 0003h. Not related to timer operations.
0	IT0	External interrupt 0 signal type control bit. Set to 1 by program to enable external interrupt 1 to be triggered by a falling edge signal. Set to 0 by program to enable a low-level signal on external interrupt 0 to generate an interrupt.

SCON SFR

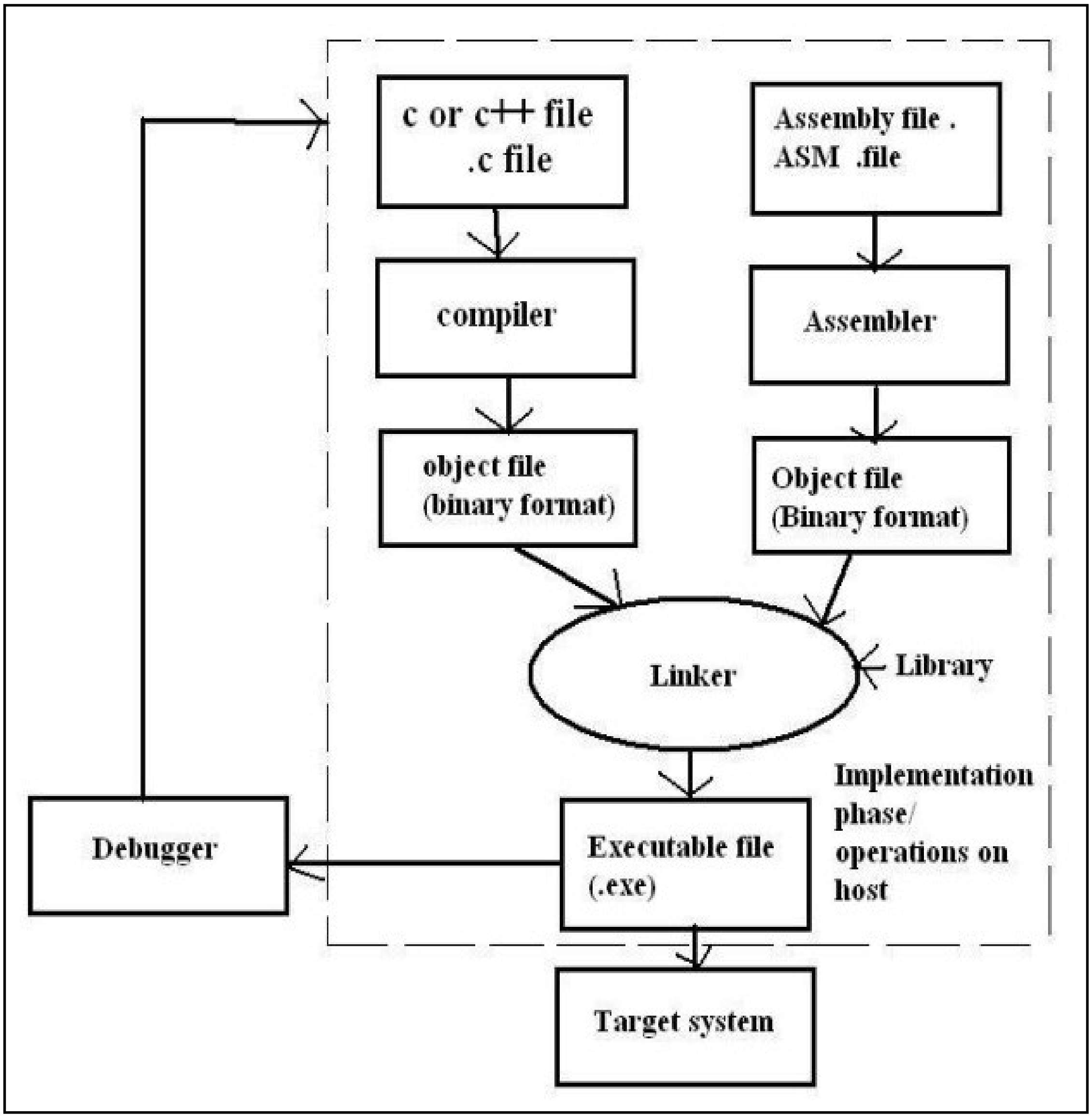


SCON	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
Value after reset	0	0	0	0	0	0	0	0
Bit name	SM0	SM1	SM2	REN	TB8	RB8	TI	RI

<table>
<tr><th>Bit</th><th>Symbol</th><th>SCON Bit Function</th></tr>
<tr><td>7</td><td>SM0</td><td>Serial port mode bit 1. Set/cleared by program to select mode.</td></tr>
<tr><td>6</td><td>SM1</td><td>Serial port mode bit 1. Set/cleared by program to select mode.
<table>
<tr><th>SM0</th><th>SM1</th><th>Mode</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Shift register; baud = f/12</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>8-bit UART; baud = variable</td></tr>
<tr><td>1</td><td>0</td><td>2</td><td>9-bit UART; baud = f/32 or f/64</td></tr>
<tr><td>1</td><td>1</td><td>3</td><td>9-bit UART; baud = variable</td></tr>
</table></td></tr>
<tr><td>5</td><td>SM2</td><td>Multiprocessor communications bit. Set/cleared by program to enable multiprocessor communications in modes 2 and 3. When set to 1 an interrupt is generated if bit 9 of the received data is a 1; no interrupt is generated if bit 9 is a 0. If set to 1 for mode 1, no interrupt will be generated unless a valid stop bit is received. Clear to 0 if mode 0 is in use.</td></tr>
<tr><td>4</td><td>REN</td><td>Receive enable bit. Set to 1 to enable reception; cleared to 0 to disable reception.</td></tr>
<tr><td>3</td><td>TB8</td><td>Transmitted bit 8. Set/cleared by program in modes 2 and 3.</td></tr>
<tr><td>2</td><td>RB8</td><td>Received bit 8. Bit 8 of received data in modes 2 and 3; stop bit in mode1. Not used in mode 0.</td></tr>
<tr><td>1</td><td>TI</td><td>Transmit Interrupt flag. Set to one at the end of bit 7 time in mode 0, and at the beginning of the stop bit for other modes. Must be cleared by the program.</td></tr>
<tr><td>0</td><td>RI</td><td>Receive Interrupt flag. Set to one at the end of bit 7 time in mode 0, and halfway through the stop bit for other moves. Must be cleared by the program.</td></tr>
</table>

v) State various steps in software development cycle an embedded system. **4M**

Ans:



Steps in Software development cycle:

1. Edit or write the Program
2. Convert program in machine language
3. Debug the program, if there is error repeat the step 1,2,3
4. Load hex file in target system

(Software development cycle diagram: 1 mark and Explanation: 3 marks)

		1. Edit or write the program: Editor is used to write the program. Program is written in assembly i.e .ASM or inembedded C i.e .C. 2. Convert program in machine language To convert the program in machine we have to use either Assembler or Compiler. If program is in assembly then we have to use assembler. Or if program is in embedded C the we have to use Compiler. 3. Debug the program, if there is error repeat the step 1,2,3: Debugger is used to debug i.e to find out errors in the program. If there is error in the program then we have to correct the program i.e remove the errors. For that we have to repeat step 1,2 and 3 until errors become zero. 4. Load hex file in target system When errors are zero, load the hex file into the target board. Target board is the final product or output of the embedded system	
	vi)	**State eight applications of embedded system.**	**4M**
	Ans:	1. Security systems 2. Telephone and banking 3. Defense and aerospace 4. Communication 5. Displays and Monitors 6. Networking Systems 7. Image Processing 8. Network cards and printers 9. Digital Cameras 10. Set top Boxes 11. High Definition TVs	**(½ Marks Each applications)**

		12. DVDs 13. Motor and cruise control system 14. Body or Engine safety 15. Entertainment and multimedia in car 16. E-Com and Mobile access 17. Robotics in assembly line 18. Wireless communication 19. Mobile computing and networking	
6.		**Attempt any four of the following:**	**16 Marks**
	i)	**Explain the following assembler directives with one example of each:** **i) ORG** **ii) END** **iii) DB** **iv) EQU**	**4M**
	Ans:	**i) ORG:- Origin** It is used to indicate the beginning of address. Syntax: **ORG Syntax** The address can be given in either hex or decimal there should be a space of at least one character between ORG & address fields. Some assemblers use ORG should not begin in label field. **ii) END:** This directive must be at the end of every program. meaning that in the source code anything after the END directive is ignored by the assembler. This indicates to the assembler the end of the source file(asm). Once it encounters this directive, the assembler will stop interpreting program into machine code. e.g. END ; End of the program. **iii) DB:- Data Byte** Syntax: **LABLE : DB Byte** Where byte is an 8-bit number represented in either binary, Hex, decimal or ASCII form. There should be at least one space between label & DB. The colon (:) must present after label. This directive can be used at the beginning	**(Each directive – 1 mark)**

of program. The label will be used in program instead of actual byte. There should be at least one space between DB & a byte.

iv) EQU: Equate

It is used to define constant without occupying a memory location.

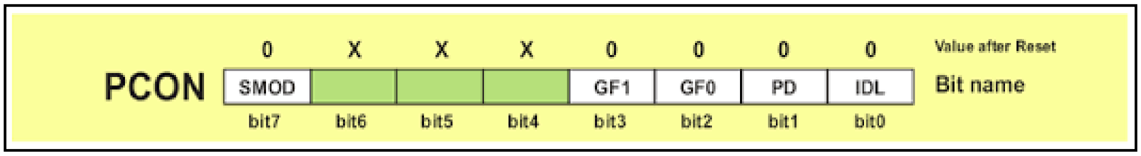
Syntax: **Name EQU Constant**

By means of this directive, a numeric value is replaced by a symbol. For e.g. MAXIMUM EQU 99 After this directive every appearance of the label "MAXIMUM" in the program, the assembler will interpret as number 99 (MAXIMUM=99).

ii) **Draw the format PCON in 8051 microcontroller. Explain function of each bit.** **4M**

Ans:

(PCON) Special Function Register



	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	
	0	X	X	X	0	0	0	0	Value after Reset
PCON	SMOD				GF1	GF0	PD	IDL	Bit name

(Format of PCON :2 marks Explanation : 2 marks)

Bit	**PCON Bit Function**
7	SMOD -- Serial baud modify bit. Set to 1 by program to double baud rate using timer 1 for modes 1, 2, and 3. Cleared to 0 by program to use timer 1 baud rate.
6	-- Not implemented.
5	-- Not implemented.
4	-- Not implemented.
3	General purpose user flag bit 1. Set/cleared by program.
2	General purpose user flag bit 0. Set/cleared by program.
1	Power down bit. Set to 1 by program to enter power down configuration for CHMOS processors.

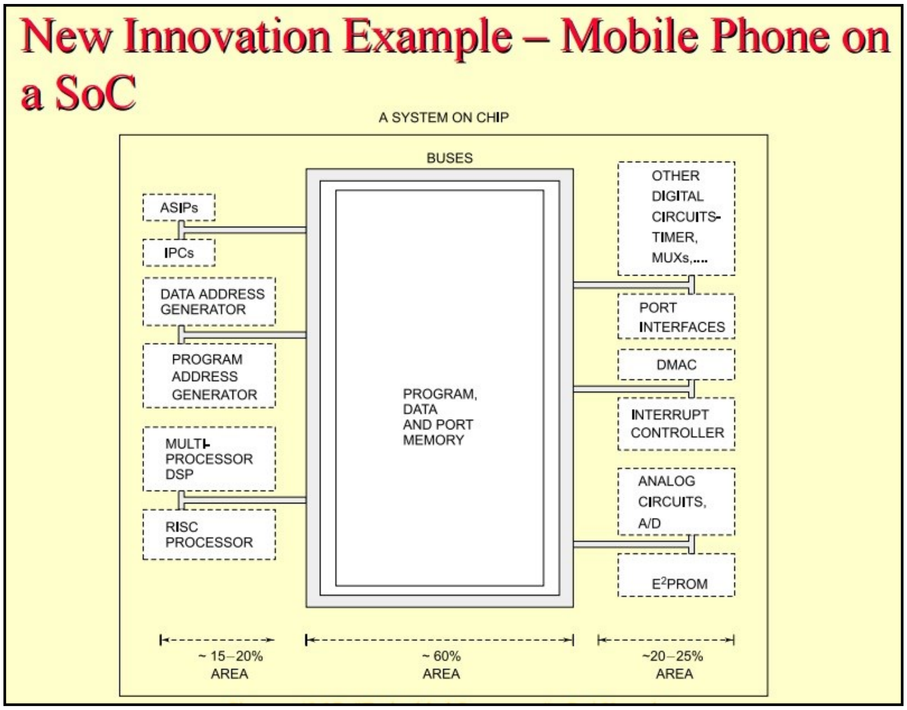
		0 Idle mode bit. Set to 1 by program to enter idle mode configuration for CHMOS processors.	
	iii)	**State two features of simulator and two features of debugger.**	**4M**
	Ans:	**Features of Simulator:** 1. Defines the processor or processing device family as well as its various versions for the target system. 2. Monitors the detailed information of a source code part with labels and symbolic arguments as the execution goes on for each single step. 3. Provides the detailed information of the status of RAM and ports (simulated) of the defined target system as the execution goes on for each single step. 4. Provides the detailed information of the status of peripheral devices (simulated, assumed to be attached) with the defined system. 5. Provides the detailed information of the registers as the execution goes on for each single step or for each single module. 6. Monitors system response and determines throughput. 7. The windows on the screen providing details 8. Detailed information of the status of stack, devices and ports (simulated) of the defined microcontroller system 9. Helps the window on the screen to provide the detailed meaning of the present command. 10. Monitors the detailed information of the simulator commands as these are entered from the keyboard or selected from the menu 11. Employs RTOS scheduler that preempts task. 12. Simulates the inputs from the interrupts, the timers, ports and peripherals. So it tests the codes for these. **Features of Debugger:** 1. Debugger gives list of errors in the program. 2. Debugger gives the location i.e line number where error is present. 3. Debugger gives bugs in a piece of electronic hardware.	**(Any two feature of simulator:2 marks,** **Any two feature of debugger : 2 marks)**

	iv)	**Explain the concept of inter process communication in Real Time Operating System (RTOS).**	**4M**
	Ans:	Interprocess communication (IPC): 1. Interprocess communication (IPC) is a set of programming interfaces that allow a programmer to coordinate activities among different program processes that can run concurrently in an operating system. 2. This allows a program to handle many user requests at the same time. 3. Since even a single user request may result in multiple processes running in the operating system on the user's behalf, the processes need to communicate with each other. 4. The IPC interfaces make this possible. 5. Each IPC method has its own advantages and limitations so it is not unusual for a single program to use all of the IPC methods. **IPC methods:** 1. Pipes - Named pipes and un named pipes. 2. Message queue 3. Semaphores 4. Shared memory 5. Sockets	**(Concept of inter process communication – 4 marks)**
	v)	**Explain System-On-Chip (SOC) in Embedded system.**	**4M**
	Ans:	**SOC in Embedded System:** SOC is a System on Chip that has all needed analog as well as digital circuits, processors and software. **System on Chip Embeds following:** • Embedded processor GPP or ASIP core, • Single purpose processing cores or multiple processor cores, • A network bus protocol core, • An encryption and decryption functions cores, • Cores for FFT and Discrete cosine transforms for signal processing	**(Explanation of SOC: 4 marks)**

applications,

- Memories
- Multiple standard source solutions, called IP (Intellectual Property) cores,
- Programmable logic device and FPGA (Field Programmable Gate Array) cores.
- Other logic and analog units.

Example of SOC is single-chip mobile phone:



New Innovation Example – Mobile Phone on a SoC

vi) State the need of RTOS in embedded system and describe the specification of RTOS. **4M**

Ans: **Need of RTOS in embedded system:**

(RTOS need : 2 marks, RTOS Specifications :2 marks)

Most of Embedded system, runs only one applications at a time. And most of them are not real time. So To run multiple task or operation in real time we require Real Time Operating System i.e. RTOS.

The scheduler used in a multi user operating system. The scheduler in a Real Time Operating System (RTOS) is designed to provide a predictable (normally described as deterministic) execution pattern. This is particularly of interest to embedded systems as embedded systems often have real time requirements.

A real time requirements is one that specifies that the embedded system must respond to a certain event within a strictly defined time (the deadline). A guarantee to meet real time requirements can only be made if the behaviour of the operating system's scheduler can be predicted (and is therefore deterministic)

Specifications of RTOS:

1. **Reliability:**
   The RTOS is reliable, because it is available for all time and normally it does not fail to perform any function/operation. The reliability of system also depends on the hardware board support package and application code.
2. **Predictability:**
   In RTOS, the user knows within How much time period the RTOS is going to perform the task i.e. The RTOS has predictability. We can predict, determine how much time takes by RTOS.
3. **Performance:**
   The performance of RTOS is very fast so that it can fulfill all timing requirement
4. **Compactness:**
   The RTOS provide compactness. It required less memory space for storage and hence can be used for portable application, like cellphone, ECG machine, etc.
5. **Scalability:**
   RTOS can be used in a wide variety of embedded. They must be able to scale-up or scale-down to suit the application..