SUMMER – 19 EXAMINATION

Subject Name: Embedded System **Model Answer** **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		**Attempt any FIVE of the following :**	**20 M**
	a	**Draw the format of TMOD SFR. Explain the function of each bit.**	**4 M**
	Ans	**TMOD SFR (TIMER MODE SPECIAL FUNCTION REGISTER):**	Format : 2M Explanation : 2M

D7	D6	D5	D4	D3	D2	D1	D0
GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0

$\underbrace{\text{D7 D6 D5 D4}}_{\text{TIMER1}}$ $\underbrace{\text{D3 D2 D1 D0}}_{\text{TIMER0}}$

Bit D7 and D3 (Gate):

It is set by programmer to control timer/counter by external hardware interrupts $\overline{INT}1$ and $\overline{INT}0$

If Gate =0 respective timer/counter operation is independent of external hardware interrupt inputs such as $\overline{INT}1$ and $\overline{INT}0$

		If Gate =1 respective timer/counter operation is dependent of hardware interrupt inputs such as $\overline{INT1}$ and $\overline{INT0}$. **Bit D6 and D2(C/$\overline{T}$):** If C/$\overline{T}$ = 1 input frequency source for respective timer will be external clock supply and hence called as Counter. If C/$\overline{T}$ = 0 input frequency source for respective timer will be internal clock supply and hence called as Timer. **Bit D5, D4 and D1, D0 (M1, M0):** These bits decides the mode of respective timer or counter M1 M0 Mode selected 0 0 mode 0 - 13 bit timer 0 1 mode 1 - 16 bit timer 1 0 mode 2 - 8-bit Auto-reload timer 1 1 mode 3 - 8-bit split timer	
	b	**State various addressing modes available in 8051 with two examples each.**	**4 M**
	Ans	8051 supports following addressing modes. 1. **Immediate Addressing Mode:** In this addressing mode the operand/data is specified within an instruction itself. Examples: **MOV A, #20H** This instruction moves immediate data 20H into Accumulator. **MOV R5, #88H** This instruction moves immediate data 88H into R5 register. 2. **Direct Addressing Mode:** In this addressing mode the operand/data is not given rather its address is mentioned where it is situated. Examples: **MOV A,20H** This instruction copies the content of internal RAM location 20H in an Accumulator **MOV 30H,20H** This instruction copies the content of internal RAM location 20H in the internal RAM location 30H	Any four Addressing modes Each Addressing Mode with correct example : 1M

3. Indirect Addressing Mode: In this addressing mode as the name employed address of the operand is not given directly, rather appropriate register acts as memory pointers which locate the operand.

Examples:

MOV A, @R0 This Instruction looks into R0, it provides an 8-bit address. The content of this address is then copied into an Accumulator.

MOVX A, @DPTR This instruction looks into DPTR which provides 16-bit address of an external RAM, The content of this location is then copied into an Accumulator.

4. Register Addressing Mode: In this addressing mode the operand/data is specified in the register itself, here specified register gives data directly.

Examples:

MOV A, R_5 This instruction copies the data from R_5 to Accumulator.

ADD A, R_2 This instruction adds content of Accumulator with R_2 and then store result back to Accumulator.

5. Indexed Addressing Mode: This addressing mode is dedicated to access Code/ROM/Program Memory.

Examples:

MOVC A,@A+DPTR This instruction adds Accumulator with DPTR, resultant is treated as address of ROM and from that address it copies the content to Accumulator.

MOVC A, @A+PC This instruction adds Accumulator with PC, resultant is treated as address of ROM and from that address it copies the content to Accumulator.

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

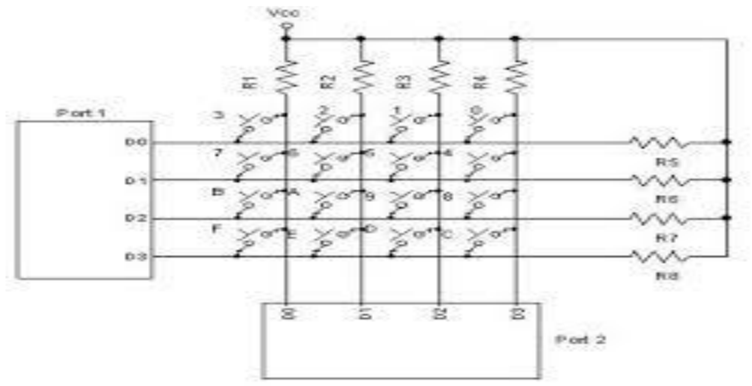
Ans

PIN	SYMBOL	SFR	SIGINIFICANCE	-
P3.0	RXD	SBUF	It is the received data pin for serial	Serial port in UART mode
P3.1	TXD	SBUF	It is the transmit data pin for serial	Serial port in UART mode
P3.2	INT0	TCON.1	It is an external interrupt	It is low level or falling edge trigerred
P3.3	INT1	TCON.3	It is an external interrupt	It is low level or falling edge trigerred

Each correct function

½ M

P3.4	T 0	-	External timer / counter 0	Input pin gives pulse to T0, register of the timer 0 to increase N by 1
P3.5	T 1	-	External timer / counter 1	Input pin gives pulse to T1, register of the timer 0 to increase by 1
P3.6	WR	-	It is an external memory writer	Pulse it is an active low pulse
P3.7	RD	-	It is an external memory reader	Pulse whenever data from memory is read

	d	**Draw labelled diagram to interface 4 x 4 keyboards to 8051.**	**4 M**
	Ans	 Vcc R1 R2 R3 R4 Port 1 D0 D1 D2 D3 3 2 1 0 7 6 5 4 B A 9 8 F E D C R5 R6 R7 R8 D0 D1 D2 D3 Port 2	Correct Diagram : 4M
	e	**State four features of embedded systems and state any four applications.**	**4 M**
	Ans	1) Embedded systems are designed to do some specific task, rather than be a general-purpose computer for multiple tasks. Some also have real-time performance constraints that must be met, for reasons such as safety and usability; others may have low or no performance requirements, allowing the system hardware to be simplified to reduce costs. 2) Embedded systems are not always standalone devices. Many embedded systems consist of small, computerized parts within a larger device that serves a more general purpose 3) The program instructions written for embedded systems are referred to as firmware, and are stored in read-only memory or Flash memory chips. They run with limited computer hardware resources: little memory, small or non-existent keyboard and/or screen. 4) Size & Weight**:** Microcontrollers are designed to deliver maximum performance for minimum size and weight. A centralized on-board computer	Any four feature & Application Each feature ½ M Each Application ½ M

system would greatly outweigh a collection of microcontrollers.
5) **Efficiency:** Microcontrollers are designed to perform repeated functions for long periods of time without failing or requiring service.

Applications: (any four of the following)

1. Applications of Embedded Systems: Embedded systems are used in different applications like automobiles, telecommunications, smart cards, missiles, satellites, computer networking and digital consumer electronics.

2. Embedded Systems in Automobiles and in telecommunications

- Motor and cruise control system
- Body or Engine safety
- Entertainment and multimedia in car
- E-Com and Mobile access
- Robotics in assembly line
- Wireless communication
- Mobile computing and networking

3. Embedded Systems in Smart Cards, Missiles and Satellites Security systems

Telephone and banking Defense and aerospace Communication

4. Embedded Systems in Peripherals & Computer Networking Displays and Monitors Networking Systems Image Processing Network cards and printers

5. Embedded Systems in Consumer Electronics Digital Cameras Set top Boxes High Definition TVs DVDs.

	f	**Differentiate between RTOS and Desktop operating system.**	**4 M**
	Ans	(see table below)	Any four points Each point : 1M

Sr.No.	**RTOS**	**Desktop O.S.**
1	It has deterministic time response	Does not have deterministic time response
2	Real time kernel	Generalized kernel
3	There is task deadline in RTOS	There is no task deadline
4	Memory required(footprint) is less	Memory required depends on version
5	Applications are compiled and linked together with RTOS	Applications are compiled separately from O.S.
6	It is used in embedded systems	It is used in general desktop Computer

7	It is more reliable	It is less reliable
8	e.g. RTLinux, VXWorks, Micro C/OSII	e.g. Windows, Linux

g	**Explain system on chip (SOC) in embedded system.**	**4 M**

Ans

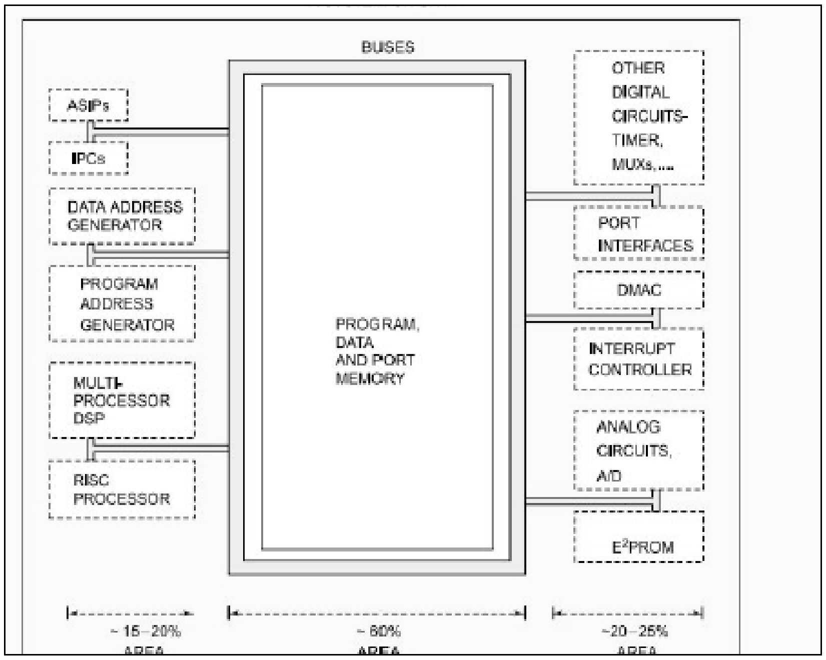
SOC in Embedded System:
SOC is a System on Chip that has all needed analog as well as digital circuits, processors and software.

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 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:



Description: 4M
Description without example can be given full marks.

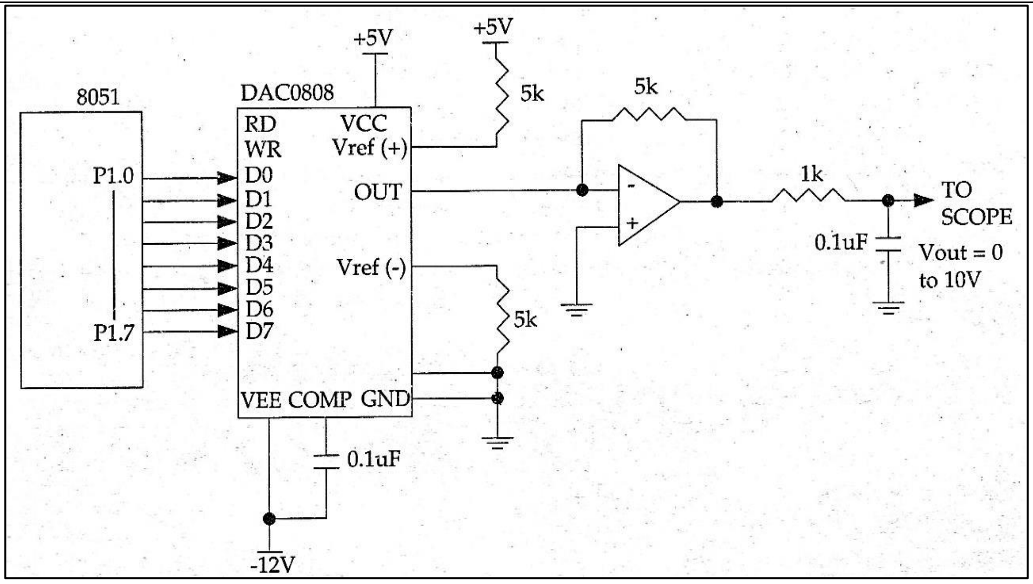
2		**Attempt any FOUR of the following :**	**16 M**
	a	**Describe any four assembler directives used in assembly language programming.**	**4 M**
	Ans	**i) DB:- Data Byte** Syntax: LABEL: DB BYTE Where byte is an 8-bit number represented in either binary, Hex, decimal or ASCII form. There should be at least one space between label & DB. The colon (:) must present after label. This directive can be used at the beginning of program. The label will be used in program instead of actual byte. There should be at least one space between DB & a byte. **ii) 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). **iii) ORG:- Origin** It is used to indicate the beginning of address. Syntax: ORG Address 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. **iv) 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.	Each directive 1M
	b	**Draw the format of IE SFR. Explain the function of each bit.**	**4 M**
	Ans	<u>**IE (Interrupt Enable) SFR :**</u> D7 | D6 | D5 | D4 | D5 | D2 | D1 | D0 EA | X | ET2 | ES | ET1 | EX1 | ET0 | EX0 (MSB) … (LSB) **Bit D7 (EA) :**	Format : 2M Explanation : 2M

		When EA=1 enable all interrupt. If EA=0 then all interrupts are disabled. **Bit D6 ‘X’:** It is don’t care bit. **Bit D5 (ET2):** It is Enable bit for Timer (only for 8052.) **Bit D4 (ES):** It is Serial Interrupt Enable bit. If ES = 1 then serial interrupt is enabled. If ES = 0 then it is disabled. **Bit D3 (ET1):** It is Timer 1 Overflow Interrupt Enable bit. If ET1 = 1 then interrupt is enabled. If ET1 = 0 then interrupt is disabled. **Bit D2 (EX1):** It is External Interrupt ($\overline{INT\ 1}$) Enable bit. If EX1 = 1 then interrupt is enabled. If EX1 = 0 then interrupt is disabled. **Bit D1 (ET0):** It is Timer 0 Overflow Interrupt Enable bit. If ET0 = 1 then interrupt is enabled. If ET0 = 0 then interrupt is disabled. **Bit D0 (EX0):** It is External Interrupt ($\overline{INT\ 0}$) Enable bit. If EX0 = 1 then interrupt is enabled. If EX0 = 0 then interrupt is disabled.	
	c	**Draw the diagram to interface DAC 0808 to 8051.Write the program in assembly or ‘C’ to generate ramp wave.**	**4 M**

Ans

Diagram : 2M

Program : 2M



Program in C

```
#include<reg51.h>
void delay(unsigned int);
unsigned int d;
void main(void)
{
while(1)

{
for(d=0; d<256; d++)
{
P1 = d;
Delay(2);
}
}
}
void delay(unsigned int i)
{
Unsigned int x,y;
for(x=0; x<i; x++)
for(y=0; y<100;y++);
}
```

Program in Assembly

```
ORG 0000H
REPEAT:CLR A
UP: MOV P1,A
INC A
LCALL DELAY
```

```
CJNE A,#0FFH,UP
SJMP REPEAT
MOV R2, #100
    DJNZ R2, $
    RET
```

d **Draw and explain the format of TCON register.** **4 M**

Ans Format of TCON:

Format : 2M

Explanation : 2M

TCON.7 TCON.6 TCON.5 TCON.4 TCON.3 TCON.2 TCON.1 TCON.0

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1 (TCON.7): Timer 1 overflows flag.
Set by hardware when the Timer/Counter 1 overflows.
Cleared by hardware as processor vectors to the interrupt service routine.

TR1 (TCON.6): Timer 1 run control bit.
Set/cleared by software to turn Timer/Counter 1 ON/OFF.

TF0 (TCON.5): Timer 0 overflow flag.
Set by hardware when the Timer/Counter 0 overflows.
Cleared by hardware as processor vectors to the service routine.

TR0 (TCON.4): Timer 0 run control bit.
Set/cleared by software to turn Timer/Counter 0 ON/OFF.

IE1 (TCON.3): External Interrupt 1 edge flag.
Set by hardware when External Interrupt edge is detected.
Cleared by hardware when interrupt is processed.

IT1 (TCON.2): Interrupt 1 type control bit.
Set/cleared by software to specify falling edge/low level triggered External Interrupt.

IE0 (TCON.1): External Interrupt 0 edge flag.
Set by hardware when External Interrupt edge detected.
Cleared by hardware when interrupt is processed.

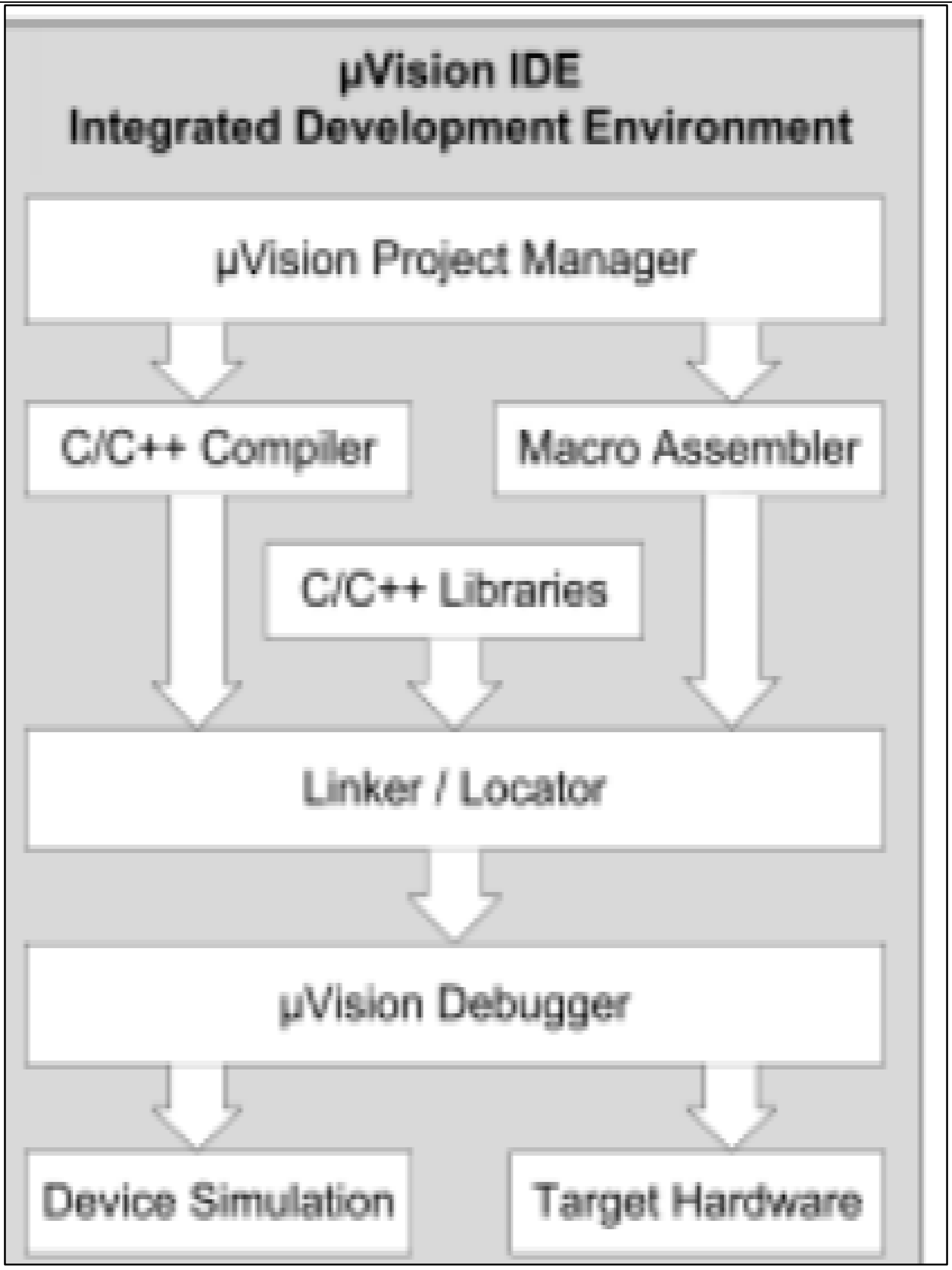
IT0 (TCON.0): Interrupt 0 type control bit.
Set/cleared by software to Specify falling edge/low level triggered External Interrupt.

	e	**Explain interprocess communication in RTOS.**	**4 M**
	Ans	Software is the basic building block of RTOS. Task is a simply subroutine. Task must be able to communicate with one another to coordinate their activities or to share the data. Kernel object is used for inter task/process communication. Kernel objects uses message queue, mail box and pipes, Shared Memory, Signal Function and RPC for inter task communication. **Message queue:** A message queue is a buffer like data structure, through which tasks and ISRs communicate with each other by sending and receiving messages and synchronize with data. It temporarily stores the message from a sender until the intended receiver is ready to read them. A message queue has queue control block, queue name, unique ID, memory buffers, a queue length. Kernel allocates the memory for message queue, ID, control block etc. **Mail box:** In general, mailboxes are similar to message queues. Mail box technique for inter task communication in RTOS based system used for one-way messaging. The task/thread creates mail box to send the message. The receiver task can subscribe the mail box. The thread which creates the mail box is known as mailbox server. The others are known as client. RTOS has function to create, write and read from mail box. No of messages (limited or unlimited) in mail box have been decided by RTOS. **Pipes:** Pipes are kernel objects used for unstructured data exchange between tasks facilities synchronization among tasks. Pipe provides a simple data transfer facility. **Shared Memory:** Shared memory is simplest way of inter process communication. The sender process writes data into shared memory and receiver process reads data. **Signal Function:** Operating system provides the signal function for messaging among task (process). **Remote Procedure Call (RPC) and Sockets:** RPC is a mechanism used by process (task) to call the procedure of another process running on same or different CPU in the network. Sockets are used for RPC communication and establish full duplex communication between tasks.	Correct explanation : 4M
	f	**State the function of the following:** **(i)In Circuit Emulator(ICE)** **(ii)Integrated Development Environment(IDE)** **(iii)Target Board**	**4 M**

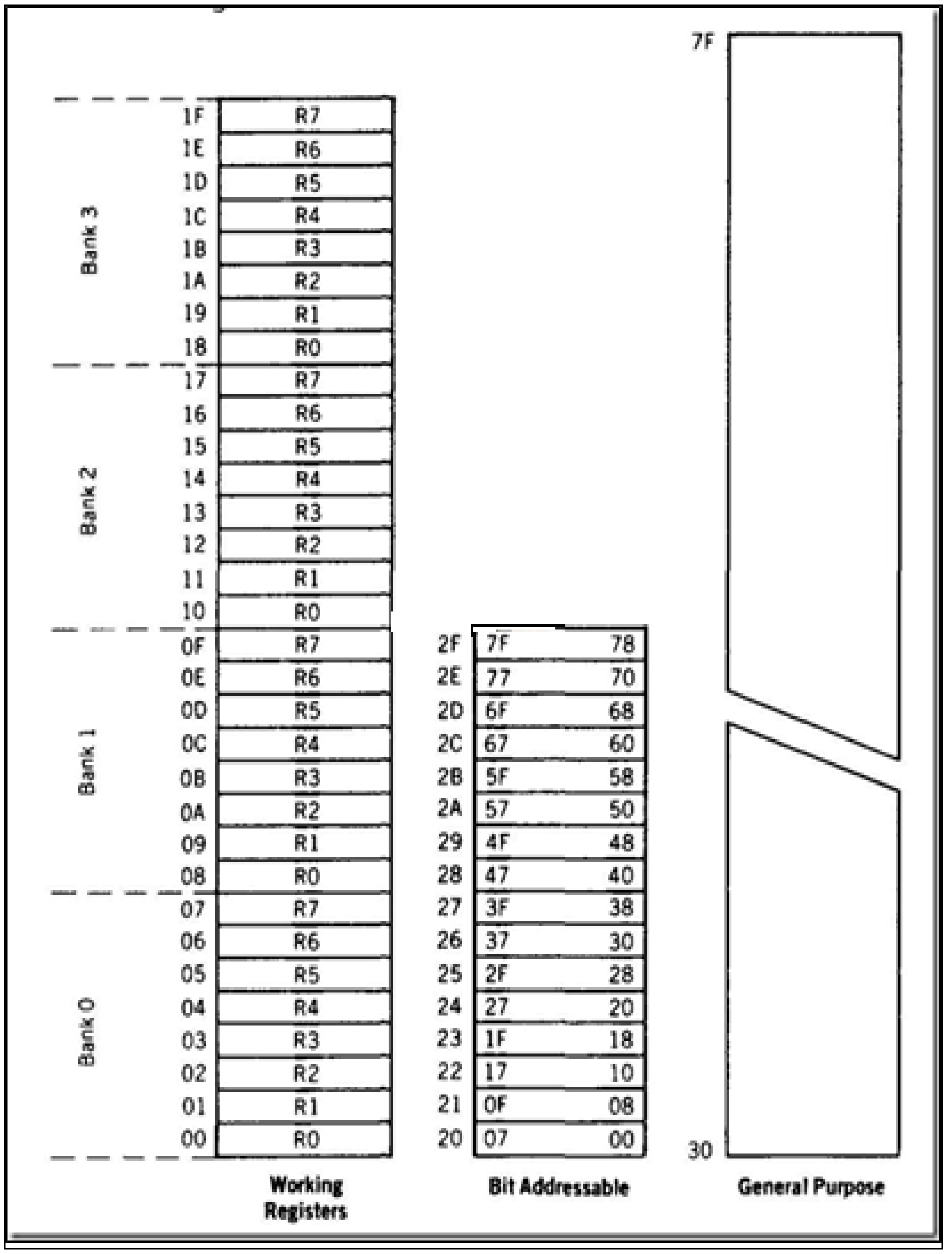
		(iv)Device programmer	
	Ans	**(i)In Circuit Emulator (ICE):** In-circuit emulator (ICE) is one of the oldest embedded debugging tools, and is still unmatched in power and capability. It is only tool that substitutes its own internal processor for the one in the target system Using one of a number of hardware tricks, the emulator can monitor everything that goes on in this on-board CPU, giving a complete visibility into the target code's operation. The emulator is bridge between the target and workstation giving an interactive terminal peering deeply into the target and a rich set of debugging resources. ICE uses another circuit with a card that connects to target processor (circuit) through a socket. **(ii)Integrated Development Environment (IDE):** An IDE is a software application that provides comprehensive facilities to computer programmers for software development. An IDE consists of: • A source code editor. • A compiler and or interpreter. • Build automation tools • A debugger. IDE is dedicated to a specific programming language, allowing a feature set that most closely matches the programming paradigms [model] of the language. IDE's typically present a single program in which all development is done. This program typically provides many features for authoring, modifying, compiling, deploying, and debugging software. **(iii)Target Board:** a. In the embedded world, most programming work is done on a computer system on which all the programming tools run. b. Only after the program has been written, complied assembled and linked it is moved to the final system. This is called the target system which is developed for customer. c. The host computer is therefore called as the development processor and the final system developed is called as the target processor system. d. In the development phase, the code of the application software has to be written in flash memory. These codes repeatedly written \ modified & tested using simulation & debugging tools till final phase (work according to specification) e. Once the application software finally working they may down load to ROM instead of flash memory in the target system.	Correct function 1M for each

		(iv) Device programmer: a. Device Programmers download a binary machine program from the development processor memory into the target processor's memory. b. The software of the device programmer runs at the host system. The host system interconnects with socket & device programmer runs at the host system. The host system interconnects with socket & device programmer circuit through serial port. c. The selected device inserted into a socket at the device programmer circuits & burns the code by transferring bytes for each address. d. Once the processor has been programmed, the entire ES can be tested real time. For example, a car programmed with engine management system can be taken out for a ride.	
3		**Attempt any FOUR of the following :**	**16 M**
	a	**Explain task synchronization and mutual exclusion in RTOS.**	**4 M**
	Ans	**Task Synchronization :-** Task 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). Task synchronization is achieved using mainly two types of mechanisms: a) Event Objects b) Semaphores a**) 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. b) **Semaphores**: 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. Other task synchronization methods are – • Message queue. • Mutual exclusion.	Task Synchroniz-ation : 2 M Mutual Exclusion : 2M

		• Dead lock. • Mailboxes. • Message Queues. **Mutual Exclusion:-** The easiest way for threads to communicate with each other is through shared data structures. This is especially easy when all threads exist in single address space and can reference global variables, pointers, buffers, linked lists, FIFOs etc. When two or more task access shared resources without corrupting data is called Mutual Exclusion. It can be performed in the following ways: • Disabling the scheduler • Disabling the interrupts • By test and set operation • Using semaphore	
	b	**State various steps in software development cycle of an embedded system.**	**4 M**
	Ans	Various steps involved in software development cycle are:-	Diagram : 2M Explanation : 2M

		 μVision IDE Integrated Development Environment μVision Project Manager C/C++ Compiler Macro Assembler C/C++ Libraries Linker / Locator μVision Debugger Device Simulation Target Hardware Development cycle involves the following steps 1. Writing codes 2. Translating codes	

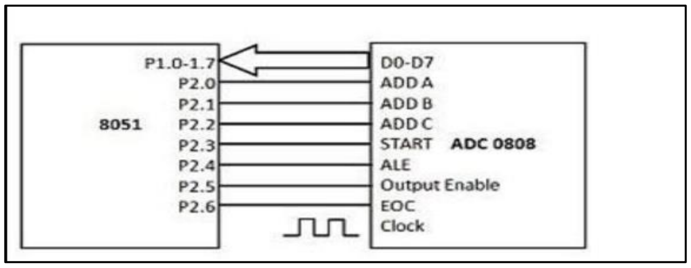
		3. Debugging the codes with the help of tools via emulators 4. Programming microcontroller to build up the first prototype of the system 1) Writing Microcontroller Code Software Code for a microcontroller is written in a programming language of choice (often Assembler or C). This source code is written with a standard ASCII text editor and saved as an ASCII text file. Programming in assembler involves learning a microcontroller's specific instruction set (assembler mnemonics), but results in the most compact and fastest code. A higher level language like C is for the most part independent of a microcontroller's specific architecture, but still requires some controller specific extensions of the standard language to be able to control all of a chip's peripherals and functionality. The penalty for more portable code and faster program development is a larger code size (20%...40% compared to assembler 2. Translating the Code Next the source code needs to be translated into instructions the microcontroller can actually execute. A microcontroller's instruction set is represented by "op codes". Op codes are a unique sequence of bits ("0" and "1") that are decoded by the controller's instruction decode logic and then executed. Instead of writing opcodes in bits, they are commonly represented as hexadecimal numbers, whereby one hex number represents 4 bits within a byte, so it takes two hex numbers to represent 8 Bits or 1 byte. For that reason a microcontroller's firmware in machine readable form is also called Hex-Code and the file that stores that code Hex-File. Assemblers, Compilers, Linkers and Librarians Assemblers or (C-) Compilers translate the human readable source code into "hex code" that represents the machine instructions (op codes) 3. Debugging the Code A debugger is a piece of software running on the PC, which has to be tightly integrated with the emulator that you use to validate your code. For that reason all emulator manufacturers ship their own debugger software with their tools, but also compiler manufacturers frequently include debuggers, which work with certain emulators, into their development suites. 4. OTP and Flash Programming It can't be stretched enough: A starter kit or emulators are no substitute for a production grade programmer. Using the microcontroller sockets on starter kit boards is ok to program one or two Samples in the lab, but those sockets cannot withstand hundreds or thousands of insertions. You will also find that starter kits do not include any sockets for surface mount devices, as those sockets are extremely expensive.	
	c	**Draw and explain the RAM structure of 8051.**	**4 M**
	Ans	RAM Structure of 8051:-	Diagram : 2M Explanation : 2M



There are 256 bytes of internal RAM on the 8051. 28 = 256, therefore the internal RAM address bus is 8 bits wide and internal RAM locations go from 00H to FFH.The first 128 locations (00H to 7FH) of internal RAM are used by the programmer for storing data while the second 128 locations (80H to FFH) are the Special Function Registers (SFRs).The diagram below is a summary of the 8051 on-chip RAM.

		Register Banks There are four register banks from 00H to 1FH. On power-up, registers R0 to R7 are located at 00H to 07H. However, this can be changed so that the register set points to any of the other three banks (if you change to Bank 2, for example, R0 to R7 is now located at 10H to 17H). Bit-addressable Locations The 8051 contains 210 bit-addressable locations of which 128 are at locations 20H to 2FH while the rest are in the SFRs. Each of the 128 bits from 20H to 2FH have a unique number (address) attached to them, as shown in the table above. The 8051 instruction set allows you to set or reset any single bit in this section of RAM. With the general purpose RAM from 30H to 7FH and the register banks from 00H to 1FH, you may only read or write a full byte (8 bits) at these locations. However, with bit-addressable RAM (20H to 2FH) you can read or write any single bit in this region by using the unique address for that bit. We will later see that this is a very powerful feature. Special Function Registers (SFRs) Locations 80H to FFH contain the special function registers. As you can see from the diagram above, not all locations are used by the 8051 (eleven locations are blank). These extra locations are used by other family members (8052, etc.) for the extra features these microcontrollers possess. Also note that not all SFRs are bit-addressable. Those that are having a unique address for each bit.	
	d	**Explain the following 8051 instructions :** **i. SET B C** **ii. SWAP A** **iii. MOV 80H, 90H** **iv. MUL AB**	**4 M**
	Ans	1. **SET B C**: - Set the CY bit of PSW register. This is Boolean instruction. After execution of this instruction CY bit will become 1. 2. **SWAP A** :- This instruction swaps bits 0-3 of the Accumulator with bits 4-7 of the Accumulator. 3. **MOV 80H, 90H**: - This instruction moves the content of memory location 90H to memory location 80H. 4. **MUL AB**: - Multiples the unsigned values of the Accumulator by the unsigned value of the "B" register. The least significant byte of the result is placed in the Accumulator and the most-significant-byte is	For each instruction : 1M

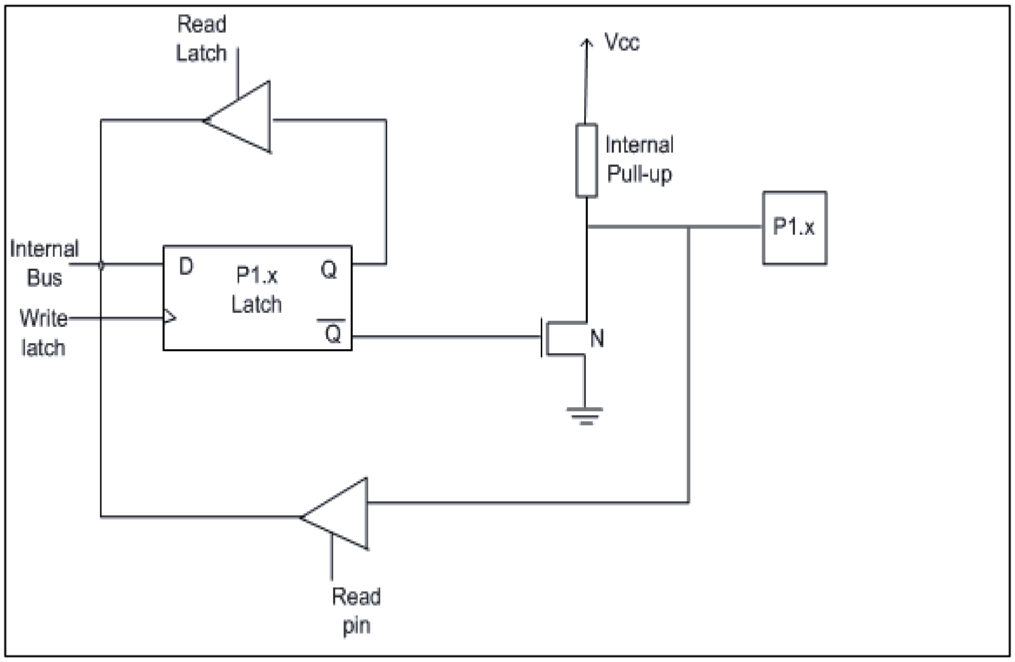
		placed in the "B" register.	
	e	**Write program in "C" or assembly language to generate a square wave of 50% duty cycle on bit "0" of part 1.**	**4 M**
	Ans	Program to generate a square wave with 50% duty cycle:- 50% duty cycle so on time and off time is same. Assume square wave of 1khz so Ton and Toff will be 500µ sec I/P clock = $(11.059 \text{ X } 10^6)/12$= 1000000 = 921.58KHz Tin = 1.085µ sec For 1 kHz square wave Fout = 1 KHz Ton= $1/ 1\text{X } 10^3$ Ton = 1000µ sec Consider half of it = Ton = 500µ sec N = Ton / Tin = 500/1.085 = 460.82 65536-461= (65075)10 = $(\text{FE33})_{16}$ NOTE: Students can consider any frequency with 50% duty cycle. Accordingly TH0 and TL0 will change. They can consider even timer 1. ORG 0000 MOV TMOD,# 01H ;____________ Mode 1,timer 0 HERE : MOV TL0,# 33H ; ________Lower byte of timer 0 MOV TH0, # 0FF ;_______________Higher byte of timer 0 CPL P1.0 ; ______________________toggle P 1.0 ACALL DELAY;	Calculation : 1M Program : 3M

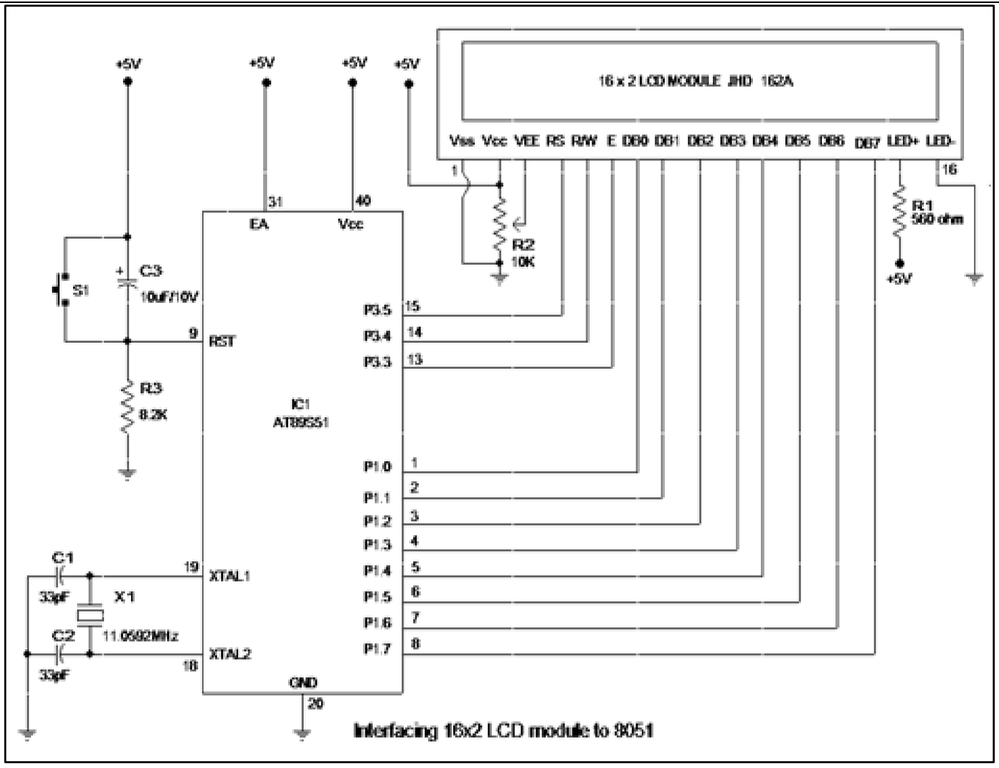
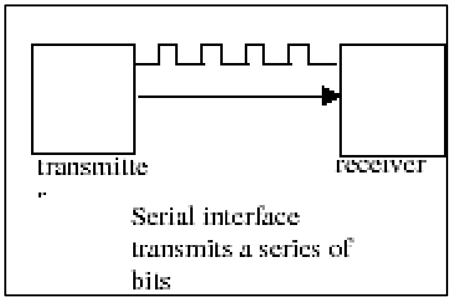
		SJMP HERE; (delay using timer 0) DELAY : SETB TR0 ; __________Start time 0 AGAIN : JNB TF0, _____________AGAIN CLR TR0 ; ____________________Stop timer 0 CLR TF0 ; RET	
	f	**Draw labelled diagram to interface Analog to Digital converter (ADC) 0808 to 0851.**	**4 M**
	Ans	Interfacing of ADC with 8051:  P1.0-1.7 P2.0 P2.1 8051 P2.2 P2.3 P2.4 P2.5 P2.6 D0-D7 ADD A ADD B ADD C START ADC 0808 ALE Output Enable EOC Clock	Diagram : 4M Note : any relevant diagram give marks
4		**Attempt any FOUR of the following :**	**16 M**
	a	**With suitable example, describe the concept of device driver.**	**4 M**
	Ans	Device Driver :- • Embedded system hardware has devices which communicate through serial & parallel ports & buses, There also may be ports for real time voice and video I/Os • A device access is required for opening, connecting, binding, reading, and writing, disconnecting or closing it. Processor accesses a device using the addresses of device registers & buffers. These devices could be internal devices, devices at the I/O ports, peripheral devices etc. • The concept of interrupt service routine is used to address & service the device I/Os and interrupts • Each device in a system needs device driver routines. An ISR relates to a	Concept : 3M Example : 1M

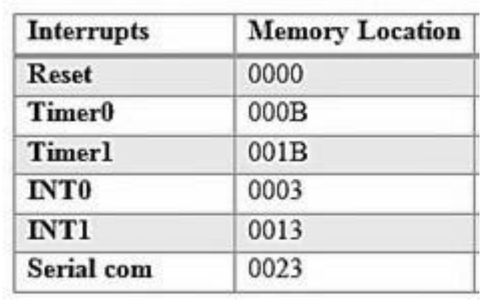
		device driver function • A device driver is a function used by a high level language programmer & does the interaction with device hardware & communicates data to the device, sends control commands to the device & runs the codes for reading the device data.	
	b	**Write an assembly language program or c program for rotating stepper motor in clockwise direction continuously using four sequence.**	**4 M**
	Ans	Rotate stepper motor:- Org 0000h Sjmp start Org 0030h Start: MOV SP,#30h MOV A,#66H ;load step sequence BACK: MOV P1,A ;issue sequence to motor RRA ;rotate right clockwise ACALL DELAY ;wait SJMP BACK ;keep going DELAY: MOV R2,#100 H1: MOV R3,#255 H2: DJNZ R3,H2 DJNZ R2,H1 RET End	Program : 3M Comment : 1M
	c	**Explain the need and requirement of RTOS in an embedded system.**	**4 M**
	Ans	Need of RTOS: 1. Scheduling, Synchronization: RTOS provides effective scheduling and	Need : 2M Requirement: 2M

		synchronization techniques which enables deterministic behavior. 2. Fast Execution: it provides running the user threads in Kernel space so that they execute fast. RTOS also provides faster memory allocation. 3. Task Priority: Pre-emption, priority in heritance takes care of task priorities. 4. Short-interrupt latency and deadlines: Interrupt latency is equal to hardware delay to get interrupt signal to the processor plus time to complete the current instruction plus time executing system code in preparation for transferring execution the device interrupt handler. Requirement of RTOS: i) Reliability Embedded systems must be reliable. Depending on the application, the system might need to operate for long periods without human intervention. Different degrees of reliability may be required. For example, a digital solar-powered calculator might reset itself if it does not get enough light, yet the calculator might still be considered acceptable. On the other hand, a telecom switch cannot reset during operation without incurring high associated costs for down time. The RTOS in these applications require different degrees of reliability. ii) Predictability Because many embedded systems are also real-time systems, meeting time requirements is key to ensuring proper operation. The RTOS used in this case needs to be predictable to a certain degree. The term deterministic describes RTOS with predictable behavior, in which the completion of operating system calls occurs within known timeframes. iii) Performance This requirement dictates that an embedded system must perform fast enough to fulfill its timing requirements. Typically, the more deadlines to be met-and the shorter the time between them-the faster the system's CPU must be. Although underlying hardware can dictate a system's processing power, its software can also contribute to system performance. Typically, the processor's performance is expressed in million instructions per second (MIPS). iv) Compactness Application design constraints and cost constraints help determine how compact an embedded system can be. For example, a cell phone clearly must be small, portable, and low cost. These design requirements limit system memory, which in turn limits the size of the application and operating system.	

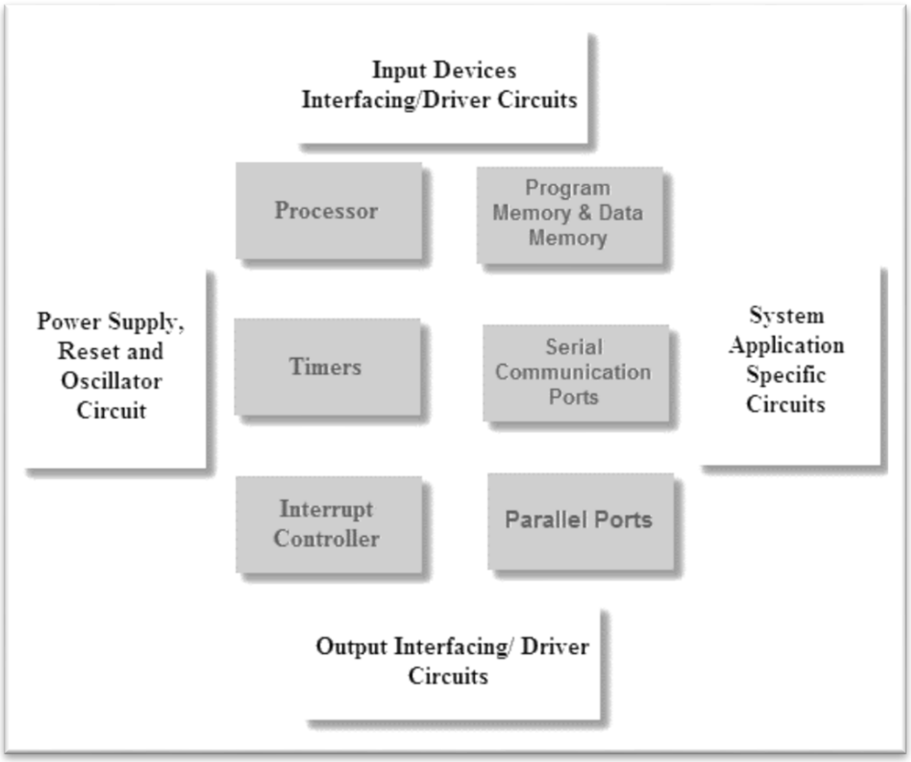
		v) Scalability Because RTOS can be used in a wide variety of embedded systems; they must be able to scale up or down to meet application-specific requirements. Depending on how much functionality is required, an RTOS should be capable of adding or deleting modular components, including file systems and protocol stacks.	
	d	**State the features of microcontroller 8051.**	**4 M**
	Ans	Features of Microcontroller 8051 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 T0 and T1 7. Five Interrupts (3 internal and 2 external). 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	Each Feature : 1 M
	e	**Write an assembly language program for the 8051 microcontroller to multiply two 8 bits numbers stored at memory location 20H and 21H. Store the product at 22H (LSB) and 23H (MSB).**	**4 M**
	Ans	Multiplication Program: ORG 0000H MOV A, 20H ; Get the first number MOV B, 21H ; get the second number MUL AB ; multiply first number with second number and results goes in A and B	Program : 3M Comment : 1M

		MOV 22H, A ; store LSB at 22h location MOV 23H, B ; store MSB at 23 h location END	
	f	**Describe the functions of part 1 of 8051 microcontroller and also draw the internal structure of part 1.**	**4 M**
	Ans	 Read Latch Vcc Internal Pull-up P1.x Internal Bus Write latch D P1.x Latch Q $\overline{Q}$ N Read pin Port 1 is called as dedicated port of microcontroller 8051. It can only be used as input output port.	Diagram : 3M Function :1M
5		**Attempt any FOUR of the following :**	**16 M**
	a	**Draw the labelled diagram to interface 16 x 2 LCD to microcontroller 8051.**	**4 M**

	Ans	 +5V 16 x 2 LCD MODULE JHD 162A Vss Vcc VEE RS R/W E DB0 DB1 DB2 DB3 DB4 DB5 DB6 DB7 LED+ LED- EA Vcc C3 10uF/10V S1 RST R3 8.2K IC1 AT89S51 P3.5 P3.4 P3.3 P1.0 P1.1 P1.2 P1.3 P1.4 P1.5 P1.6 P1.7 R1 560 ohm R2 10K C1 33pF C2 33pF X1 11.0592MHz XTAL1 XTAL2 GND Interfacing 16x2 LCD module to 8051	Diagram/ Any other correct relevant diagram : 4M
	b	**What is serial interface? Explain interrupts present in microcontroller 8051 for it.**	**4 M**
	Ans	A serial interface is a communication interface between two digital systems that transmits data as a series of voltage pulses down a wire. A "1" is represented by a high logical voltage and a "0" is represented by a low logical voltage. Essentially, the serial interface encodes the bits of a binary number by their "temporal" location on a wire rather than their "spatial" location within a set of wires. Encoding data bits by their "spatial" location is referred to as a parallel interface and encoding bits by their "temporal" location is referred to as a serial interface.  transmitter receiver Serial interface transmits a series of bits 8051 architecture handles 5 interrupt sources, out of which two are internal (Timer Interrupts), two are external and one is a serial interrupt. Each of these interrupts has their interrupt vector address. Highest priority interrupt is the	Serial Interface : 1M Interrupts : 3M

		Reset, with vector address 0x0000. Reset **Reset** is the highest priority interrupt, upon reset 8051 microcontroller start executing code from 0x0000 address.  **Internal interrupt** (Timer Interrupt) 8051 has two internal interrupts namely timer0 and timer1. Whenever timer overflows, timer overflow flags (TF0/TF1) are set. Then the microcontroller jumps to their vector address to serve the interrupt. For this, global and timer interrupt should be enabled. **Serial interrupt** 8051 has serial communication port and have related serial interrupt flags (TI/RI). When the last bit (stop bit) of a byte is transmitted, TI serial interrupt flag is set and when last bit (stop bit) of receiving data byte is received, RI flag get set.	
	c	**Draw the block diagram of embedded system. Explain various hardware units.**	**4 M**
	Ans	The various elements included in embedded system are:	Block diagram : 2M Explanation : 2M

Interrupts	Memory Location
Reset	0000
Timer0	000B
Timer1	001B
INT0	0003
INT1	0013
Serial com	0023


Input Devices Interfacing/Driver Circuits
Processor
Program Memory & Data Memory
Power Supply, Reset and Oscillator Circuit
Timers
Serial Communication Ports
System Application Specific Circuits
Interrupt Controller
Parallel Ports
Output Interfacing/ Driver Circuits

A: General Purpose Processor (GPP):

- ❖ Microprocessor: A microprocessor is a VLSI chip that has a CPU and may also have some units for caches, floating point processing arithmetic unit, pipelining and super-scalar architecture which results in faster processing of instructions. Ex: Intel 80x86
- ❖ Microcontroller: A microcontroller is a single chip VLSI unit which has limited computational capabilities but possesses enhanced input-output capabilities and a number of on-chip functional units. Ex: Intel 8051, 80251.

A. Application Specific System Processor [ASSP]: A processing unit for specific tasks like image compression and decompression, real time processing required in embedded system for HDTV, DVD players, Web phones and that is integrated through the buses with the main processor in the embedded system. ASSP alone can provide faster solutions to all these tasks. The ASSP is configured and interfaced with rest of the embedded system. Also ASSP can be used for encryption and Decryption and can be used as an additional processing unit for running the application specific tasks in place of processing using embedded software.

B. Multiprocessor system using GPP and Application Specific Instruction Processor [ASIP]: Multi processors are used when a single processor does not meet the requirements of different tasks that have to be performed concurrently. The operations of all the processors are synchronized to obtain an optimum performance. Real time video processing and multimedia applications most often need a multiprocessor unit in the embedded system.

C. GPP core or ASIP core integrated into either an Application Specific IC [ASIC] or a Very Large Scale IC [VLSI] circuits or an FPGA core integrated with processors unit in a VLSI (ASIC) chip.

Power Source: An internal power source or charge pump is used in every system. An embedded system has to perform tasks continuously from power up to power off and may even be kept on continuously. For embedded system software a performance analysis during its design phase must also include the analysis of power dissipation during program execution and during standby.

Clock Oscillator Circuit and Clocking units: For the processing units, a highly stable oscillator is required and the processor clock out signal provides the clock for synchronizing all the system units.

Reset Circuit, Power up reset and Watch dog timer: A program that is reset and runs on a power up can be one of the following:

- A system program that executes from the start.
- A system boot up program.
- A system initialization program.

The watch dog timer reset is used in control applications.

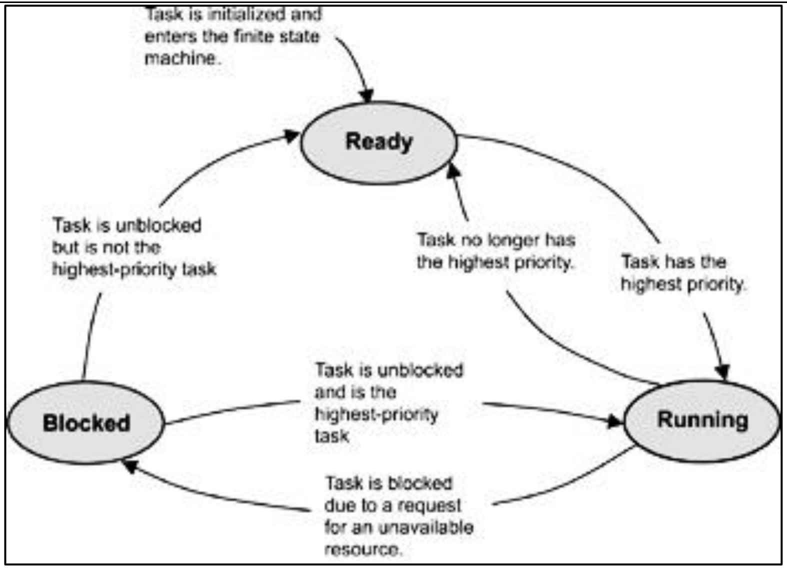
Memories: A system embeds the following in the internal ROM, PROM or in external ROM or PROM: boot up programs, initialization data, strings for an initial screen display or initial state of the system, the programs for the various tasks, ISR’s and kernel. The system has RAM’s for saving temporary data, stack and buffers that are needed during a program run. The system has flash memory for storing non-volatile results.

Input, Output and I/O ports, IO buses and IO peripherals: A system connects to the external physical devices and systems through parallel or serial I/O ports. Demultiplexers and Multiplexers facilitate communication of signals from multiple channels through a common path. A system often networks to the other devices and systems through an I/O bus like I^2C bus, CAN, USB, ISA, EISA and PCI bus.

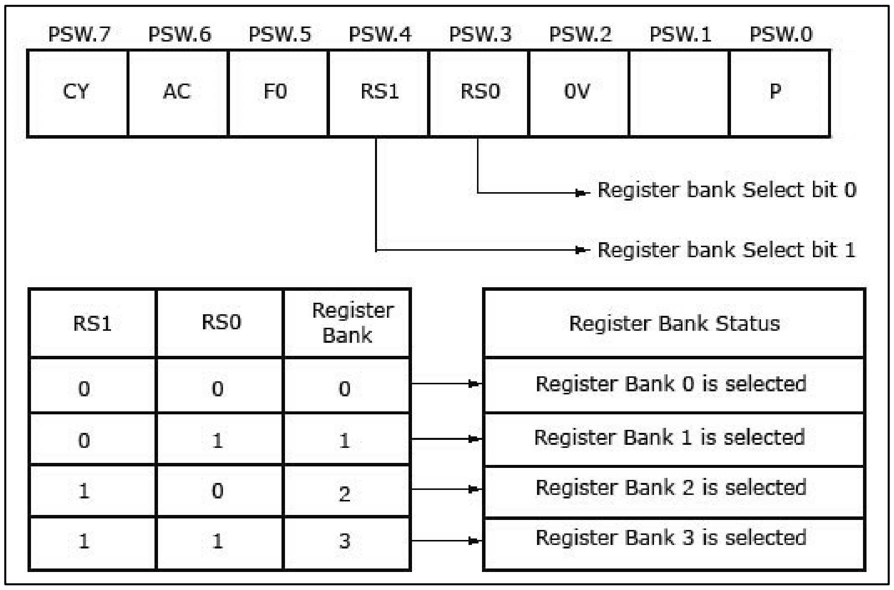
Interrupt Handler: A system must have interrupt handler mechanism for executing the ISR’s in case of the interrupts from physical devices, systems and software exceptions.

LCD and LED displays: For displaying and messaging, the LCD matrix displays and LED arrays are used in a system. The system must provide

		necessary interfacing circuit and software for the output to LCD display controller and LED interfacing ports. **Keyboard:** For inputs, a keyboard interface to a system. The system must provide necessary interfacing circuit and software to receive inputs directly from the keys or through the controller.	
	d	**What is a task in embedded system? What are various states of tasks**?	**4 M**
	Ans	Task: To achieve concurrency in real time application program, the application is decomposed into small, schedulable and sequential program units known as "Task." In real time context task is the basic unit of execution and is governed by three time critical properties; release time, deadline and execution time. Task States: Ready State: When a task is first created and made ready to run, the kernel puts it into the ready state. In this state, the task actively competes with all other ready tasks for the processor's execution time. As Figure shows, tasks in the ready state cannot move directly to the blocked state. Task first needs to run so it can make a blocking call, which is a call to a function that cannot immediately run to completion, thus putting the task in the blocked state. Ready tasks, therefore, can only move to the running state. Because many tasks might be in the ready state, the kernel's scheduler uses the priority of each task to determine which task to move to the running state. For a kernel that supports only one task per priority level, the scheduling algorithm is straightforward-the highest priority task that is ready runs next. In this implementation, the kernel limits the number of tasks in an application to the number of priority levels. However, most kernels support more than one task per priority level, allowing many more tasks in an application. In this case, the scheduling algorithm is more complicated and involves maintaining a task-ready list. Some kernels maintain a separate task-ready list for each priority level; others have one combined list.	Define task : 1M Task States : 3M

		 Task is initialized and enters the finite state machine. Ready Task is unblocked but is not the highest-priority task Task no longer has the highest priority. Task has the highest priority. Task is unblocked and is the highest-priority task Blocked Running Task is blocked due to a request for an unavailable resource. Running State: On a single-processor system, only one task can run at a time. In this case, when a task is moved to the running state, the processor loads its registers with this task's context. The processor can then execute the task's instructions and manipulate the associated stack. As discussed in the previous section, a task can move back to the ready state while it is running. When a task moves from the running state to the ready state, it is pre-empted by a higher priority task. In this case, the pre-empted task is put in the appropriate, priority-based location in the task-ready list, and the higher priority task is moved from the ready state to the running state. Unlike a ready task, a running task can move to the blocked state in any of the following ways: • By making a call that requests an unavailable resource, • By making a call that requests to wait for an event to occur, and • By making a call to delay the task for some duration. In each of these cases, the task is moved from the running state to the blocked state. Blocked State: The possibility of blocked states is extremely important in real-time systems because without blocked states, lower priority tasks could not run. If higher priority tasks are not designed to block, CPU starvation can result.	
	e	**Draw the format of PSW. Explain the function of each bit.**	**4 M**
	Ans	The program status word (PSW) register is an 8-bit register. It is also referred to as the *flag register*. Although the PSW register is 8 bits wide, only *6* bits of it are used by the 8051. The two unused bits are user-definable flags. Four of the flags are called *conditional flags,* meaning that they indicate some conditions that result after an instruction is executed. These four are CY (carry),	Format : 1 M Bit functions : 3M

AC (auxiliary carry), P (parity), and OV (overflow).
The bits PSW.3 and PSW.4 are designated as RSO and RSI, respectively, and are used to change the bank registers. They are explained in the next section.
The PSW.5 and PSW.l bits are general-purpose status flag bits and can be used by the programmer for any purpose.


PSW.7 PSW.6 PSW.5 PSW.4 PSW.3 PSW.2 PSW.1 PSW.0
CY AC F0 RS1 RS0 0V P
Register bank Select bit 0
Register bank Select bit 1
RS1 RS0 Register Bank
Register Bank Status
0 0 0 Register Bank 0 is selected
0 1 1 Register Bank 1 is selected
1 0 2 Register Bank 2 is selected
1 1 3 Register Bank 3 is selected

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 Is.

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.

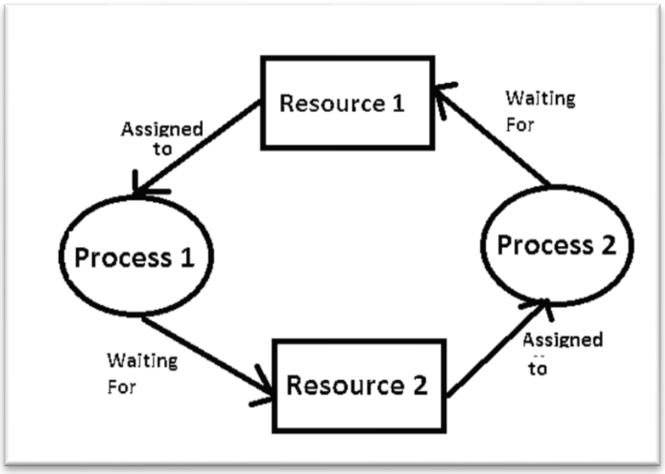
	f	**Write a program to unpack the 8 bit number using 8051 microcontroller**	**4 M**

		instructions using C or assembly language.	

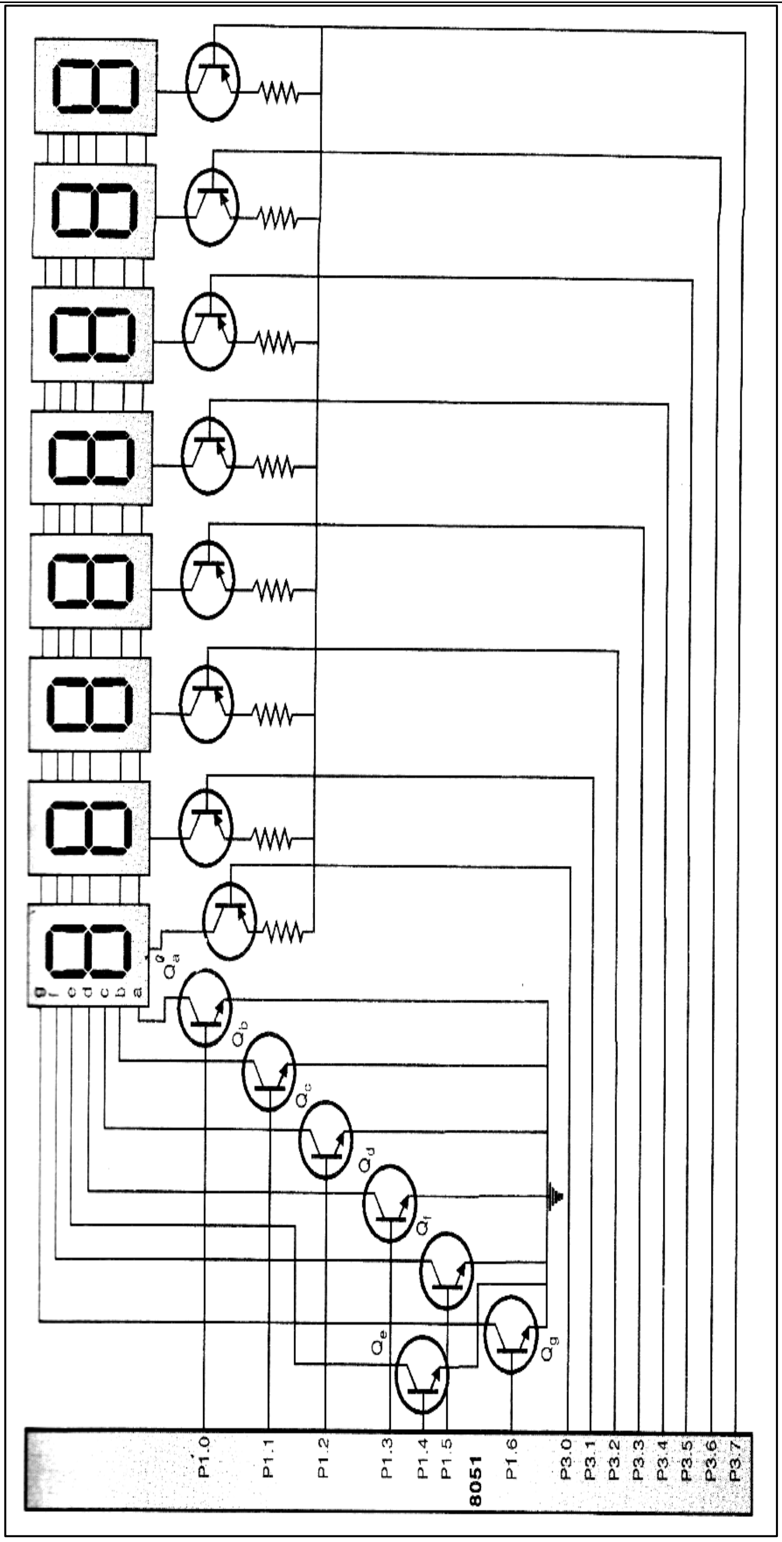
	Ans	**Assembly language Program:**
		ORG 00H
		MOV R0,50H ;get packed number from memory location 50H
		MOV A,R0 ;copy packed number to accumulator
		ANL A,#0FH ; mask upper nibble of packed number
		MOV 51H,A ;save lower digit to 51H
		MOV A,R0 ;get packed number again
		ANL A,#0F0H ;mask lower nibble of packed number
		SWAP A ;exchange lower and upper nibbles
		MOV 52H,A ;save upper digit
		END
		
		C language Program:
		
		#include <reg51.h>
		void main (void)
		{
		unsigned int x,y;
		unsigned char mybyte = 0x29;
		x= mybyte & 0x0f; ----- Mask lower 4 bits.
		P1=x;
		y= mybyte & 0xf0; ------------Mask upper 4 bits.
		y= y >> 4; ------------Shift it to lower 4 bits.
		P2 = y;
		}
			Any correct program : 4M

6		**Attempt any FOUR of the following :**	**16 M**
	a	**Write a program to toggle the LED connected to P1.7 on every occurrence of external interrupt INTO.**	**4 M**
	Ans	ORG 0000H LJMP MAIN ; ISR for hardware external interrupt.	Correct Program : 4M

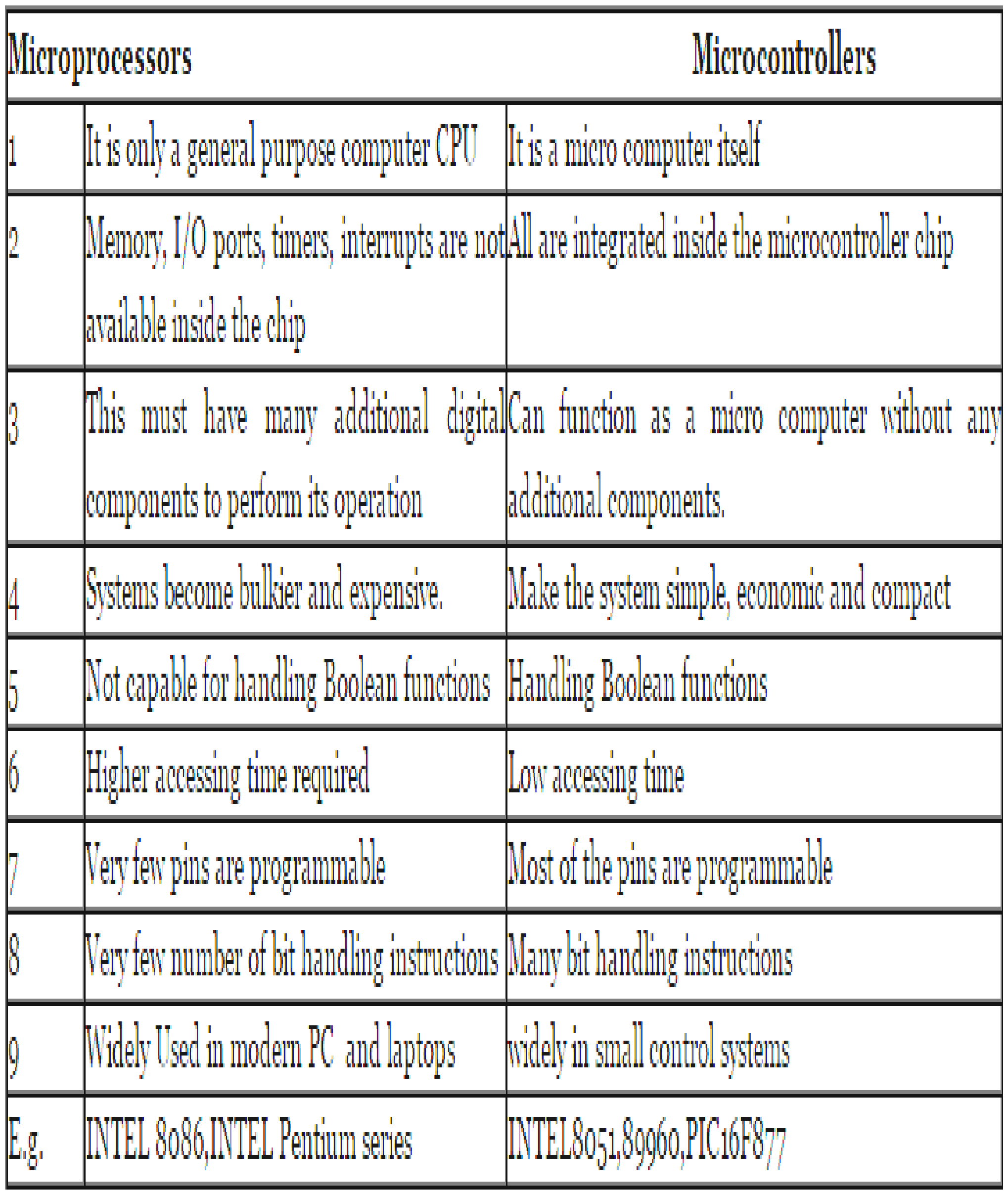
		ORG 0003H SETB P1.7 ; Turn on the LED. MOV R0, 200 ; Delay WAIT: DJNZ R0, WAIT CLR P.7 ; Turn off the LED RETI ORG 30H MAIN: SETB IT0 MOV IE, #84H WAIT2: SJMP WAIT2 END	
	b	**Explain deadlock. How it can be avoided.**	**4 M**
	Ans	A deadlock is a situation where in two or more competing actions are each waiting for the other to finish, and thus neither ever does. In computer science, deadlock refers to a specific condition when two or more processes are each waiting for the other to release a resource, or more than two processes are waiting for resources in a circular chain. A deadlock, also called as deadly embrace, is a situation in which two threads are each unknowingly waiting for resource held by other. • Assume thread/process T1 has exclusive access to resource R1. • Thread/ process T2 has exclusive access to resource R2. • If T1 needs exclusive access to R2 and T2 needs exclusive access to R1, • Neither thread can continue. • They are deadlocked. 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. Following is a deadlock situation between two tasks.	Deadlock : 3M Avoidance : 1M

		 Resource 1 Assigned to Waiting For Process 1 Process 2 Waiting For Resource 2 Assigned to In this example, process 1 wants the resource 2 ex; scanner while holding the resource 1 ex: printer. Process 1 cannot proceed until both the printer and the scanner are in its possession. Process 2 wants the printer while holding the scanner. Process 2 cannot continue until it has the printer and the scanner. Because neither process 1 nor process 2 is willing to give up what it already has, the two tasks are now deadlocked because neither can continue. The simplest way 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	
	c	**Why 8051 is known as Boolean processor? Explain with suitable instructions**.	**4 M**
	Ans	The 8051 instruction set is optimized for the one-bit operations so often desired in real-world, real-time control applications. The Boolean processor provides direct support for bit manipulation. This leads to more efficient programs that need to deal with binary input and output conditions inherent in digital-control problems. Bit addressing can be used for test pin monitoring or program control flags. It has its own 1 bit internal RAM of 16 bytes ranging from 20 H to 2F H which can be addressed at bit level. Also it has single bit manipulation instructions. For example, instructions for Boolean function are as given below: (a) ORL P0, #1; Set P0.0 (b) XRL P0, #1; Toggle P0.0 (c) ANL C, P1.4; AND the bit on P1.4 with carry	Boolean Processor : 2M Instructions : 2M

		(d) ANL C,! (P1.4); AND inverted bit on P1.4 with carry	
	d	**State various advantages and disadvantages of embedded systems.**	**4 M**
	Ans	<u>Advantages:</u> a. Cost is low. b. Small in size. c. Highly reliable. d. Operation is fast. e. Easy for mass production. f. Less interconnection. g. Optimizes use of system resources. h. Improves product quality. <u>Dis-Advantages:</u> a. Hard for maintenance as it is use and throw device. b. No technological improvement. c. Hard to take backup of embedded files. d. Less power supply durability if it is battery operated.	Advantages : 2M Disadvantages : 2M
	e	**Draw the interfacing diagram of seven segment multiplexed display with 8051 microcontroller.**	**4 M**

	Ans		Diagram : 4M

	f	**State the difference between microprocessor and microcontroller. (Any 4 points)**	**4 M**
	Ans		Any four points : 1M each



Microprocessors		Microcontrollers
1	It is only a general purpose computer CPU	It is a micro computer itself
2	Memory, I/O ports, timers, interrupts are not available inside the chip	All are integrated inside the microcontroller chip
3	This must have many additional digital components to perform its operation	Can function as a micro computer without any additional components.
4	Systems become bulkier and expensive.	Make the system simple, economic and compact
5	Not capable for handling Boolean functions	Handling Boolean functions
6	Higher accessing time required	Low accessing time
7	Very few pins are programmable	Most of the pins are programmable
8	Very few number of bit handling instructions	Many bit handling instructions
9	Widely Used in modern PC and laptops	widely in small control systems
E.g.	INTEL 8086,INTEL Pentium series	INTEL8051,89960,PIC16F877