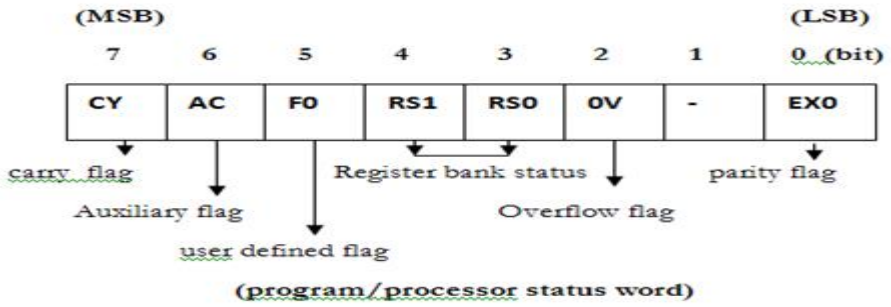
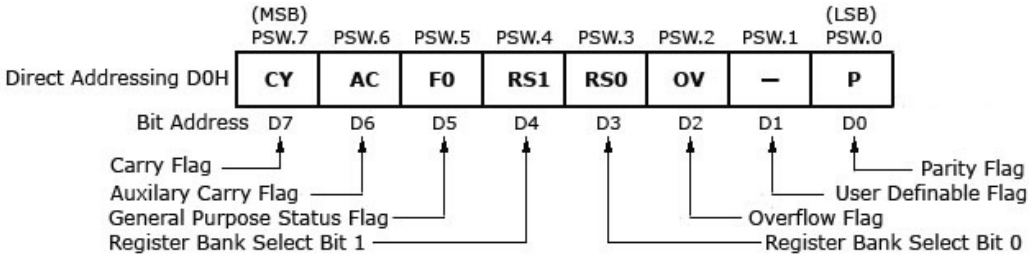
SUMMER– 18 EXAMINATION

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.	Answers	Marking Scheme
1.	**a)**	**Attempt any three of the following:**	**12 Marks**
	i)	**Explain the following pins of 8051 microcontroller.** **a)$\overline{\text{PSEN}}$ b) ALE c) $\overline{\text{EA}}$ d) RESET**	**4M**
	Ans:	a) **PSEN(bar):** Pin 29 is the Program Store Enable Pin (PSEN). Using this pins, external Program Memory can be read. b) **ALE** : Pin 30 is the Address Latch Enable Pin. Using this Pins, external address can be separated from data (as they are multiplexed by 8051). c) **EA(bar):** Pin 31 is the External Access Enable Pin i.e. allows external Program Memory. Code from external program memory can be fetched only if this pin is LOW. For normal operations, this pins is pulled HIGH. d) **Reset :** Pin 9 is the Reset Input Pin. It is an active HIGH Pin i.e. if the RST Pin is HIGH for a minimum of two machine cycles, the microcontroller will be reset. During this time, the oscillator must be running.	**(Each: 1 mark)**

	ii)	**State the purpose of the following branch instructions in 8051 microcontroller.** **a) AJMP add b) LJMP add c) SJMP add d) JC Label**	**4M**
	Ans:	**a) AJMP, Add:** AJMP unconditionally jumps to the indicated **code address**. The new value for the Program Counter is calculated by replacing the least-significant-byte of the Program Counter with the second byte of the AJMP instruction, and replacing bits 0-2 of the most-significant-byte of the Program Counter with 3 bits that indicate the page of the byte following the AJMP instruction. Bits 3-7 of the most-significant-byte of the Program Counter remain unchanged. **b) LJMP add:** The **LJMP** instruction transfers program execution to the **specified 16-bit address**. The PC is loaded with the high-order and low-order bytes of the address from the second and third bytes of this instruction respectively. No flags are affected by this instruction. **c) SJMP add:** The **SJMP** instruction transfers execution to the **specified address.** The address is calculated by adding the signed relative offset in the second byte of the instruction to the address of the following instruction. The range of destination addresses is from 128 before the next instruction to 127 bytes after the next instruction. **d) Jc label: The JC** instruction branches to the specified address if the carry flag is set. Otherwise, execution continues with the next instruction. No flags are affected by this instruction.	**(Each: 1 mark)**
	iii)	**Draw the PSW register and explain each bit in detail.**	**4M**
	Ans:	**PSW FORMAT:**  **OR**  	**(Draw PSW format: 2 marks, Explanation: 2 marks)**

Explanation: 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.

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". More about these and other bit-addressable instructions will be given in Chapter 8.

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. See Chapter 6 for more information.

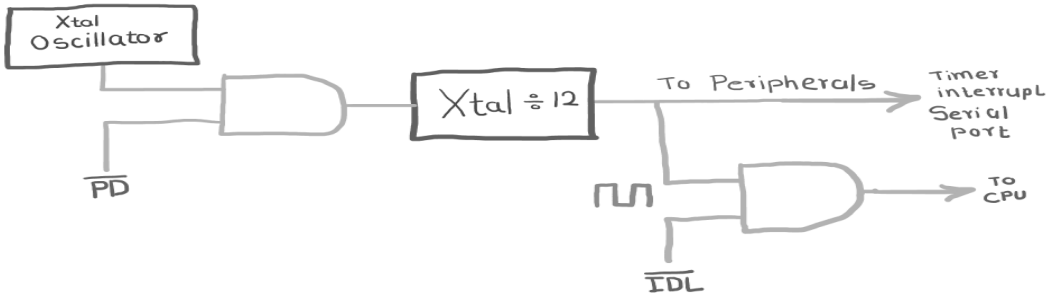
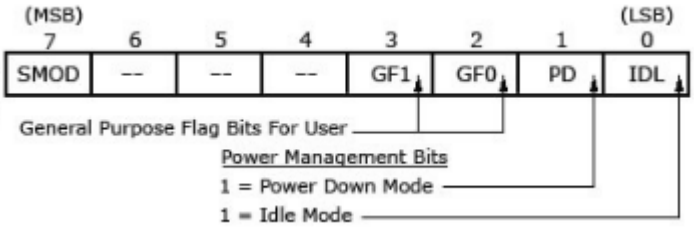
P, the parity flag:

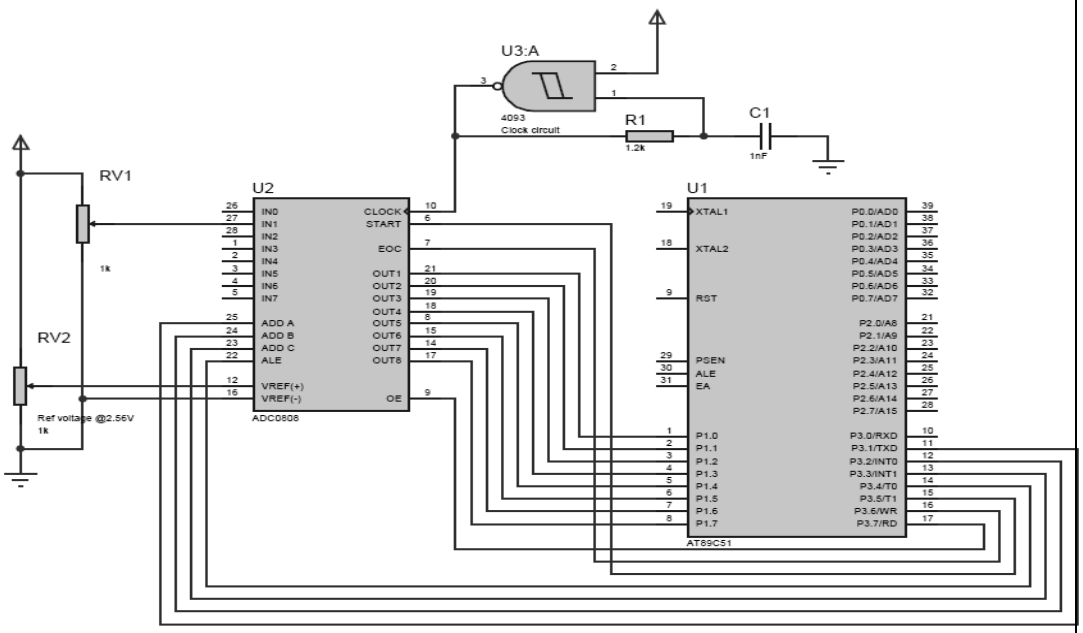
The parity flag reflects the number of 1 s in the A (accumulator) register only. If the A register contains an odd number of 1s, then P = 1. Therefore, P = 0 if A has an even number of 1s.

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. The overflow flag is only used to detect errors in signed arithmetic operations.

RS 1	RS 0	Register Bank
0	0	Register bank 0
0	1	Register bank 1
1	0	Register bank 2
1	1	Register bank 3

	iv)	**Explain power saving operation of 8051 microcontroller.**	**4M**
	Ans:	Control logic diagram  **Explanation:** 8051 has two power saving mode, • Power Down Mode • Idle Mode Two control bits are there, IDL and PD, which are used for Idle and Power down mode respectively. In **Power Down** mode, the oscillator clock provided to system is OFF i.e. CPU and peripherals clock remains inactive in this mode.In **Idle** Mode, only the clock provided to CPU gets deactivated, whereas peripherals clock will remain active in this mode. Hence power saved in power down mode is more than in idle mode.8051 has power control register for power control. **PCON Register: Power control register:** PCON (Power control) register is used to force the 8051 microcontroller into power saving mode. Power control register of 8051 contains two power saving mode bits and one serial baud rate control bit.  	**(Diagram:1 mark, PCON Format: 1 mark ,Operation: 2 marks)** **PCON explanation is not required.**

	b)	**Attempt any one of the following:**	**06 Marks**
	i)	**Draw interfacing diagram of ADC with 8051 microcontroller. Write assembly or C language program to convert analog input data into digital using ADC for 8051.**	**6M**
	Ans:	 **C program**	**(C Program: 3 marks, ALP: 3 marks)**

```
#include <reg51.h>
 #define ALE            P3_4
#define OE              P3_7
#define START           P3_5
#define EOC             P3_6
#define SEL_A           P3_1
#define SEL_B           P3_2
#define SEL_C           P3_3
#define ADC_DATA        P1
 void main()
{
        unsigned char adc_data;

        /* Data port to input */
        ADC_DATA = 0xFF;

        EOC = 1; /* EOC as input */
```

```
        ALE = OE = START = 0;

        while (1) {
                /* Select channel 1 */
                SEL_A = 1; /* LSB */
                SEL_B = 0;
                SEL_C = 0; /* MSB */

                /* Latch channel select/address */
                ALE = 1;
                /* Start conversion */
                START = 1;
                ALE = 0;
                START = 0;
                /* Wait for end of conversion */

                while (EOC == 1);
                while (EOC == 0);
                /* Assert Read signal */
                OE = 1;
                /* Read Data */
                adc_data = ADC_DATA;
                OE = 0;
                /* Now adc data is stored */
                /* start over for next conversion */
        }
}
```

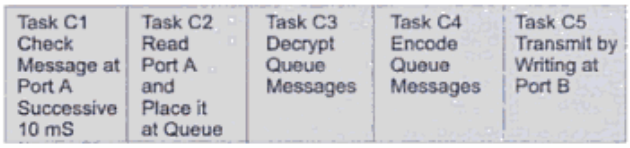
OR

8051 Assembly language program

```
ALE EQU P3.4
OE EQU P3.7
START EQU P3.5
EOC EQU P3.6
SEL_A EQU P3.1
SEL_B EQU P3.2
SEL_C EQU P3.3
ADC_DATA EQU P1
ORG 0H
;DATA PORT TO INPUT
MOV ADC_DATA, #0FFH
;EOC AS INPUT
```

```
        SETB EOC
        ;REST OF OUTPUT SIGNALS
        CLR ALE
        CLR OE
        CLR START

MAIN_LOOP:
        ;SELECT ANALOG CHANNEL 1
        SETB SEL_A
        CLR SEL_B
        CLR SEL_C
        ;LATCH CHANNEL SELECT
        SETB ALE
        ;START CONVERSION
        SETB START
        CLR ALE
        CLR START
        ;WAIT FOR END OF CONVERSION
        JB EOC, $ ; $ MEANS JUMP TO SAME LOCATION
        JNB EOC, $
        ;ASSERT READ SIGNAL
        SETB OE
        ; Read Data
        MOV A, ADC_DATA
        CLR OE
      ADC DATA IS NOW IN ACCUMULATOR
        ;START OVER FOR NEXT CONVERSION
        SJMP MAIN_LOOP
        END
```

	ii)	**Describe the concept of multitasking In RTOS.**	**6M**
	Ans:	**Scheduling of Multiple Tasks in Real Time by RTOS** An RTOS lets the system schedule the various tasks in real time. A real time system responds to th event within a bound time limit and within an explicit time. A scheduler for the time-constrained task: can be understood by a simple example.  Task C1 to Task C5 **Fig a Five task C1-C5**	**(Descriptions: 4 marks ,Diagram: 2 marks)**

Time	Processor Context	Saved Context	Task C1	Task C2	Task C3	Task C4	Task C5
0-10 ms	Task C1		—				
10-13 ms	Task C2		✓	—			
13-15 ms	Task C3	C2	✓	?	—		
15-17 ms	Task C4	C2,C3	✓	?	?	—	
17-20 ms	Task C5	C2,C3,C4	✓	?	?	?	—

Started/Initiated
Blocked after Saving Context ?
Running —
Finished ✓

Time Slicing Scheduling by the RTOS Kernel

Fig b Task program with schedule

Suppose after every ten milliseconds, there is a stream of coded messages reaching at port A of an embedded system, it is then read and decrypted and retransmitted to the port after encoding each decrypted message. The multiple processes consist of five tasks, C1, C2, C3, C4 and C5, as follows:

1. Task C1: Check for a message at port A at every 10 ms.
2. Task C2: Read Port A and put the message in a message queue.
3. Task C3: Decrypt the Message from message queue.
4. Task C4: Encode the Message from the queue.
5. Task C5: Transmit the encoded message from the queue to Port B.

Figure (a) shows five tasks, C1 to C5, that are to be scheduled. Figure (b) shows the five contexts in five time schedules, between 0 to 10 ms, 10 to 13 ms, 13 to 15 ms, 15 to 17 ms and 17 to 20 ms, respectively. Let RTOS initiate C1 to C5. Let there be RTC tick interrupts at each ms. Task C1 is scheduled by RTOS to bring it in running state from its blocked state as soon as a timer triggers an event. If it is known that after every ten millisecond a byte reaches port A, let a timer, RTCSWT, trigger an event every 10 ms. Task C1 finishes after 10 ms, and C2 starts running.

Figure (b) shows at the different time slices, the real time schedules, saved contexts and processor contexts.

(Note: Any other multitasking example can be consider**)**

2.		**Attempt any four of the following:**	**16 Marks**
	a)	**Explain the following instructions and what will be the content of Register A and B after the execution of following instructions?** **MOV A, # 38 H** **ADD A, # 54 H** **DA A** **MOV B, A**	**4M**
	Ans:		**(Each step: ½ mark, result: 1 mark each)**

Instruction	Explanation
MOV A,#38H	Move 38H immediately in to A register
Add A,#54H	Add 54H with content of A and store result in A register
DAA	Add correction factor 6 with result if carry generates or in valid result
MOV B,A	Store the result in to B register
A register has : 92 H & B register has 92H	

	b)	**Explain the assembler directives.** **i) DB ii) ORG iii) EQU iv) END**	**4M**
	Ans:		**(1 mark each)**

DB (define byte):
The DB directive is the most widely used data directive in the assembler. It is used to define the 8-bit data. When DB is used to define data, the numbers can be in decimal, binary, hex, or ASCII formats. For decimal, the "D" after the decimal number is optional, but using "B" (binary) and "H" (hexadecimal) for the others is required. To indicate ASCII, simply place the characters in quotation marks
examples:

```
          ORG  500H
DATA1:    DB   28                 ;DECIMAL(1C in hex)
DATA2:    DB   00110101B          ;BINARY (35 in hex)
DATA3:    DB   39H                ;HEX
          ORG  510H
DATA4:    DB   "2591"             ;ASCII NUMBERS
          ORG  518H
DATA6:    DB   "My name is Joe"   ;ASCII CHARACTERS
```

2)ORG (origin):
The ORG directive is used to indicate the beginning of the address. The number that comes after ORG can be either in hex or in decimal. If the number is not followed by H, it is decimal and the assembler will convert it to hex.
Example ORG 0030H
3)EQU (equate):
This is used to define a constant without occupying a memory location. The EQU

directive does not set aside storage for a data item but associates a constant value with a data label so that when the label appears in the program, its constant value will be substituted for the label. The following uses EQU for the counter constant and then the constant is used to load the R3 register.

```
COUNT      EQU  25
...        ....
MOV        R3,#COUNT
```

4)END directive:
Another important pseudo code is the END directive. This indicates to the assembler the end of the source (asm) file. The END directive is the last line of an 8051 program, meaning that in the source code anything after the END directive is ignored by the assembler.
Example:
ORG 00h
MOV A,#25H
END

c) Explain the following instructions of 8051 microcontroller. **4M**
i) XCH A, RI ii) ADD A, 40 H iii) DJNZ R_n , add iv) ADD A, # 40H

Ans: **(1 mark each)**

i)XCH A, R1:

Exchanges the value of the Accumulator with the value contained in *registerR1*

(A) $\rightleftarrows$ (R1)

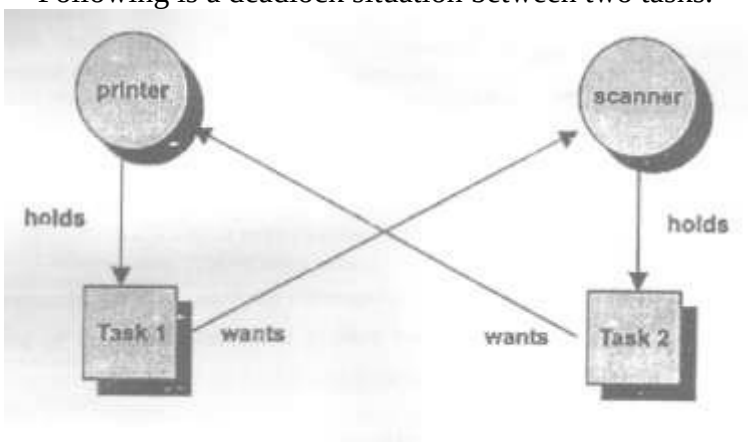
ii) ADDA,40H :
Adds the content of data memory 40H or direct byte from data memory location 40H to the accumulator
[A+ (40)]⟶[A]

iii) DJNZ Rn, Add:

DJNZ decrements the value of register by 1. If the initial value of register is 0, decrementing the value will cause it to reset to 255 (0xFF Hex). If the new value of register is not 0 the program will branch to the address indicated by relative addr. If the new value of register is 0 program flow continues with the instruction following the DJNZ instruction.

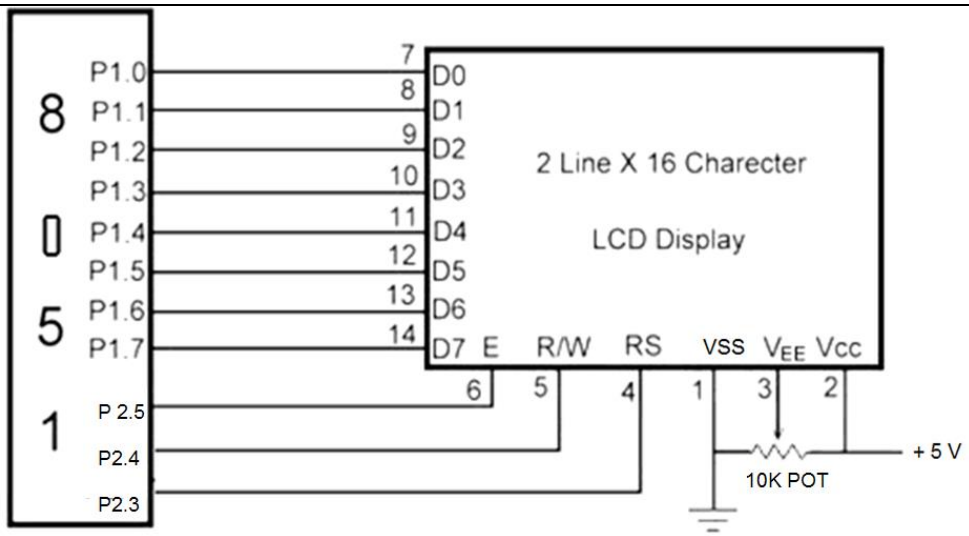
iv)ADD A,#40H :

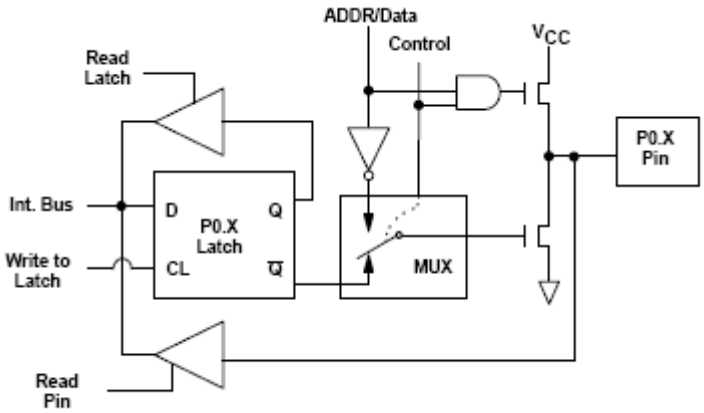
add immediate 8 bit data 40H with A and store the result in A
Operation: (A) ← (A) + #40H

	d)	**What is a dead lock? How it can be prevented?**	**4M**
	Ans:	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 T1 has exclusive access to resource R1. • Thread 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. **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 • 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.  **We can prevent Deadlock by eliminating any of the following** • Mutual Exclusion • Hold and Wait • No preemption • Circular wait.	**(Dead lock: 2 marks, Prevention : 2 marks)**

	e)	**State any eight features of 8051 micro controller.**	**4M**
	Ans:	1. 8bit CPU 2. 16-bit Program Counter 3. 8-bit Processor Status Word (PSW) 4. 8-bit Stack Pointer 5. Internal RAM of 128bytes 6. Special Function Registers (SFRs) of 128 bytes 7. 32 I/O pins arranged as four 8-bit ports (P0 - P3) 8. Two 16-bit timer/counters : T0 and T1 9. Two external and three internal vectored interrupts 10. One full duplex serial I/O 11. 4 KB on chip program memory.(ROM) 12. 8-bit data bus 13. 16-bit address bus 14. Bit as well as byte addressable RAM area of 16 bytes. 15. Microsecond instruction cycle with 12 MHz Crystal.	**(Any eight Features:1/2 mark each)**
	f)	**List the alternate functions of 8051 port 3 pins.**	**4M**
	Ans:		**(Function of each Correct port pin: 4 marks)**

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

3.		**Solve any four of the following:**	**16 Marks**
	a)	**Draw neat labelled diagram to interface 16x2 LCD with 8051 and explain.**	**4M**
	Ans:	 8 data pins of LCD are connected to Port 1 of 8051. Control pins RS, R/W & E are connected to port pins P2.3, P2.4 & P2.5 respectively. **RS - register select:** If RS = 0, the instruction command code register is selected, If RS = 1 the data register is selected, **R/W - read/write:** R/W input allows the user to write information to the LCD or read information from it. R/W = 1 when reading; & R/W =0 when writing. **E - Enable:** The enable pin is used by the LCD to latch the information present on the data pins. A high-to low pulse is needed to latch the data. This pulse must be a minimum of 450 ns wide.	**(Explanation: 2 marks, Diagram: 2 marks)**

	b)	**Draw the structure of PORT 0 of 8051. Why pull up resistors are required here?**	**4M**
	Ans:	 **Port0 pin circuit** The pins of P0 are connected internally to an "open drain" circuit (similar to open collector but using MOS transistors). Therefore, it must be connected to an external pull-up resistor to operate properly as an output port	**(Structure: 2 marks, Requirement : 2 marks)**
	c)	**Write a program in assembly or C language to generate a square wave of 10 KHz on pin P2.4 of 8051 using timer 0.**	**4M**
	Ans:	**Initial count Calculations:** **Assume XTAL frequency is 11.0592 MHz.** Frequency = 10 KHZ Therefore Time period T = 1 /10KHZ = 0.1 ms Therefore Required time delay = 0.1ms / 2 = 0.05 ms = 50usec. Required time delay = (12 / Fosc) x number of increments (N) 50us = (12 / 11.0592MHZ) x number of increments (N) 50us = 1.085 usec. x N N = 46 Using TIMER 0 in MODE 1, INITIAL COUNT = 2^{16} - N INITIAL COUNT = 65536 – 46 = 65490 = FFD2 H **Therefore TH1 = 0XFF & TL1 =0XD2**	**(Count calculation: 1 mark, Program: 3 marks)**

ASSEMBLY PROGRAM

```
PROGRAM:
        ORG 0000H
         MOV TM0D,#01H      ;TIMER 0, MODE 1
AGAIN:MOV TH0,#0FFH         ;Load higher byte of count
         MOV TL0,#0D2H      ; load lower byte of count
         SETB TR0           ;start timer 0
HERE:  JNB TF0,HERE         ;CHECK IF TF0 IS SET
        CPL P2.4            ; Complement P2.4
        CLR TR0             ; IF TF0 =1 , STOP TIMER 0
        CLR TF0             ; CLEAR TF1
        SJMP AGAIN          ;REPEAT AGAIN
        END
```

OR

'C' LANGUAGE PROGRAM

```
#include <reg51.h>
void T0M1Delay(void);
sbit SQR=P2^4;
void main(void)
{
while (1)          // repeat forever
{
SQR = ~SQR ;    // Complement  bit P2^4
T0M1Delay( ) ;   // delay of 50 usec.
}                  // end of while
}                  // end of main

void T0M1Delay(void)
{
TMOD=0x01;//timer0 mode1
TH1=0XFF; // Load higher byte of count
TL1=0XD2; // load lower byte of count
TR0=1; // start timer1
while (TF0==0); // wait for Timer to overflow
TR0=0; // stop timer1
TF0=0; // clear  TF1
}
```

(Note: any other program logic can be given marks. Also count calculation with xtal frequency 12 MHz can be considered. **)**

d) Explain the features of RTOS. How it differ from general operating system? **4M**

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: ½ mark each, Difference(Any four points): ½ mark)

Sr.no.	General / Desktop O.S.	RTOS
1.	Does not have deterministic time response	It has deterministic time response
2.	Generalized kernel	Real time kernel
3.	There is no task deadline	There is task deadline in RTOS
4.	Memory required depends on version	Memory required(footprint) is less
5.	Applications are compiled separately from O.S.	Applications are compiled and link with RTOS
6.	It is used in general desktop computer	It is used in embedded systems
7.	It is less reliable	It is more reliable
8.	e.g. Windows, Linux	e.g. RTLinux, VXWorks, Micro C

	e)	**State any 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	**Any eight-(½ mark Each application)**
	f)	**Explain with example what do you mean by share data problem? How it is avoided?**	**4M**
	Ans:	• A real time operating system intended to serve real time application process data in real time; with processing time in milliseconds. • Essentially, RTOS has an advanced algorithm for scheduling multiple tasks. One of the key expectations from RTOS is to be able to quickly and predictably respond to multiple tasks. • The challenge of running multiple tasks is of sharing of common resource; it could be memory, I/O address or device. •For example If task 1 calls a function Read X for reading a shared data that is being interrupted and being modified by task 2, there is a chance that data read by task 1 is erroneous. • If access to shared resource is not synchronized, it may lead to unpredictable behavior of the task. • If a variable is used in two different processes and another task interrupts before the operation on that data is completed, then the value of the variable may differ from the one expected if the earlier operation has been completed. This condition is known as shared data problem. Methods to avoid shared data problem: 1.Semaphore 2. mailbox 3.mutex 4.message queue	**(Explanation: 2 marks,** **Example: 1 mark,** **How to avoid: 1 mark)**

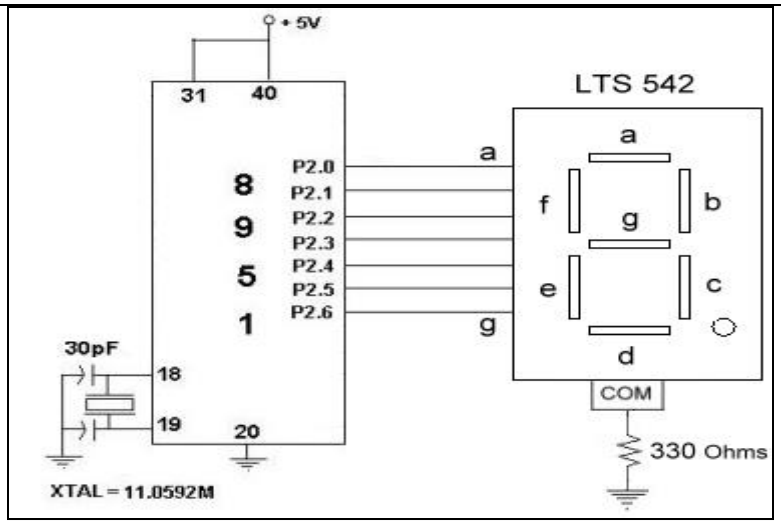
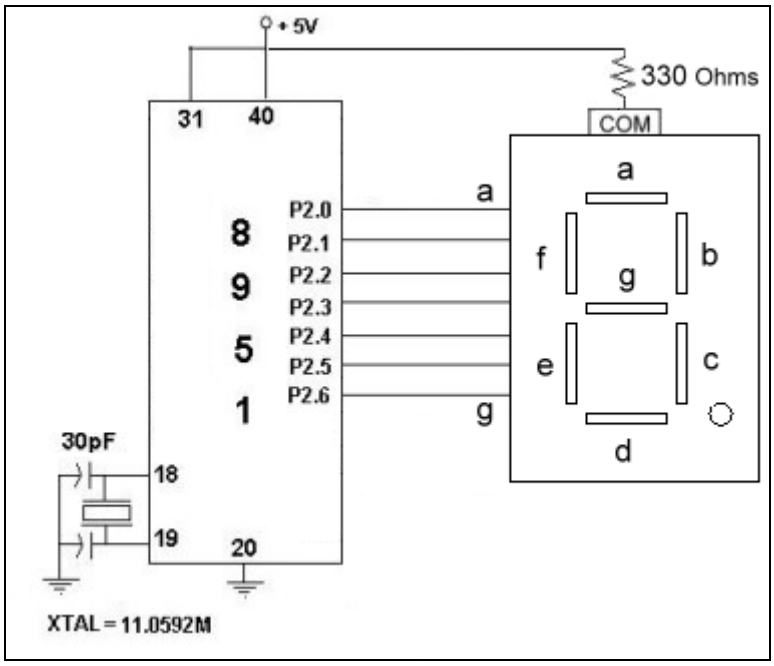
4.	**a)**	**Attempt any three:**	**12 Marks**
	i)	**With neat sketch explain the interfacing of seven segment display with 8051 microcontroller.**	**4M**
	Ans:	 **Fig: Common cathode 7-Segment Display Interfacing with 8051** **OR**  **Fig: Common Anode 7-Segment Display Interfacing with 8051**	**(Correct Diagram: 4 marks)**

		Diagram shows common cathode seven segment display interfaced with 8051. Segments are connected to port 1 of 8051. Common pin of display is grounded. To turn ON the segment port pin should output logic high. **(**Note: common anode display can also be interfaced. Accordingly common pin should be connected to VCC (+5v dc) and port pins should output logic 0 to turn on the segment**)**	
	ii)	**Explain various debugging tools used in embedded system.**	**4M**
	Ans:	**1) Emulator :** An emulator is hardware or software or both that duplicates (or *emulates*) the functions of one system using a different system, so that second system behaves like the first system. Emulation refers to the ability of a computer program in an electronic device to emulate (imitate) another program or device. A hardware emulator is an emulator which takes the form of a hardware device. Emulation tricks the running software into believing that a device is really some other device. Emulator uses the circuit consisting of the microcontroller or processor itself. The emulator emulates the target system with extended memory and with codes downloading ability. **2) In-Circuit Emulator (ICE):** An in-circuit emulator (ICE) is a hardware device used to debug the software of an embedded system. The ICE is temporarily installed between the embedded system and an external terminal or personal computer so that the programmer can observe and alter what takes place in the embedded system. ICE uses another circuit with a card that connects to target processor through a socket. ICE provides greater flexibility and ease for developing various applications on a single system instead of testing multiple targeted systems **3) Simulator:** Simulator is a development and debugging tool It is a software that simulates (imitates) all operations of microprocessor /microcontroller. **.**It provides interactive method for developing programs. It provides detailed information of all internal registers, memory and peripherals on monitor. It provides facility to run single step through source program It allows to view/edit contents of registers/memory **.**It allows to set multiple breakpoints **.**It simulates the inputs from interrupts , timers,ports, peripherals. It also simulates real time processes **4) Debugger:** It is a special program used to find errors (*bugs*) in other programs(target program). A debugger allows a programmer to stop a program at any point and examine and change the values of variables.It includes features like simple and complex breakpoints, watch windows, execution control , logic analyzer etc.eg. Keils µVision	**(Any four Debugging Tools: 4 marks)**

Debugger

5) JTAG port:
It is debugging tool to debug programs running on target system. Joint Access Group(JTAG) standardized a mechanism for providing debugging through a port called JTAG port. JTAG port provides access to the internals of processor. The standard IEEE1149.1a-1993 gives details of protocol used in JTAG port. Using a technique called Boundary Scan the connections between processor and the memory/peripherals can be probed by giving appropriate signals at output pins and reading response from input pins.

iii) State four features of embedded system. **4M**

Ans: **(any four-1 mark Each feature)**

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 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.

iv) State the Difference between microcontroller and microprocessor (any four). **4M**

Ans: **(Any four- 1 mark Each difference)**

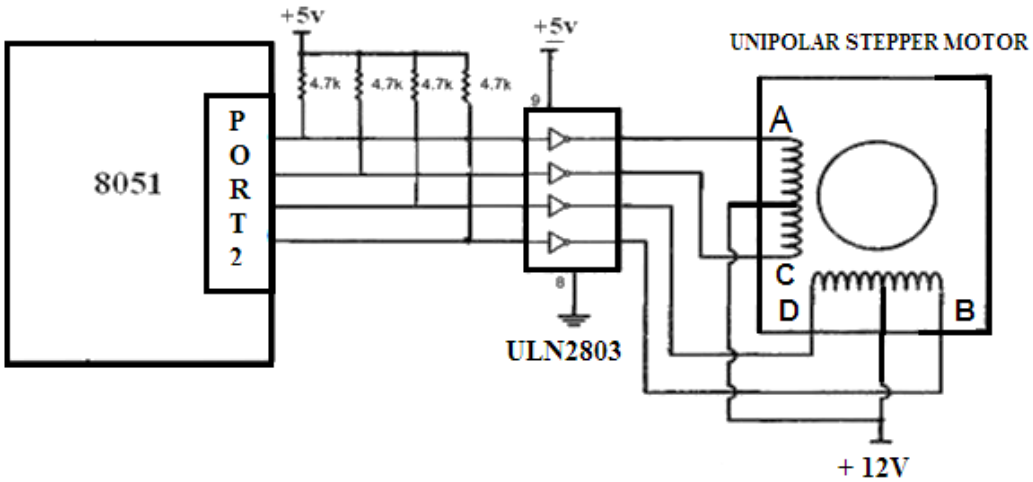
S.No	Microprocessor	Microcontroller
1	A microprocessor is a general purpose device which is called a CPU	A microcontroller is a dedicated chip which is also called single chip computer.
2	A microprocessor do not contain on chip I/O Ports, Timers, Memories etc..	A microcontroller includes RAM, ROM, serial 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.

b) Attempt any one of the following: 06 Marks

i) Draw interfacing diagram of stepper motor with 8051 microcontroller. Write ALP to rotate stepper motor in anti clockwise direction continuously using full step sequence. 6M

Ans: **(Inter facing: 3 marks, Program: 3 marks)**



Full step excitation sequence:

STEP	P2.3	P2.2	P2.1	P2.0	Hex value
	A	B	C	D	
1	1	0	0	1	99 H
2	1	1	0	0	CC H
3	0	1	1	0	66 H
4	0	0	1	1	33H

- **Program to rotate motor continuously in Anti-clockwise direction**

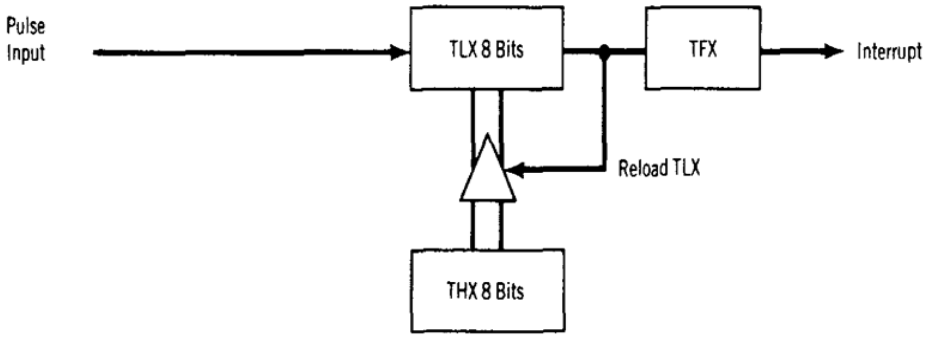
ASSEMBLY LANGUAGE PROGRAM

```
 ORG 0000H
UP:MOV P2 ,#33H  ; Send first step sequence to Port 2
   ACALL DELAY
   MOV P2 ,#66H   ; Send second step sequence to Port 2
   ACALL DELAY
   MOV P2 ,#0CCH  ; Send third step sequence to Port 2
   ACALL DELAY
   MOV P2 ,#99H   ; Send fourth step sequence to Port 2
   ACALL DELAY
   SJMP UP
   DELAY SUBROUTINE
DELAY:MOV R1,#0FFH
THERE:MOV R2,#0FFH
HERE:DJNZ R2 , HERE
     DJNZ R1 , THERE
      RET
```

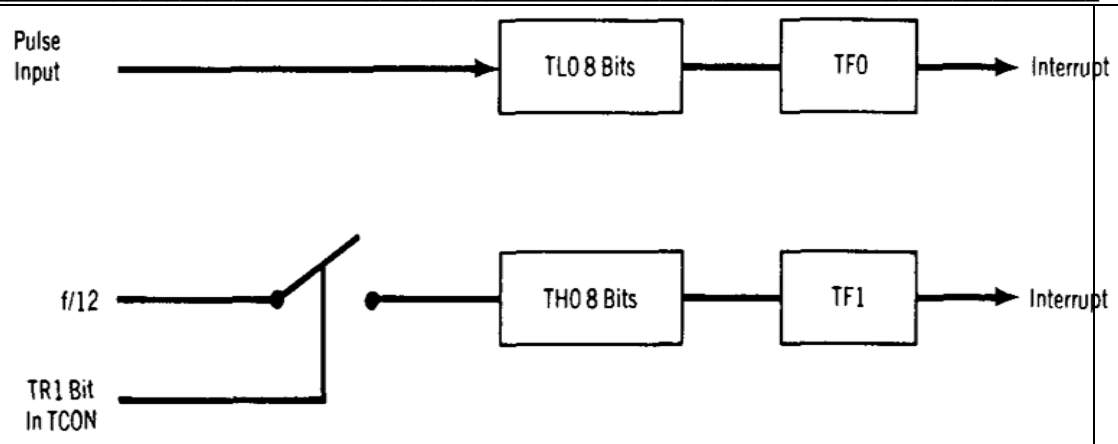
	ii)	**Explain with diagram four timer modes in 8051.**	**6M**
	Ans:	**There are four timer / counter modes:** **1. MODE 0 - 13 bit Timer / Counter:** In mode 0, the timer/counter is configured as a 13-bit timer/counter. The upper 8 bits of the count are in TH .The lower 5 bits are in the lower 5 bits of TL. The upper 3 bits of TL are not used. The TFx flag will be set when the timer /counter Overflows from all 1's to all 0's. The timer continues to count. 	**(Explanation of each mode:1 and ½ mark)**

2. **MODE 1- 16 bit Timer / Counter:** In mode 1, the timer/counter is configured as a 16-bit timer/counter. The upper 8 bits of the count are in TH & The lower 8 bits are in TL. The TFx flag will be set when the timer / counter overflows from all 1's to all 0's. The timer continues to count.

3. **Mode 2 – 8 bit Auto Reload:**TL operates as an 8-bit Timer / counter.TH holds a reload value. When TL overflows (reached FFH), the TFx flag is set, TL is reloaded from the value in TH and counting continues.



4. **Mode 3- Split Mode:** Timer 0 is split into two independent 8-bit timers.TL0 acts as 8 bit Timer / Counter When TL0 overflows, it sets the TF0 flag.TH0 acts as 8 bit Timer ,When TH0 overflows, it sets the TF1 flag. Timer 1 is stopped in mode 3. It can be switched independently to a different mode. However, when it overflows it will NOT set the TF1 flag.



5.		**Attempt any four of the following:**	**16 Marks**
	a)	**Draw and describe IE SFR of 8051.**	**4M**
	Ans:	**Interrupt Enable register (IE)**: Byte Address: A8H , bit address A8H to AFH	**(Format: 2 marks, Explanation: 2 marks)**

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.

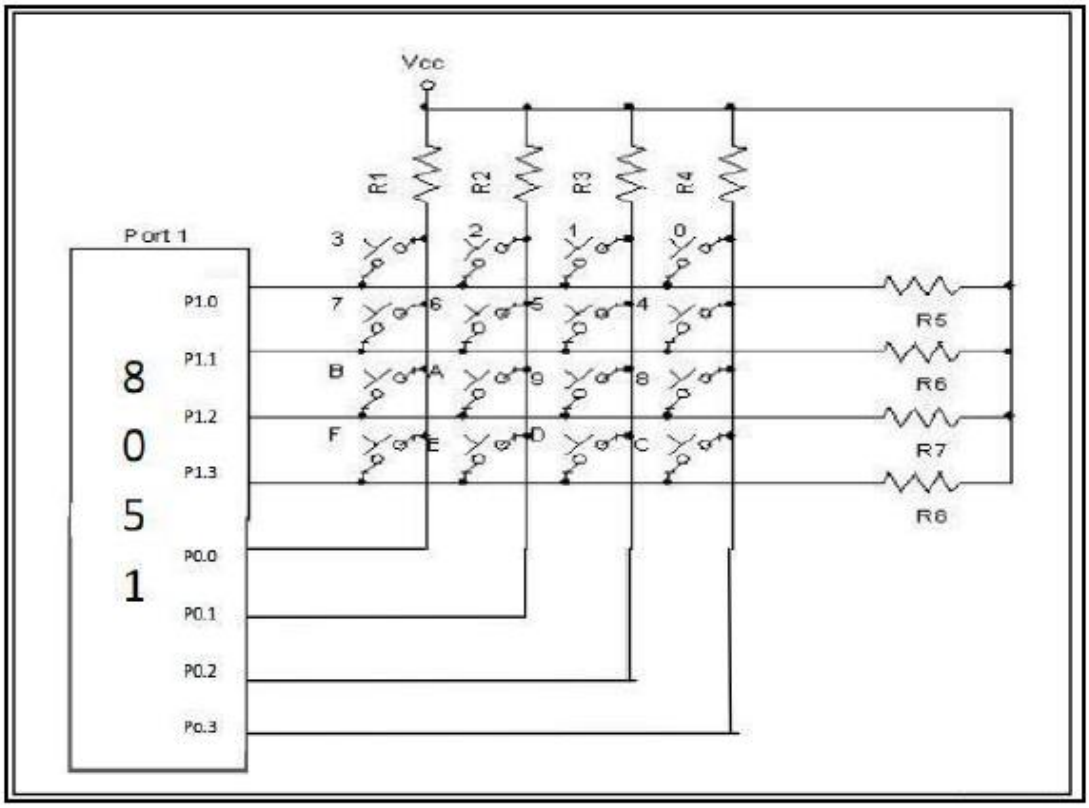
Bit0 to bit5 i.e. all the bits except EA bit are the local enable/ disable bit. When the bit is zero then the respective interrupt is disabled. And when the bit is set and EA is also set, then the respective interrupt is enabled. But if interrupt bit is set and EA=0, all the interrupts are disabled.

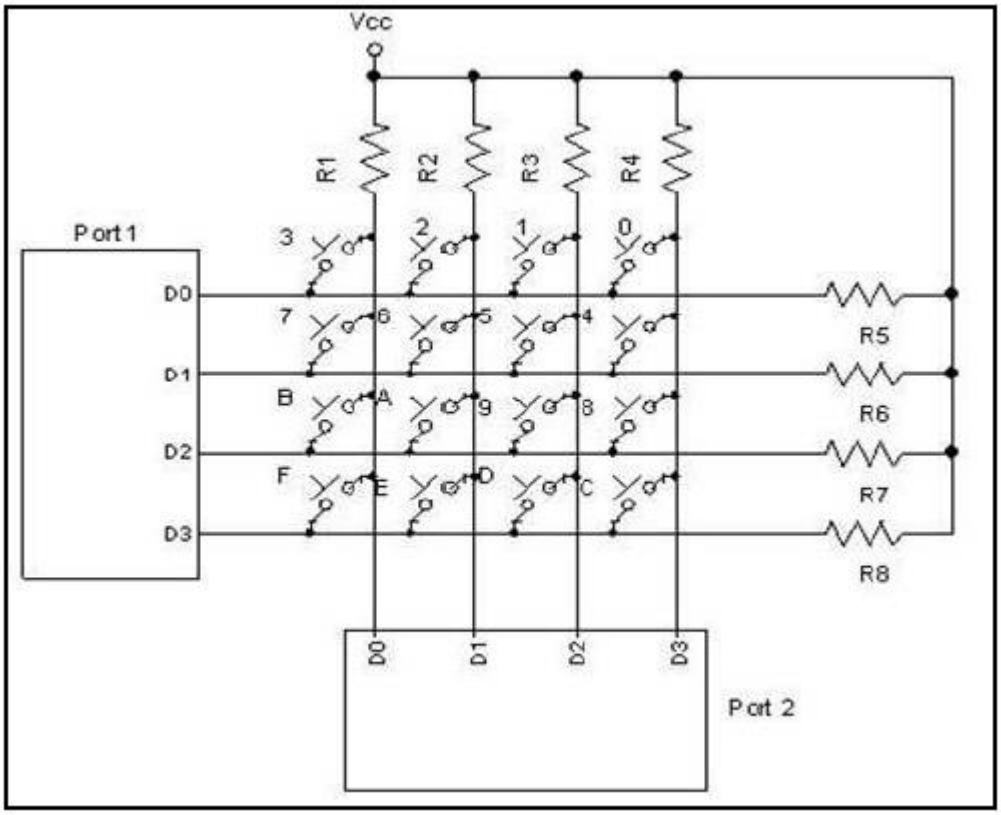
	b)	**Write ALP to move 10 bytes of data from internal RAM memory address 40H to internal RAM located from 50 H as starting address.**	**4M**
	Ans:	ORG 0000h : Starting address MOV R0,#40h : Source address MOV R1,#50h : destination address CLR A : Clear A MOV R2,#0Ah : Number of bytes to be transfer L1: MOV A,@R0 : Take first number from source in A MOV @R1,A : move first number to first destination INC R0 : increment source address INC R1 : increment destination address DJNZ R2,L1 : transfer all numbers till count becomes zero SJMP $: wait END **(**NOTE: Program may change. Please check the logic and understanding of students. **)**	**(Correct program: 4 marks,** **(Comments optional))**
	c)	**Write a program for serial data transfer. "G "at 9600 baud rate continuously.**	**4M**
	Ans:	**We are assuming crystal value 11.0592MHz.** Timer clock Frequency is= XTAL/12 = 11.0592MHz /12=921.6KHz UART Frequency= Timer clock Frequency/32 = 921.6KHz / 32= 28.8 KHz Baud rate = UART Frequency/ COUNTER Value COUNTER Value = UART Frequency/ Baud Rate = 28.8KHz / 9600 = 3 As timer in microcontroller is upcounter / timer, so counter value	**(4 Marks for correct program** **(Either assembly language)** **OR** **C language** **(students can also skip calculation for baud rate, can**

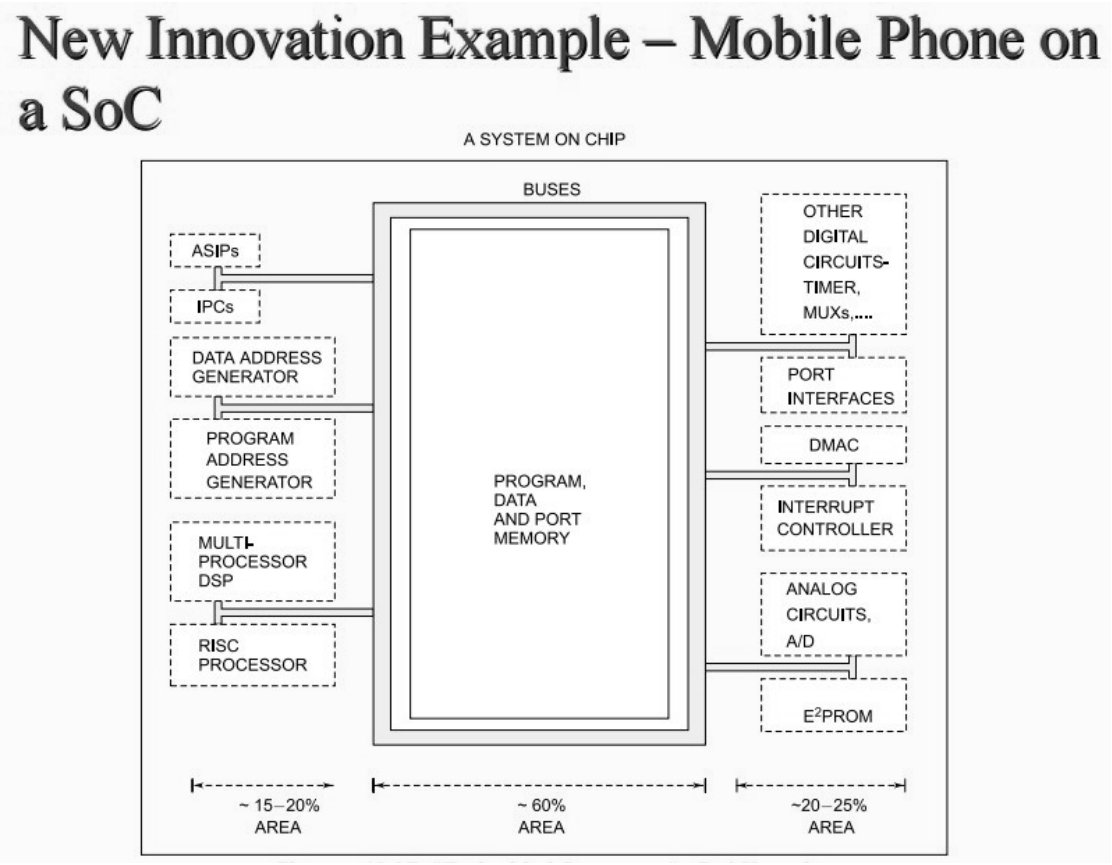
		= **-3**
		Hex count=255-3=252=**FDh**
		ORG 0000H
		MOV SCON, #50H ; serial port mode 1
		
		MOV TMOD, #20H ;timer mode 2 (configure timer 1 in auto-reload)
		MOV TH1, #-3 ;reload value for 9600 baud (or MOV TH1,#0FD)
		SETB TR1 ;start timer1
		LOOP: MOV SBUF, #'G' ;transmit character
		WAIT: JNB TI, **WAIT** ; wait for end of transmission
		CLR TI ;clear TI
		CLR TR1 stop timer1
		JMP **LOOP** ;re-transmit
		END
		OR
		(Using C language)
		28800 is the maximum baud rate of the 8051 microcontroller
		28800/9600= 3
		That baud rate '3' or hex 0FD 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=- 3; // load the baud rate (TH1 =0x0FD)
			directly load value in TH for given baud rate)

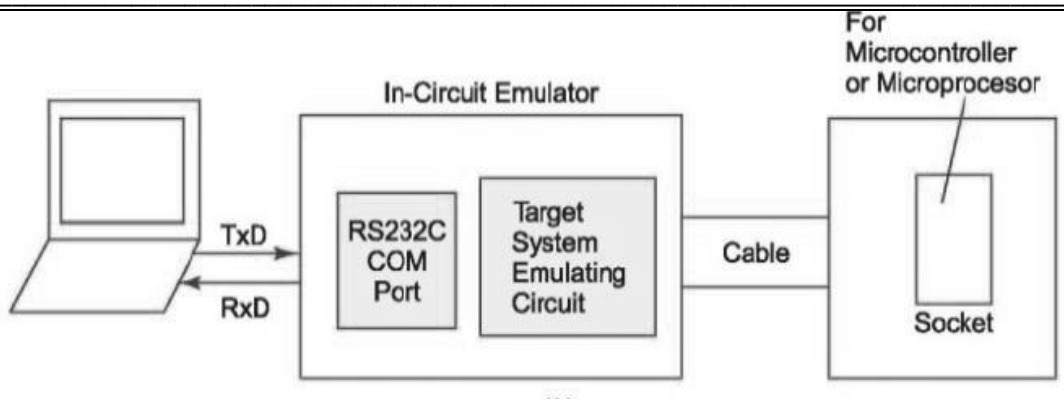
		TR1=1; //Timer ON
		SBUF = 'G'; // transmit character
		while(TI==0); //wait for end of transmission
		TI=0; // clear TI
		TR1=0; //OFF the timer
		while(1); //continuous loop
		}
		(NOTE: Program may change. Please check the logic and understanding of students.**)**	

	d)	**Describe the concept of device driver in embedded system.**	**4M**
	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 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.	**(Description: 4 marks)**

	e)	**Draw interfacing diagram of 4x4 matrix keyboard with 8051 microcontroller.**	**4M**
	Ans:	 **OR**	**(Correct diagram: 4 marks)**

			
	f)	**Describe the concept of SOC in embedded system.**	**4M**
	Ans:	**(**Note: Consider any example of SOC given by students. **)** **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.	**(Description: 4 marks) Description without example can be given full marks.**

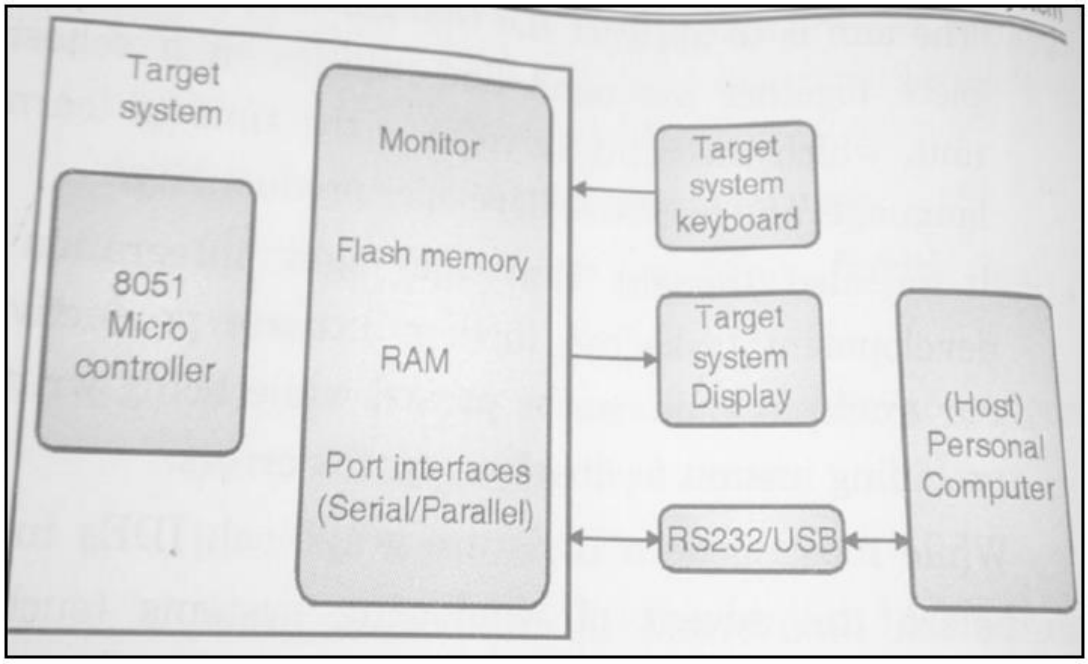
		• Other logic and analog units. **Example of SOC is single-chip mobile phone:**  	
6.		**Attempt any four of the following:**	**16 Marks**
	a)	**Explain the terms:** **i) In circuit emulator** **ii) Target board.**	**4M**
	Ans:	**(i) In circuit Emulator:** • An **in-circuit emulator** (ICE) is a hardware device used to debug the software of an embedded system. • The ICE is temporarily installed between the embedded system and an external terminal or personal computer so that the programmer can observe and alter what takes place in the embedded system.	**(Each Explanation: 2 marks)**



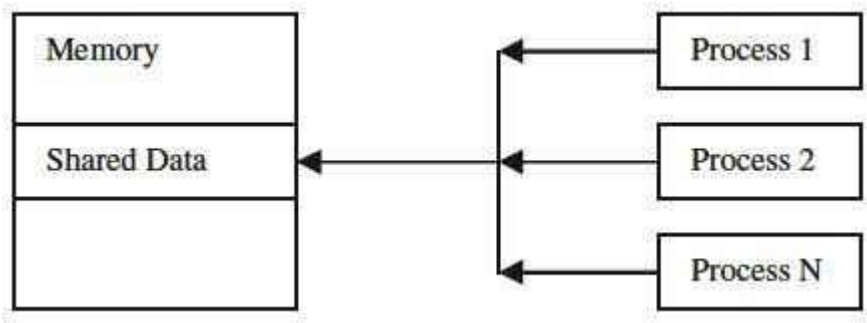
ii) Target board :

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

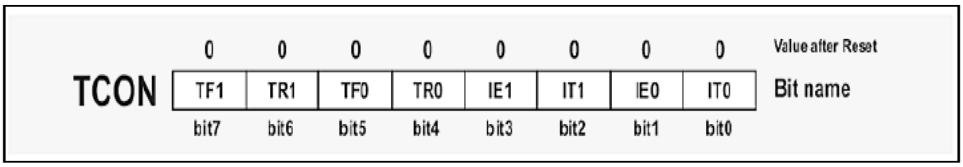


The target board differs from the final system as it interfaces with personal computers as well as work as a standalone system which requires a repeated downloading of the codes during development phase in the flash memory. Also requires repeated

		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.	
	b)	**What is intertask communication? State the various mechanism to achieve it.**	**4M**
	Ans:	**(**Note: Explanation of mechanism is optional**)** **Inter-task communication:** Different tasks in an embedded system typically must share the same hardware and software resources or may rely on each other in order to function correctly. For these reasons, embedded OSs provide different mechanisms that allow for tasks in a multitasking system to intercommunicate and synchronize their behavior so as to coordinate their functions, avoid problems, and allow tasks to run simultaneously in harmony. Embedded OSs with multiple intercommunicating processes commonly implement interprocess communication (IPC) and synchronization algorithms based upon one or some combination of memory sharing, message passing, and signaling mechanisms. With the shared data model shown in Figure, processes communicate via access to shared areas of memory in which variables modified by one process are accessible to all processes.  **Memory sharing.** While accessing shared data as a means to communicate is a simple approach, the major issue of race conditions can arise. A race condition occurs when a process that is accessing shared variables is pre-empted before completing a modification access, thus affecting the integrity of shared variables. To counter this issue, portions of processes that access shared data, called critical sections, can be earmarked for mutual exclusion (or Mutex for short). Mutex mechanisms allow shared memory to be locked up by the process accessing it, giving that process exclusive access to shared data. Embedded OS offers three different mechanisms for inter-task communication: **semaphores, mailboxes, and message passing.**	**(Defining: 2 marks, stating mechanisms: 2 marks)**

	c)	**Draw the format of TCON and TMOD register and label each bit.**	**4M**
	Ans:	**(**Note: Detail explanation of bits is optional**)**	**(Each: 2 marks(1mark for format and 1mark for labeling))**

TCON Format:



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.

TMOD Format:

TMOD: TIMER/COUNTER MODE CONTROL REGISTER. NOT BIT ADDRESSABLE.

GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0
TIMER 1				TIMER 0			

Bits of this register have the following function:
GATE enables and disables Timer 1 by means of a signal brought to the INT1 pin (P3.3):

1 - Timer 1 operates only if the INT1 bit is set.

0 - Timer 1 operates regardless of the logic state of the INT1 bit.

C/$\overline{T1}$ selects pulses to be counted up by the timer/counter 1:

1 - Timer counts pulses brought to the T1 pin (P3.5).

0 - Timer counts pulses from internal oscillator.

M1, M0 These two bits select the operational mode of the Timer 1.

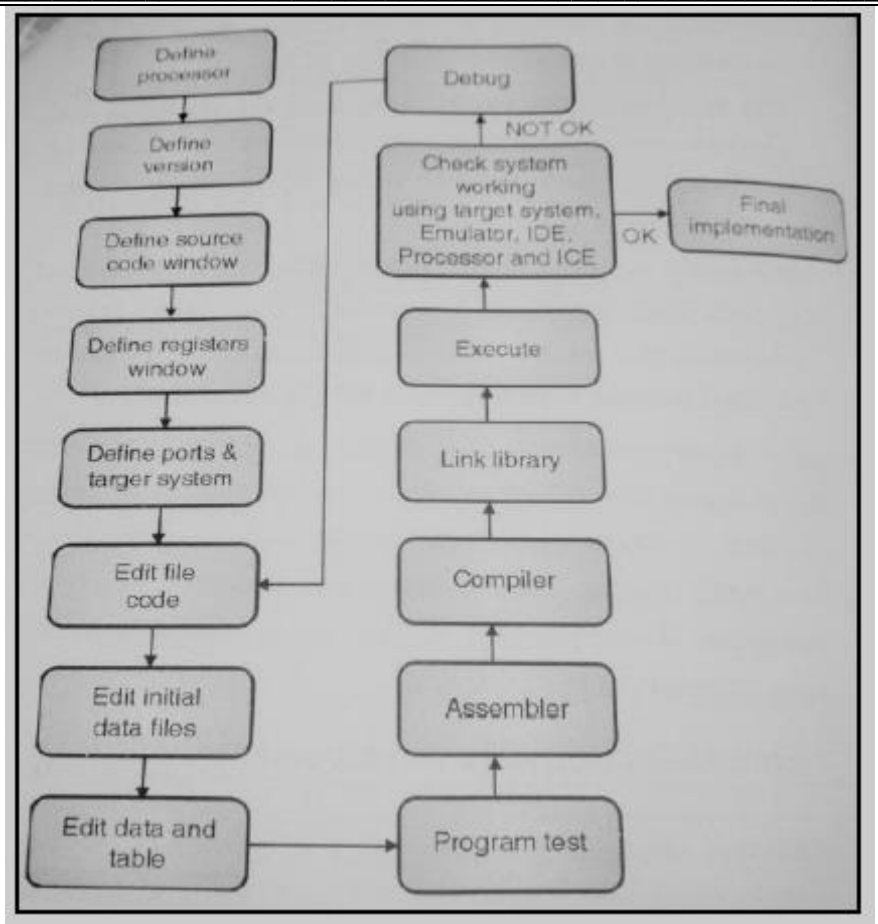
M1	M0	MODE	DESCRIPTION
0	0	0	13-bit timer
0	1	1	16-bit timer
1	0	2	8-bit auto-reload
1	1	3	Split mode

GATE enables and disables Timer 0 using a signal brought to the INT0 pin (P3.2):
1 - Timer 0 operates only if the INT0 bit is set.
0 - Timer 0 operates regardless of the logic state of the INT0 bit.

C/$\overline{T0}$ selects pulses to be counted up by the timer/counter 0:

		1 - Timer counts pulses brought to the T0 pin (P3.4). **0** - Timer counts pulses from internal oscillator. **M1, M0** These two bits select the operational mode of the Timer 0.	
	d)	**What is task synchronization? Explain in brief.**	**4M**
	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). **Task synchronization is achieved using 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. A semaphore is a single variable that can be incremented or decremented between zero and some specified maximum value. The value of the semaphore can communicate state information. A mail box flag is an example of a semaphore. The flag can be raised to indicate a latter is waiting in the mailbox. A semaphore is a means of protecting a resource/data shared between threads. It is a token based mechanism for controlling when a thread can have access to the resource/data. Usually a semaphore handle will be able to be received from the system by name/id. Semaphores are used for two purposes 1) Process Synchronization 2) Critical Section problem / Mutual Exclusion Semaphore is like a key that allows a test to carry out some operation or to access a resource A kernel supports many different types of semaphores **Binary:** Binary semaphores are used for both mutual exclusion and synchronization purposes. A binary semaphore is used to control sharing a single resource between	**(Definition: 2 marks, Explanation: 2 marks)**

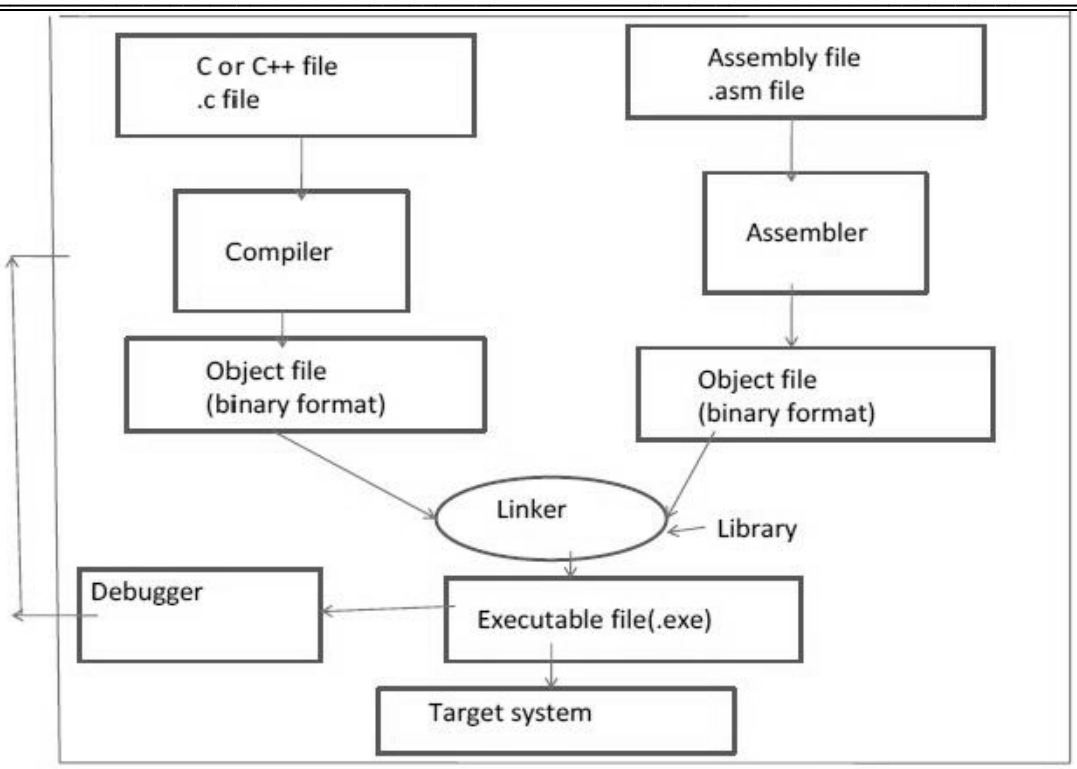
		tasks. Its internal counter can have only the values of 1 (available) and 0 (unavailable). A semaphore test passes if the count is 1, in which case, the current task is allowed to proceed. **Counting:** it is a semaphore that increments when an IPC is given by a task. It decrements when a waiting task unblocks and starts running. **Mutex:** Mutexes are binary semaphores that include a priority inheritance mechanism. Mutexes are the better choice for implementing simple mutual exclusion (hence 'MUT'ual 'EX'clusion). When used for mutual exclusion the mutex acts like a token that is used to guard a resource. When a task wishes to access the resource it must first obtain ('take') the token. When it has finished with the resource it must 'give' the token back - allowing other tasks the opportunity to access the same resource.	
	e)	**Describe steps in embedded software development cycle.**	**4M**
	Ans:	**Steps:** 1) Define the processor /processing device (family and version) for the target system. 2) Defining the source code window with labels and symbolic arguments as execution goes on for each single step. 3) Define the processor registers for each step /module. 4) Define details of ports and target system. 5) Editor to edit source code files, initial data files, data and tables. 6) Define assembler/compiler for program test with link library. 7) Execute the source code to check the target system, else debug the source code. 8) For system working properly as per the specifications, then final implementation is carried out. 9) Finally application software is embedded in the system by using device programmer.	**(Steps listing: 2 marks, Flow chart: 2 marks)**



OR

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



	f)	**Write assembly or C language program to take data from P2 and P3.EX- OR this received data and send result on P1 of microcontroller.**	**4M**

	Ans:	ORG 0000H
		MOV P2 , #0FFH ; P2 as input port
		MOV P3 ,#0FFH ; P3 as input port
		MOV P1 , #00H ; P1 as output port
		MOV A, P2 ; transfer the first value to A
		MOV R1, A ;transfer the value to R1
		MOV A, P3 ; transfer the second value to A
		XRL A, R1 ; X-OR contents
		MOV P1 , A ; transfer the result to P1
		END
		
		OR
			(Correct program : 4 marks)(Assembly or C language)

		include<reg51.h>
		void main(void)
		{
		unsigned char x ,y, z;
		P2 =0x0ffh;
		P3= 0x0ffh;
		P1 = 0x00h;
		x= P2;
		y = P3;
		z = x^y;
		P1 = z;
		}
		
		(NOTE: Program may change. Please check the logic and understanding of students.**)**	