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 tryto 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.

<table>
<tr><th>Q. No.</th><th>Sub Q. N.</th><th>Answer</th><th>Marking Scheme</th></tr>
<tr><td>1.</td><td></td><td><b>Attempt any FIVE of the following:</b></td><td><b>20 Marks</b></td></tr>
<tr><td></td><td>a)</td><td><b>Describe the following pins of 8051 microcontroller.</b>
(i) TXD
(ii) RXD
(iii) $\overline{INT_0}$
(iv) $\overline{INT_1}$</td><td><b>4M</b></td></tr>
<tr><td></td><td><b>Ans:</b></td><td>
<table>
<tr><th>Pin</th><th></th><th>Name</th><th>Bit Address</th><th>Function</th><th>Pin number</th></tr>
<tr><td>P3.0</td><td></td><td>RXD</td><td>B0H</td><td>Receive data for serial port
SBUF has two register one is to read only and hold to receive data from external sources vis RXD</td><td>10</td></tr>
<tr><td>P3.1</td><td></td><td>TXD</td><td>B1H</td><td>Transmit data for serial port
SBUF has two register another is to write only and hold the data to be transmitted out of 8051 via TXD</td><td>11</td></tr>
<tr><td>P3.2</td><td></td><td>INT0-bar</td><td>B2H</td><td>External interrupt 0
INTO is an alternate function P3.2
A signal received at these pins will evoke the interrupts accordingly. But not all signals will evoke the interrupt The signal received at pins should be either a low level one or it should be a falling edge signal to evoke the corresponding interrupt. However to</td><td>12</td></tr>
</table>
</td><td>(1 mark each)</td></tr>
</table>

				serve the interrupt upon receiving the signal at pins,	
P3.3		INT1-bar	B3H	External interrupt 1 INT1 is an alternate function of P3.3. signal received at these pins will evoke the interrupts accordingly. But not all signals will evoke the interrupt! The signal received at pins should be either a low level one or it should be a falling edge signal to evoke the corresponding interrupt. However to serve the interrupt upon receiving the signal at pins,	13

b)
Ans:

Give example for 8051 microcontroller as a Boolean processor.

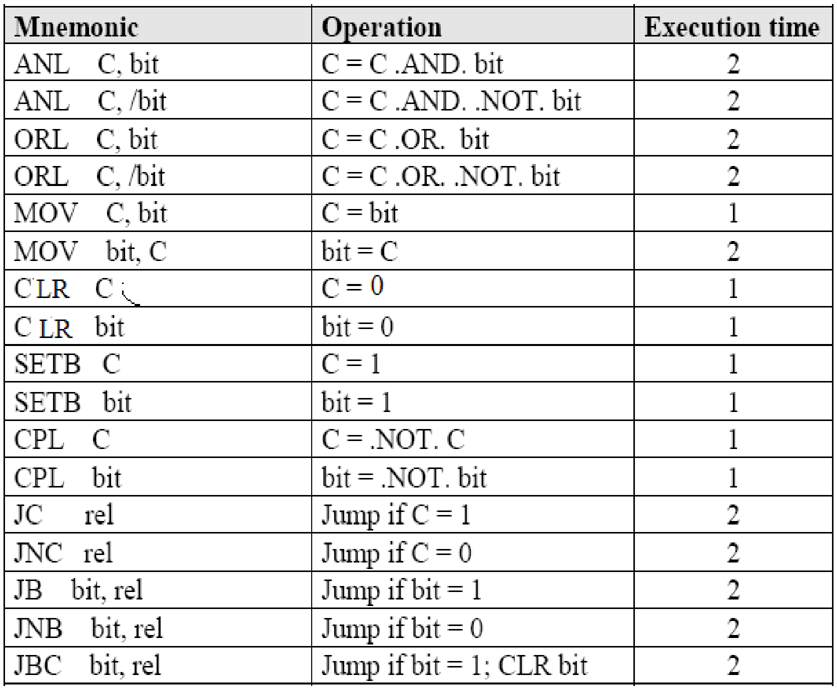
4 M (1 mark each example(any 4 example))

The 8051 includes a number of special features which support the direct manipulation and testing of individual bits and allow the use of single-bit variables in performing logical operations. These features are referred to as “Boolean Processor”.

In digital control application, bit-processing capacities in conjuction with the microcomputer’s byte-processing and numerical capabilities become of immense significance.

The processor is power full to do the numerical calculations on byte but controller deals with bits not byte. We use controller to make ON or OFF the switch, relay, motor.

8051 provides separate processor to handle bits as a data type. The bit processing can be done by byte process but remaining 7 bit of 1 byte wasted and affected to overcome this disadvantage Boolean processor is used. Boolean processor can do direct bit manipulation, testing of individuals bit logical manipulations. processor provides 17 Boolean instructions



Mnemonic	Operation	Execution time
ANL C, bit	C = C .AND. bit	2
ANL C, /bit	C = C .AND. .NOT. bit	2
ORL C, bit	C = C .OR. bit	2
ORL C, /bit	C = C .OR. .NOT. bit	2
MOV C, bit	C = bit	1
MOV bit, C	bit = C	2
CLR C	C = 0	1
CLR bit	bit = 0	1
SETB C	C = 1	1
SETB bit	bit = 1	1
CPL C	C = .NOT. C	1
CPL bit	bit = .NOT. bit	1
JC rel	Jump if C = 1	2
JNC rel	Jump if C = 0	2
JB bit, rel	Jump if bit = 1	2
JNB bit, rel	Jump if bit = 0	2
JBC bit, rel	Jump if bit = 1; CLR bit	2

Table 8 List of Boolean Instructions

c) **List various addressing modes of 8051 microcontroller along with one example of each.**

4 M

Ans:

{**Note : any other correct example can write **}

(Example: 2 marks and List:2 marks (any one example only))

List :

1) Immediate addressing mode
2) Direct addressing mode
3) Register direct addressing mode
4) Register indirect addressing mode
5) Indexed addressing mode

1) Immediate addressing mode

Ex 1 MOV A, #6AH

Ex 2 MOV R1,#55 H

2) Direct addressing mode

Ex1 MOV A, 04H

Ex2 MOV P1 , A

3) Register direct addressing mode

Ex1MOV A, R4

Ex2 ADD A, R7

4)Register Indirect address mode

Ex1 MOV A, @R0

EX2 MOV B,@R1

5) Indexed addressing mode.

Ex1 MOVC A, @A+DPTR

EX2 MOVC A, @A+PC

d) **Write an assembly language C language program of 8051 microcontroller for adding ten numbers in an array. Assume suitable data.** **4M**

Ans: **ALP**

```
          ORG 0000H
    MOV R2,#00H             ; INITIALIZE CARRY
    MOV R3,#0AH             ; INITIALIZE COUNT
   MOV R0,#40H              ;      INITIALIZE MEMEROY ADDRESS
    CLR A
    UP: ADD A,@R0           ;ADD FIRST NO WITH 00H
    JNC DN    ;   IF NO CARRY THEN GO DN
    INC R2
    DN: INC R0   ; INCREMENT MEMORY POINTER
    DJNZ R3,UP          ; DECREMENT COUNT IF COUNT IS NOT 0 THEN
                                      GO TO UP
    MOV 60H,A   ; IF COUNT IS ZERO STORE RESULT
    MOV 61H, R2         ; STORE CARRY
    LOOP: SJMP LOOP
```

(Correct program: 4 marks (any other correct method can be considered))

OR
[C language]

Solution:

```
#include <reg51.h>
void main(void)
{
unsigned char mydata[]={0x2,0x6,0x3,0x5,0X4,0X6,0X7,0X8,0X1,0X10};
unsigned char sum=0;
unsigned char x;
for (x=0;x<=9;x++)
{
P2=mydata[x];
sum=sum+mydata[x];
P1=sum;
}
}
```

e) **With proper format, describe the interrupt Priority (IP) register.** **4M**

Ans:

D7							D0
--	--	PT2	PS	PT1	PX1	PT0	PX0

Priority bit = 1 assigns high priority. Priority bit = 0 assigns low priority.

--	IP.7	Reserved
--	IP.6	Reserved
PT2	IP.5	Timer 2 interrupt priority bit (8052 only)
PS	IP.4	Serial port interrupt priority bit
PT1	IP.3	Timer 1 interrupt priority bit
PX1	IP.2	External interrupt 1 priority bit
PT0	IP.1	Timer 0 interrupt priority bit
PX0	IP.0	External interrupt 0 priority bit

User software should never write 1s to unimplemented bits, since they may be used in future products.

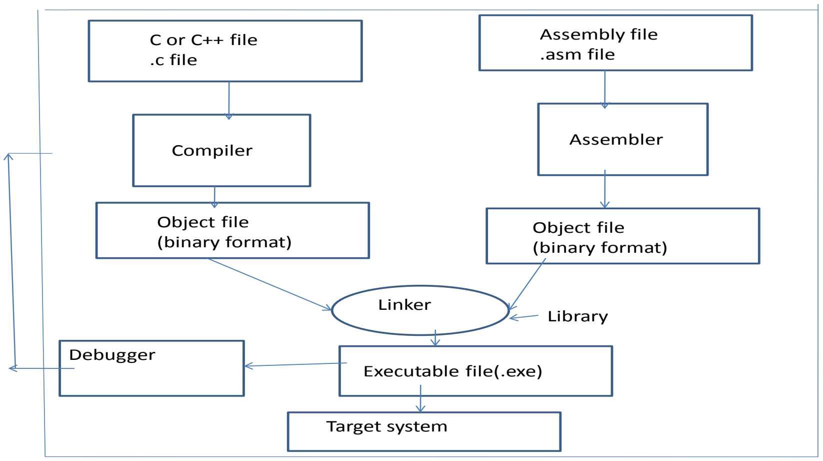
(Format: 2 marks,description:2 marks)

f) **Describe the steps in the embedded software development cycle.**

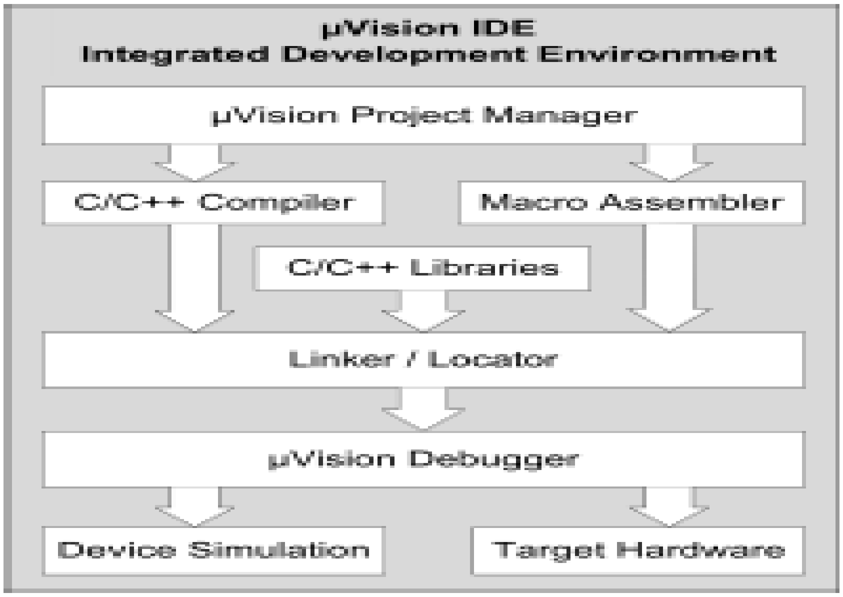
4M

(Diagram: 1 mark and Description: 3 marks)

Ans:



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

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). To support modular code and reusable libraries of code, most assemblers and compilers today come with Linkers and Librarians. Linkers, link code modules saved in different files together into a single final program. At the same time they take care of a chip's memory allocation by assigning each instruction to a microcontroller memory addresses in such a way that different modules do not overlap.

Librarians help you to manage, organize and revision control a library of re-usable code modules. Once the ASCII source code text file has been assembled (with an Assembler) or compiled (with a Compiler) and the files have been linked (with the Linker), the output results in a number of files that can be used for debugging the software and programming the actual microcontroller's memory.

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.

A Debugger allows you to download your code to the emulator's memory and then control all of the functions of the emulator from a PC. Common debugging features include the capability to examine and modify the microcontroller's on-chip registers, data- and program-memory; pausing or stopping program executing at defined program locations by setting breakpoints; single-stepping (execute one instruction at a time) through the code; and looking at a history of executed code (trace). So far we've talked about several different pieces of software: Text Editor, Assembler or Compiler, Linkers, Librarians and Debugger. You can easily imagine that it can become quite a time-consuming challenge to alternate back and forth between all of these programs during the debugging process (discover a bug, edit the source code, compile it again, link it again, download the modified code to the emulator, etc.). This is where an integrated development environment (IDE) comes in.

4. OTP and Flash Programming

It can't be stretched enough: A starter kit or emulator 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.

g) **Describe the concept of Round Robin Scheduling with reference to real time operating system (RTOS).** **4M**

(Diagram: 1 mark and Explanatio : 3 marks)

Ans:

Round robin algorithm

In the round robin algorithm, the kernel allocates a certain amount of time for each task waiting in the queue. The time slice allocated to each task is called quantum. As shown in fig. If three tasks 1, 2, 3 are waiting in the queue the CPU first executes task1 then task2 then task 3 and the again task1 in round robin algorithm each task waiting in the queue is given a fixed time slice. the kernel gives control to the next task if the current task has completed its work within the time slice or if the current task has completed it allocated time

The kernel gives control to the next task if

(a) the current task has completed within the time slice

(b) the current task has no work to do

(c) The current task has completed its allocated time slice this algorithm is very simple to implement but there is no priorities for any task. All tasks are considered of equal importance. If time critical operation are not involved then this algorithm will be sufficient. digital millimeter , microwave oven has this algorithm

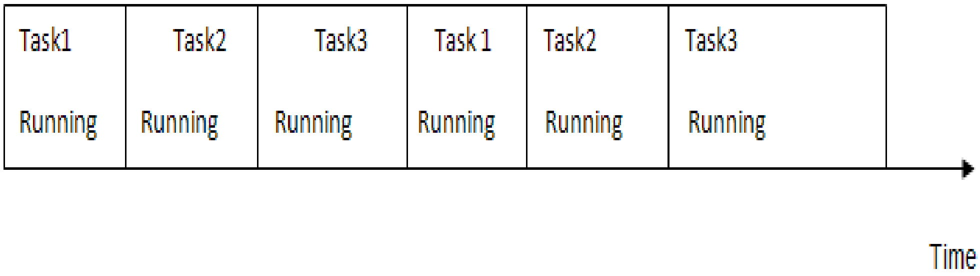


Fig. Round robin scheduling algorithm

2.

Attempt any FOUR of the following:

16 Marks

a) **Write important features of 8051 microcontroller.**

4M (Features of 8051 Microcontroller (Any four Each 1 mark))

Ans:

1. 4 KB on chip program memory (ROM or EPROM)).
2. 128 bytes on chip data memory (RAM).
3. 8-bit data bus
4. 16-bit address bus
5. 32 general purpose registers each of 8 bits
6. Two -16 bit timers T_0 and T_1
7. Five Interrupts (3 internal and 2 external).
8. Four Parallel ports each of 8-bits (PORT0, PORT1,PORT2,PORT3) with a total of 32 I/O lines.
9. One 16-bit program counter and One 16-bit DPTR (data pointer)
10. One 8-bit stack pointer
11. One Microsecond instruction cycle with 12 MHz Crystal.
12. One full duplex serial communication port.

b) **Write the instruction to exchange databytes at address 40H and 50H using assembly language programming or C language programming.**

4M

{**Note: Equivalent 'C' language program full marks should be given**}

(Correct program: 4 marks)

Ans:

```
ORG 0000H
MOV A,40H  ; TAKE NUMBER FROM 40H INTO A
MOV B,50H  ; TAKE NUMBER FROM 50H INTO B
MOV 40H,B   ;STORE THE NUMBER FROM B AT 40H
MOV 50H,A  ;STORE THE NUMBER FROM A AT 50H
SJMP $
END
```

OR

```
ORG 0000H
MOV R0,#40H ; initialize pointer register R0 by 40h
MOV R1,#50H; initialize pointer register R1 by 50h
MOV R7, #05H  ; get count 05h in R7
UP: MOV A,@R0  ;Get    content  of Data memory  whose address is in R0 in A
MOV B,@R1    ; Get    content  of Data memory  whose address is in R1 in B
MOV @R0, B ; exchange data
MOV @R1, A       ;exchange data
INC R0    ; increment memory pointer R0
INC R1             ;increment memory pointer R1
DJNZ R7, UP   ;Decrement count if not 0 then go to up
END
```

c)

Ans:

Write an assembly program or C language to generate a square wave of 1kHz at port pin 1.5 using auto reload mode of Timer O.

4M

(Calculation: 1 mark and Program: 3 marks)

Look at the following steps for 1 KHz frequency calculations with 12 MHz.
The period of the square wave = 1 / 1 KHz
=1 ms.
The high or low portion of the square wave = Time period / 2
=   1ms / 2
= 0.5m Sec.
Timer clock Frequency is = XTAL / 12
= 12 MHz / 12
= 1 MHz
Timer clock period is = 1/ Timer Frequency
= 1 / 1 MHz
= 1 uSec
Counter = Delay / timer clock period
=0.5mSec / 1 uSec
= 50 0
50 x10
Timer Reload value = Maximum Count – Counter
= 256 – 50
= (206)d
Timer Reload value in HEX = (206)d
= (CE   ) h .
TH0 = 0xCE h.

C language program to generate square wave over Port Pin P1.5 using timer 0 3 marks

```
#include <Intel\8052.h>
```

```
#include <standard.h>
Void delay (void); //Timer 0, Mode 2(8 bit timer)
SBIT OUTPUT P1^5; // Initialize Port pin P1.5 as output
Void main ()
{
While (1)
{
OUTPUT= ~ OUTPUT;          // toggle P1.5
delay ();            // delay
}
}
Void delay ()  ;
{
Unsigned char  x;
For(x=0;x<=10;x++);
{
TMOD = 0x02h;              // Timer 0, Mode2(8bit auto reload timer)
TH0 = 0xCEh;             //Load TH0
TR0 = 1;                 //Run the timer 0
while (TF0 = = 0)           // Wait for TF0 to overflow
TR0 = 0;               //Stop the timer 0
TF0 = 0;                 //Clear TF0
}
}
```

//Assembly language program to generate square wave over Port Pin P1.5 using timer 0 3Marks

```
    ORG 0000H
    MOV     TMOD,#02H   ;TIMER0 IN MODE2
UP:   CPL  P1.5
    MOV     R3,#10
    MOV     TH0,# 0CEH   ;load count Th0
RPT: SETB     TR0       ;START TIMER0
BACK : JNB   TF0 , BACK    ;wait till TF=1
    CLR  TF0
    DJNZ     R3 , RPT
    SJMP     UP
    END
```

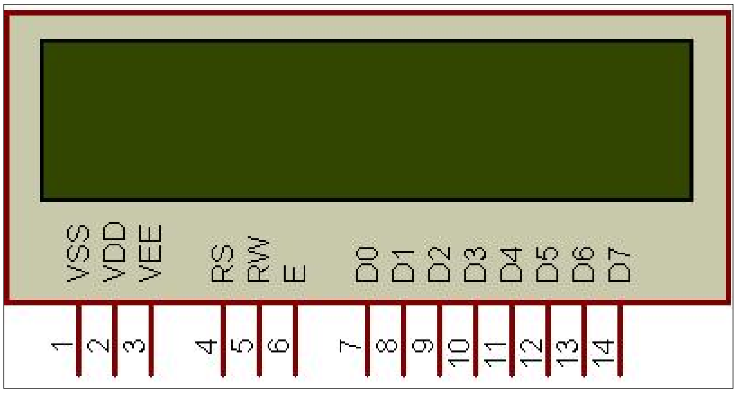
d) **Draw the pin out of 14 pin LCD display and state the function of following:** **4M**

- **(i) RS**
- **(ii) R/W**
- **(iii) EN**

Ans:

(Pin out diagram: 1 mark and Explanations of each pin: 1 mark)

Pin out diagram of 14 pin LCD



Explanation of RS, R/W and EN

RS: - RS is used to make the selection between data and command register.

RS=0, command register is selected

RS=1 data register is selected.

RW: -R/W gives you the choice between writing and reading.

R/W=1, reading is enabled.
R/W=0 then writing is enabled.

EN: -Enable pins is used by the LCD to latch information presented to its data pins.

When data is supplied to data pins, a high to low pulse must be applied to this pin in-order for the LCD to latch in the data present at the data pins.

e) **State the function of the following :** **4M**

(i) Compiler

(ii) Debugger

(iii) Simulator

(iv) Emulator

Ans:

(Each 1 marks)

1) Compiler: -It is a computer program that transforms the source code written in a programming or source language into another computer language i.e. target language i.e. binary code known as object code.

2) Debugger: - Allows you to download your code to the emulator's memory and then control all of the functions of the emulator from a PC. Common debugging features include the capability to examine and modify the microcontroller's on-chip registers, data- and program-memory; pausing or stopping program executing at defined program locations by setting breakpoints; single-stepping (execute one instruction at a time) through the code; and looking at a history of executed code (trace).

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

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

4) Emulator: - It duplicates the functions of one system using a different system, so that the second system behaves like the first system. A hardware emulator is an emulator which takes the form of hard ware device.

- Emulators maintain the original look, feel, and behavior of the digital object as digital data itself.
- Reduces labor hours.
- It allows video games exclusive to one system to be played on another

f) **Differentiate between Desktop operating system and Real time operating system based on following parameter:** **4M**

(i) Time Behaviour
(ii) Application
(iii) Kernal
(iv) Delays
(v) Example

Ans:

(Correct answer full marks))

Sr. No.	**OS**	**RTOS**
Time behavior	It is used for systems/applications that are not time critical.	**It is** used for time critical systems.
Applications	OSes are used in a wide variety of applications Used for general universal applications	RTOSes are generally embedded in devices that require real time response **Used for dedicated electronics applications**
Delay	OSes use a time sharing design to allow for multi-tasking	RTOSes either use a time sharing design or an event driven design

Kernal	A normal OS does not have preemption at the kernel level	RTOS has a preemptive kernel
Example	Ex: Windows, Linux, Unix	EX : Vx Works, uCos

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

a) **Compare data memory and program memory with respect to 8051 microcontroller on following parameter** **4M**

(i) Usage
(ii) Signals for interfacing
(iii) On chip size
(iv) Extendable memory
(v) Pointers used

Ans:

Sr.No.		**Data Memory**	**Program Memory**
i	**Usages**	**To store temporary Data**	**To Store Program code**
ii	**Signals for Interfacing**	**/RD and /WR**	**/PSEN**
iii	**On Chip Size**	**128 Byte**	**4 K.Byte**
iv	**Extended Memory**	**Upto 64KB**	**Upto 64KB**
v	**Pointers Used**	**DPTR**	**PC**

(Any four point : 4 marks)

b) **Give complete classification of JUMP instruction.** **4M**

Ans:

1. Unconditional Jump

```
1.1   LJMP      add16        ;LONG  JUMP
1.2   AJMP      add11        ;ABSOLUTE  JUMP
1.3   SJMP    relative add   ;SHORT  JUMP
```

2. Conditional Jump.

```
2.1 CJNE    ;Compare and Jump if Not Equal
2.2 DJNZ    ; Decrement Register and Jump if Not Zero
2.3 JB      ; Jump if Bit Set
2.4 JNB     ; Jump if Bit Not Set
2.5 JBC     ; Jump if Bit Set and Clear Bit
2.6 JC      ; Jump if Carry Set
2.7 JNC     ; Jump if Carry Not Set
2.8 JNZ     ; Jump if Accumulator Not Zero
2.9 JZ      ; Jump if Accumulator Zero
```

(Unconditional JUMP: 2 marks and Conditional JUMP: 2 marks)

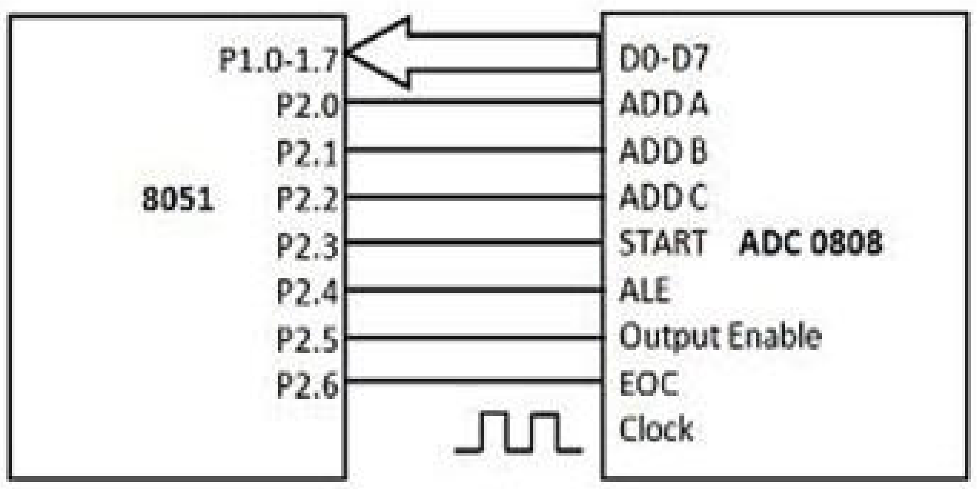
c) **How will you implement single step operation in IC 8051.** **4M**

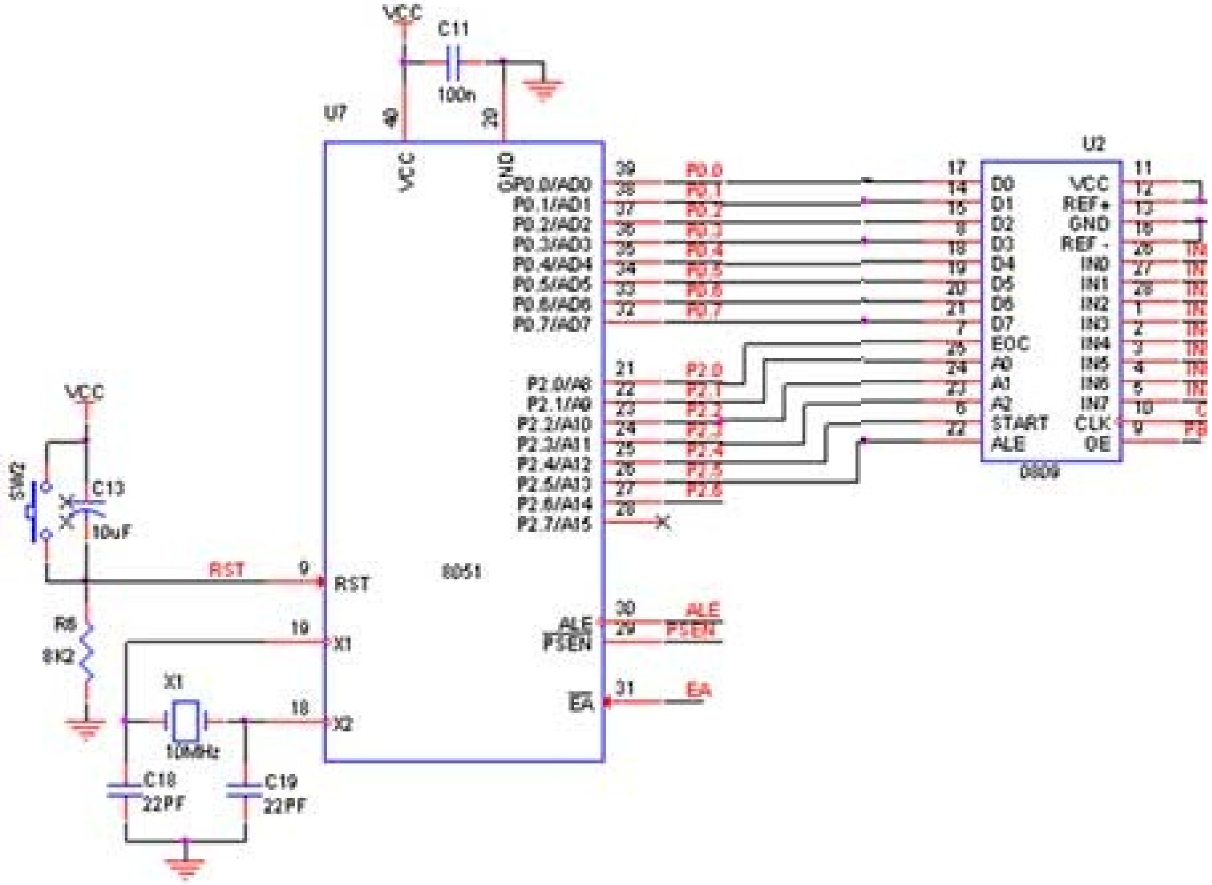
Ans:

Technique: Single step operation can be implemented by using interrupt structures available in 8051 microcontroller.

Explanation: In ISR, once an interrupt routine has been entered, it con not be re-entered until at least on instruction of the interrupted program is executed. This feature is used for single

(Technique: 2 marks and Explanation: 2 marks)

		step operation. In this one of the external interrupt is to be level triggered activated. The service routine for the interrupt will terminate with following code: JNBP3.2, $; Wait here till INT0 goes High. JB P3.2, $; Now Wait here till INT0 goes Low. RETI ; Go back and execute one instruction. Now if INT0 pin, which is also the P3.2 pin, is held normally low, the CPU will go right into the External Interrupt 0 routine and stay there until INT0 is pulsed (from low to high and high to low). Then it will execute RETI, go back to the task program, execute one instruction, and immediately re enter the External Interrupt 0 routine to await the next pulsing of P3.2.One step of the task is executed each time P3.2 is pulsed.	
	d) **Ans:**	**Draw interfacing of ADC 0808 with microcontroller 8051.**  8051 P1.0-1.7 P2.0 P2.1 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 **OR**	**4M** **(Neat ADC Interfacing diagram: 4 marks)**

		 VCC C11 100n U7 40 20 VCC GND P0.0/AD0 P0.1/AD1 P0.2/AD2 P0.3/AD3 P0.4/AD4 P0.5/AD5 P0.6/AD6 P0.7/AD7 39 38 37 36 35 34 33 32 P0.0 P0.1 P0.2 P0.3 P0.4 P0.5 P0.6 P0.7 P2.0/A8 P2.1/A9 P2.2/A10 P2.3/A11 P2.4/A12 P2.5/A13 P2.6/A14 P2.7/A15 21 22 23 24 25 26 27 28 P2.0 P2.1 P2.2 P2.3 P2.4 P2.5 P2.6 U2 17 14 15 8 18 19 20 21 7 25 24 23 6 22 D0 D1 D2 D3 D4 D5 D6 D7 EOC A0 A1 A2 START ALE VCC REF+ GND REF - IN0 IN1 IN2 IN3 IN4 IN5 IN6 IN7 CLK OE 11 12 13 16 26 27 28 1 2 3 4 5 10 9 0809 VCC SW2 C13 10uF RST 9 RST 8051 R8 8K2 19 X1 X1 18 X2 10MHz C18 22PF C19 22PF ALE PSEN 30 29 ALE PSEN EA 31 EA	
e) **Ans:**		**Describe the cross assemblers and cross complier in brief.** **Cross assembler:-**It is an assemble capable of creating machine code for a platform other tan the one on which it is running. It is useful to convert object codes for microcontrollers or processor to other codes for another microcontrollers or processor and vice versa. **Cross compiler:-**It is an compiler assemble capable of creating machine code for a platform other tan the one on which it is running. It is used to create executable code other than one on which the compiler is run. They are used to generate executables for embedded systems or multiple platforms.	**4M** **(Cross assembler: 2 marks and Cross compiler: 2 marks)**
f) **Ans:**		**State the method of task synchronization and describe any one in detail.** **The methods of task synchronization are:** ➢ Semaphore ➢ Message queue. ➢ Mutual exclusion. ➢ Dead lock.	**4M** **(Task synchronization Methods: 2 marks and Any**

one description: 2 marks)

- Mailboxes.
- Message Queues.

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

Example of using semaphores for Synchronization:

Assume two concurrent process P1 and P2 having statements S1 and S2. We want in any case S1 should execute first. this can be achieved easily by initialize Sem=0;

```
In process P1
{
// Execute whatever you want to do
// before executing P2
S1;
signal(Sem);
}
in process P2
{
wait(Sem);
S2;
}
```

OR

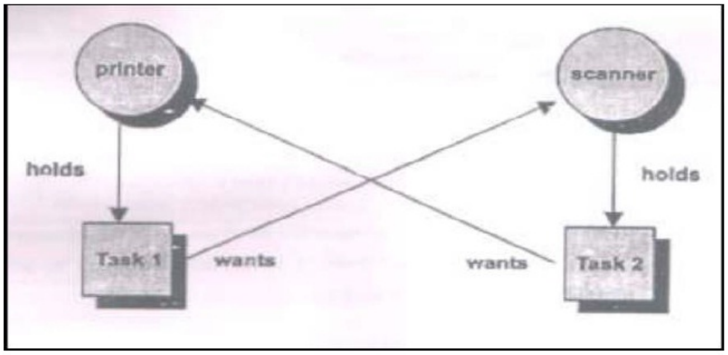
Deadlock:

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.



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

OR

- **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

OR

[Any other method of task synchronization]

4. **Attempt any FOUR of the following :** **16 Marks**

a) **Describe the PSEN, EA, ALE AND RST of 8051 IC.** **4M**

Ans:

(Each correct function:1 mark each)

Function of PSEN:

1. PSEN stands for — program store enable. The read strobe for external Program Memory is the signal PSEN (Program Store Enable). In an 8031-based system in which an external ROM holds the program code, this pin is connected to the OE pin of the ROM.

   In other words, to access external ROM containing program code, the 8031/51 uses the PSEN signal. This read strobe is used for all external program fetches. PSEN is not activated for internal fetches.

Function of EA:

1. EA which stands for external access is pin number 31 in the DIP packages. It is an input pin and must be connected to either Vcc or GND. In other words, it cannot be left unconnected.
2. The lowest 4K (or SK or 16K) bytes of Program Memory can be either in the on-chip ROM or in an external ROM. This selection is made by strapping the EA (External Access) pin to either VCC or Vss.
3. In the 4K byte ROM devices, if the pin is strapped to Vcc, then program fetches to addresses 0000H through OFFFH are directed to the internal ROM. Program fetches to addresses 1000H through FFFFH are directed to external ROM.
4. If the pin is strapped to Vss, then all program fetches are directed to external ROM. The ROM less parts must have this pin externally strapped to VSS to enable them to execute properly.

Function of ALE:

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

Function of RESET:

1. Pin 9 is the RESET pin. It is an input and is active high (normally low). Upon applying a high pulse to this pin, the microcontroller will reset and terminate all activities.
2. This is often referred to as a power-on reset. Activating a power-on reset will cause all values in the registers to be lost. It will set program counter to all 0s.

In order for the RESET input to be effective, it must have a minimum duration of two machine cycles. In other words, the high pulse must be high for a minimum of two machine cycles before it is allowed to go low

b) **Explain the following 8051 instructions:**
(i) SETB C
(ii) ADD A; @ RO
(iii) CJNE A, direct adder, label
(iv) XCHDA, @ R1

4M

Ans:

i.SETB C:
Set the CY bit of PSW register. This is Boolean instruction. After execution of this instruction CY bit will become 1.

ii.ADD A,@R0:
Add the content of accumulator with the content of internal RAM location indirectly given by the specified register R0, and store the result in accumulator.
If R0 is 40H, then after execution of ADD A,@R0 – then content of accumulator and content of internal RAM 40H are added, nd result is stored in accumlator

iii.CJNE A, direct address, lable
Compare the contents of the accumulator with the 8 bit data directly mentioned in the instruction and if they are not equal then jump to the lable mentioned in the instruction.
Example: CJNE A, 40H, UP
Compare the contents of the accumulator with the 40H mentioned in the instruction and if they are not equal then jump to the line of instruction where UP label is mentioned.

iv.XCHD A, @R1
Exchange the low order nibble of the accumulator with the internal ram location indirectly addressed by the specified register R1. The higher order nibbles of each register are not affected.
Example: XCHD A, @R1
Exchange the low order nibble of the accumulator with the lower order nibble of the content of memory location pointed by the R1 register.

(Each correct instruction explanation:1 mark each)

c) **Describe the operating modes of serial port of 8051 microcontroller.**

4M

Ans:

SM0	**SM1**	**MODE**	**operation**	**transmit rate**
0	0	0	Shift register	fixed (xtal/12)
0	1	1	8 bit UART	variable (timer1)
1	0	2	9 bit UART	fixed (xtal/32 or xtal/64)
1	1	3	9 bit UART	variable (timer1)

(Each operating mode:1 mark each)

1. Serial Data Mode-0 – Shift Register (Baud Rate Fixed)

In this mode, the serial port works like a shift register and the data transmission works synchronously with a clock frequency of f_{osc} /12. Serial data is received and transmitted through RXD. 8 bits are transmitted/ received aty a time. Pin TXD outputs the shift clock pulses of frequency f_{osc} /12, which is connected to the external circuitry for synchronization. The shift frequency or baud rate is always 1/12 of the oscillator frequency.

2. Serial Data Mode-1 (standard 8 bit UART mode) (baud rate is variable)

In mode-1, the serial port functions as a standard Universal Asynchronous Receiver Transmitter (UART) mode. 10 bits are transmitted through TXD or received through RXD. The 10 bits consist of one start bit (which is usually '0'), 8 data bits (LSB is sent first/received first), and a stop bit (which is usually '1'). Once received, the stop bit goes into RB8 in the special function register SCON. The **baud rate is variable.**

3. Serial Data Mode-2 Multiprocessor (baud rate is fixed)

In this mode 2 bits are transmitted through TXD or received through RXD. The various bits are as follows: a start bit (usually '0'), 8 data bits (LSB first), a programmable 9 th (TB8 or RB8)bit and a stop bit (usually '1'). While transmitting, the 9th data bit (TB8 in SCON) can be assigned the value '0' or '1'. For example, if the information of parity is to be transmitted, the parity bit (P) in PSW could be moved into TB8. On reception of the data, the 9th bit goes into RB8 in 'SCON', while the stop bit is ignored. The baud rate is programmable to either 1/32 or 1/64 of the oscillator frequency $\mathbf{f_{baud} = (2^{SMOD} /64) f_{osc}}$

Data Mode-3 - Multi processor mode(Variable baud rate)

In this mode 3 bits are transmitted through TXD or received through RXD. The various bits are: a start bit (usually '0'), 8 data bits (LSB first), a programmable 9 th bit and a stop bit (usually '1'). Mode-3 is same as mode-2, except the fact that the baud rate in mode-3 is variable (i.e., just as in mode-1).

$\mathbf{f_{baud} = (2^{SMOD} /32) * (f_{osc} / 12 (256\text{-}TH1))}$

d) **Write an assembly language program to generate a saw tooth wave from when DAC is interfaced with 8051 microcontroller.** **4M**

{**Note: Student can write any one program**}

Ans: **Program to generate Positive saw tooth wave using DAC** **(Correct Program: 4 marks)**

Lable	**Instruction**	**Comments**
	ORG 0000H	;Start the program from 0000h location
	MOV A,#00H	; Take lower value i.e.00 for positive slop
UP:	MOV P1,A	; transfer ACC content to P1 i.e. DAC
	INC A	; Increment ACC

```
             SJMP  UP            ; jump up for continuous waveform
```

OR

Program to generate Negative saw tooth wave using DAC

```
Lable        Instruction         Comments
             ORG  0000H          ;Start the program from 0000h location
             MOV   A,#0FFH       ; Take maximum value i.e. FFH for negative slop
UP2:         MOV   P1,A          ; transfer ACC content to P1 i.e. DAC
             DEC    A             ; Decrement ACC
             SJMP  UP            ; jump up for continuous waveform
```

e) Ans: **State the function of Locator and Loader.**

4M

(Function of Locator: 2 marks and Function of Loader: 2 marks)

Locator:-It is used for relocation process . It is done during compilation also it can be done at run time by a relocating loader. It is a program that takes one or more objects generated by compiler and combines them into a single executable program also generate .abs file. The locator uses this information to assign physical memory addresses to each of the code and data sections within the re-locatable program.

Loader:-Loader is a program that loads machine codes of a program into the system memory. In Computing, a loader is the part of an Operating System that is responsible for loading programs. It is one of the essential stages in the process of starting a program. Because it places programs into memory and prepares them for execution.

Loading a program involves reading the contents of executable file into memory. Once loading is complete, the operating system starts the program by passing control to the loaded program code. All operating systems that support program loading have loaders.

f) Ans: **Describe the concept of mutual exclusion.**

4M

(Concept of mutual exclusion: 4 marks)

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

5. **Attempt any FOUR of the following:**

16 Marks

a) **Describe the function of XTAL1, XTAL2, T0 and T1 pins of 8051 microcontroller.**

4M

Ans: XTAL 1 AND XTAL2: PIN 19 AND PIN18

The XTAL 1 and XTAL2 pins are provided for external quartz crystal connection, in order to generate the required clock for the microcontroller. The maximum frequency of quartz crystal that can be connected to 8051 microcontroller is 12 MHz.

T0(Port3.4) and T1(Port3.5): External input to timer 0 and timer 1 respectively. The internal timer register count is incremented by one, when these pins make negative transition (when signal changes from high to low).

(Function of each pin: 1 mark (1*4=4 marks))

b) **Write a program to find 2's complement of databyte stored in location 20H.**

4M

Ans:

```
ORG 0000H
MOV A,20H
CPL A
ADD A,#01H
END
```

(Program with correct logic: 4 marks)

c) **Give data to enable serial interrupt, timer O and external hardware interrupt 1.**
{Note: Drawing the format of IE not expected & not compulsory**)**

4M

Ans:

EA	**-**	**ET2**	**ES**	**ET1**	**EX1**	**ET0**	**EX0**
1	**0**	**0**	**1**	**0**	**1**	**1**	**0**

IE = 96H

(Correct data byte (binary/ Hexadecimal): 4 marks)

d) **Write a program in C or assembly language for generating fine wave with interface of DAC along with IC8051 microcontroller.**

4M

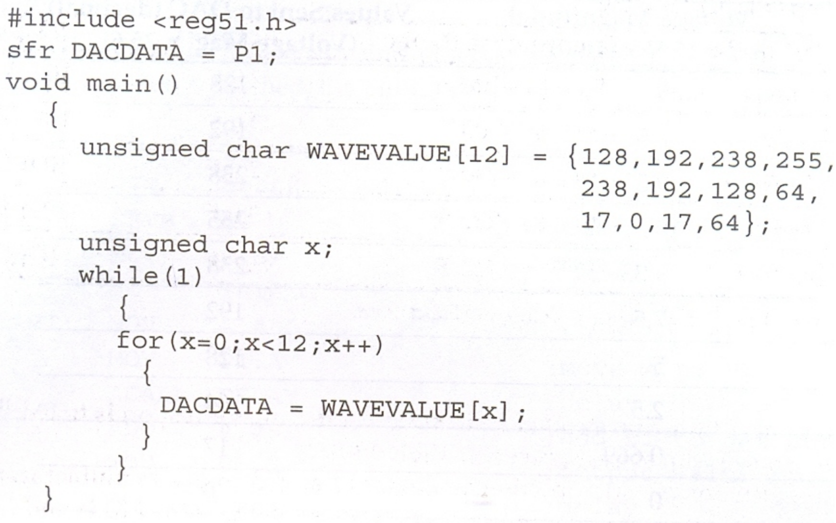
Ans:

Assembly Program

```
AGAIN:  MOV    DPTR , #TABLE
        MOV    R2 , #COUNT
BACK:   CLR  A
        MOVC  A, @A+DPTR
        MOV    P1 ,A
        INC  DPTR
        DJNZ   R2, BACK
        SJMP   AGAIN
        ORG    300
TABLE:  DB   128,192,238,255,238,192 ;See Table    13-7
        DB   128,64,17,0,17,64,128
```

Program using C language

(Program with correct logic: 4 marks)



```
#include <reg51.h>
sfr DACDATA = P1;
void main()
  {
    unsigned char WAVEVALUE[12] = {128,192,238,255,
                                   238,192,128,64,
                                   17,0,17,64};
    unsigned char x;
    while(1)
      {
      for(x=0;x<12;x++)
        {
         DACDATA = WAVEVALUE[x];
        }
      }
  }
```

e)
Ans:

Describe the four advantages of an embedded system.

4M

(1 mark each (1*4=4 marks))

1. Design and Efficiency: The central processing core in embedded system is generally less complicated, making it easier to design. The limited function required of embedded system allows them to design to most efficiently perform their function.
2. Cost: The streamline make-up of most embedded system allows their parts to be smaller less expensive to produce.
3. Accessibility: If something goes wrong with certain embedded systems they can be too inaccessible to repair. This problem is addressed in the design stage, so by programming an embedded system. So that it will not affect related system negatively when malfunctioning.
4. Maintenance: Embedded systems are easier to maintain because the supplied power is embedded in the system and does not required remote maintenance.
5. Redundancies: Embedded system does not involve the redundant programming

f)
Ans:

State the concept of Simaphores in real time operating system (RTOS).

4M

(Concept of Semaphore: 4 marks)

Semaphores: It is a system of sending message by using flags. Multiple concurrent threads of execution with an application must be able to synchronize their execution & co-ordinate mutually exclusive access to shared resources. To fulfill these requirement RTOS kernel provide a semaphore object that one or more threads of execution can acquire or release for the purpose of synchronization or 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 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.

6. **Attempt any FOUR of the following:** **16 Marks**

a) **State the function of Program Counter (PC) and Data Pointer (DPTR) in 8051 microcontroller.** **4M**

Ans: DPTR is a 16-bit register, which can be used as two individual registers:DPL/DPH (Data Pointer Low/High, Addresses 82h/83h).The SFRs DPL and DPH work together to represent a 16-bit value called the *Data Pointer*. The data pointer is used in operations regarding external RAM and some instructions involving code memory. Otherwise it can be used as general purpose registers.

(Function of PC: 2 marks and Function of DPTRP: 2 marks)

Program Counter (PC) is 16-bit register which holds the address of the next instruction to be executed. It is the only register which does not have internal memory address. On reset its value is 0000H

b) **Write an assembly language program for the 8051 microcontroller to rotate two 8 bit numbers stored at memory location 20H and 21H. Sore the product at 22H and 23H.** **4M**

{**Note:The program given below is for multiplication of two numbers. However any program with rotation instruction should be considered and given full marks. Comments not compulsory**)

Ans:

```
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
MOV 22H, A; store LSB at 22h location
MOV 23H, B; store MSB at 23 h location
END
```

((ALP) Program with correct logic: 4 marks)

c) **Describe the difference between the timer and counter operation of 8051 microcontroller.** **4M**

Ans:

Timer	**Counter**
In timer mode, internal clock pulses are counted.	In counter mode, external pulses applied at p3.4 (timer 0) & p3.5 (timer 1) are counted.
Timer is selected by resetting the $\overline{C}$/T bits in TMOD.	Counter is selected by setting the $\overline{C}$/T bits in TMOD.
Timer registers are incremented (by 1), at every machine cycle.	When external i/p goes high to low (negative transition), counter register is incremented (by 1).
Input frequency for the timer is fixed & that is crystal frequency/12.	Max. i/p frequency that can be counted is crystal frequency/24.

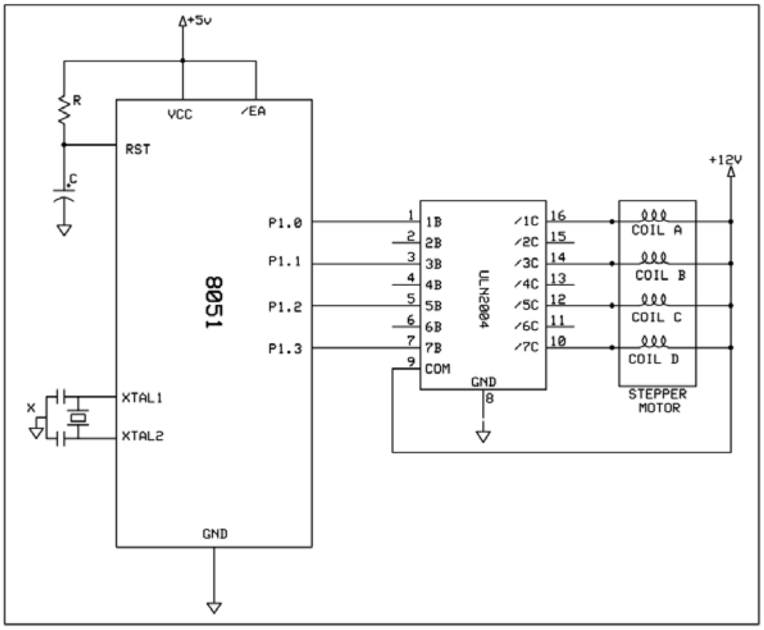
(Difference: 1 mark for each point (1*4=4 marks))

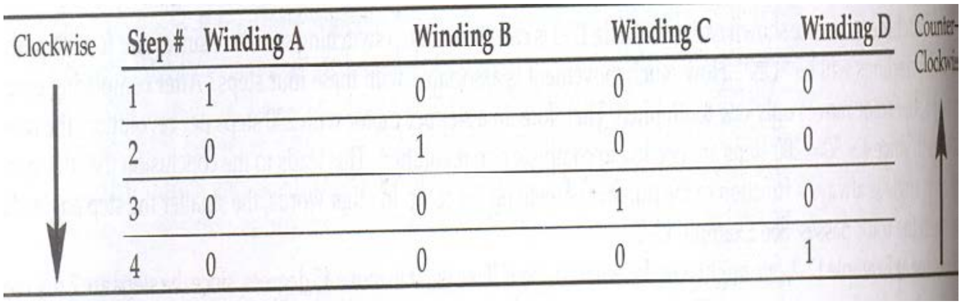
d) **Draw the labeled schematic of stepper motor interface to 8051 microcontrollers port and give bit patterns for full stepping.**

4M

(Correct Interfacing digram: 2 marks and Bit pattern for full stepping: 2marks)

Ans:





Clockwise	Step #	Winding A	Winding B	Winding C	Winding D	Counter-Clockwi
↓	1	1	0	0	0	↑
	2	0	1	0	0	
	3	0	0	1	0	
	4	0	0	0	1	

Sequence (Clockwise): 08H
04H
02H
01H

e) **State different hardware units of embedded system.**

4M

(Any four hardware units: 1 mark for each (1*4=4 marks))

Ans:

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

2. Power supply, reset & oscillator circuit:□□ Most of the systems have their own power supply. Some embedded system do not have their own power supply. These embedded systems are powered by external power supply e.g. Subbase embedded system, network interface card, Graphics Accelerator etc. are powered by Power supply.

		□□ Reset means that processor begins processing of instructions from starting address set by default in program counter on power up. □□ The clock circuit controls execution time of instructions , CPU machine cycles. **3.Timers :**Timer circuit is suitably configured as system clock or RTC (Real time clock). To schedule various tasks and for real time programming an RTC (Real Time Clock), or system clock is needed. **4. Program & data memory:**In embedded system, secondary memory like disk is avoided. Most of the embedded processors have internal memory such as ROM, RAM, flash/EEPROM, EPROM/PROM for storing program and data. **5. Interrupt controller:** It is an interrupt handling mechanism which must exist in embedded system to handle interrupts from various processes and for handling multiple interrupts simultaneously pending for service. **6. I/O ports :**I/O ports are used to interface external devices like sensors, key buttons, transducers, LEDs, LCD actuators, alarms, motors, values, printer etc. There are two types of ports ,parallel and serial port. The parallel ports are used in short distance communication while serial ports are used in long-distance communication. **7.Input& output device interfacing/driver circuits:** Some I/O devices like motors, actuators, valves, sensors are not compatible with the processor. Hence the I/O interface circuits are designed to drive such input and output devices interfaced to the embedded processor **8.System Application specific circuits:** These are the circuits that can control specific target circuits. They consist of ADC,DAC, relays, sensors etc.	
	f) **Ans:**	**State any four applications of Real Time Operating System (RTOS).** ***{** Note: Students may write any four applications from any area. Full marks can be given to students**}*** Embedded systems are used in different applications like automobiles, telecommunications, smart cards, missiles, satellites, computer networking and digital consumer electronics. **1) 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 **2) Embedded Systems in Smart Cards, Missiles and Satellites** Security systems Telephone and banking Defense and aerospace Communication **3) Embedded Systems in Peripherals & Computer Networking** Displays and Monitors Networking Systems Image Processing	**4M** **(Any four applications of RTOS: 1 mark for each (1*4=4 marks))**

		Network cards and printers **4) Embedded Systems in Consumer Electronics** Digital Cameras Set top Boxes High Definition TVs DVDs	