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

SUMMER-16 EXAMINATION
Model Answer

Subject Code: 17626 **Subject Name: Embedded System**

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 judgment 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.

Marks

1. a) Attempt any THREE of the following: **12**

(i) State the alternative function of port 3 of 8051 microcontroller.
(Port pin - 1 mark; functions - 3 marks)

Ans:

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

Or

PORT 3 ALTERNATE FUNCTIONS :

P3 BIT	FUNCTION	PIN
P3.0	RXD	10
P3.1	TXD	11
P3.2	$\overline{INT0}$	12
P3.3	$\overline{INT1}$	13
P3.4	TO	14
P3.5	TI	15
P3.6	$\overline{WR}$	16
P3.7	$\overline{RD}$	17

(ii) Explain the assembler direction DB, END, EQU and ORG of 8051 microcontroller.
(Each 1 mark; examples not compulsory)

Ans:

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. Regardless of which is used, the assembler will convert the numbers into hex:

```
        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
```

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.

Ex ORG 0000h

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.

Ex: 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
```

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.

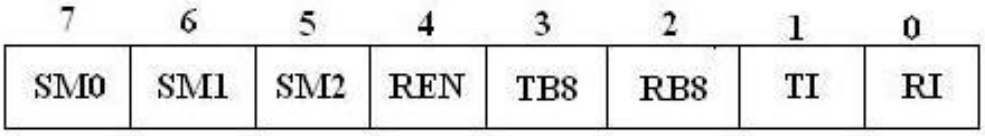
EX:

```
Org 2000h
Mov A,#24h
Mov R3,A
END
```

(iii) Draw the format of SCON register and explain the function of each bit.
(Format - 2 marks; function - 2 marks)

Ans:

SCON Register format:



Function:

SM0, SM1 bits in SCON are used to define the type of the serial communication, baud rate and framing.

SM0	SM1	Mode	Function
0	0	0	Shift register, baud rate = f/12, **Synchronous** serial communication mode
0	1	1	8-bit UART; baud = variable , asynchronous serial communication mode
1	0	2	9-bit UART; baud = f/32 or f/64, asynchronous serial communication mode
1	1	3	9-bit UART; baud = variable, asynchronous serial communication mode

SM2 - bit is used for multiprocessor communication. Set or cleared by the program to enable multiprocessor communications in mode 2 and 3. When set to 1 an interrupt is generated if bit 9 of the received data is a 1; no interrupt is generated if bit 9 is 0;
REN – Receiver enable bit. To accept reception of data this bit must be 1;
TB8 – Transmitted bit 8.
RB8 – Received bit 8.
TI – Transmit interrupt flag. This will be enabled when all bits in the transmitted buffer is shifted out.
RI – Receive interrupt flag. This will be enabled when a character is received in the receiver buffer.

(iv) State any four application of embedded system.
(Each 1 mark)

Ans:

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
- Network cards and printers

4. Embedded Systems in Consumer Electronics

- Digital Cameras
- Set top Boxes
- High Definition TVs
- DVDs

(v) List the interrupts used in 8051 microcontroller. Give their priorities and vector addresses.
(List - 2 marks; priorities -1 mark; vector address - 1mark)

Ans:

List of interrupts:

1. External hardware interrupt0($\overline{INT0}$)
2. Timer0 interrupt(TF0)
3. External hardware interrupt 1($\overline{INT1}$)
4. Timer1 interrupt(TF1)
5. Serial COM interrupts (RI and TI)

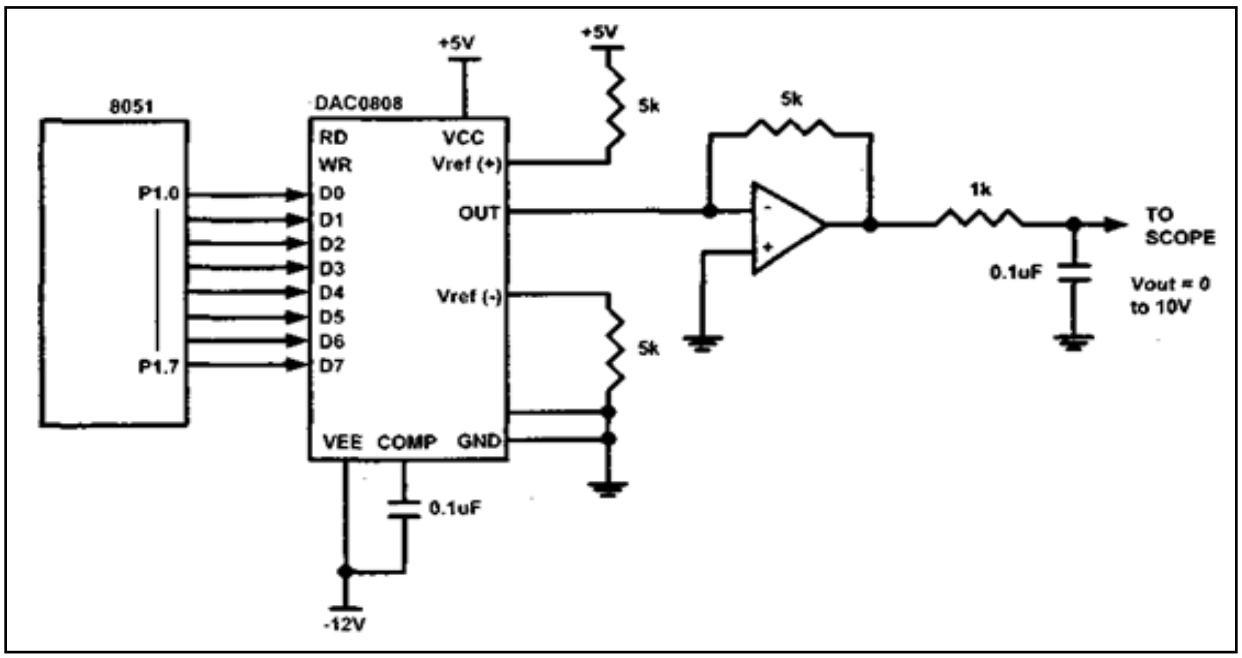
Interrupt Source	Vector address	Interrupt priority
External Interrupt 0 –INT0	0003H	1
Timer 0 Interrupt	000BH	2
External Interrupt 1 –INT1	0013H	3
Timer 1 Interrupt	001BH	4
Serial Interrupt	0023H	5

Marks

b) Attempt any ONE of the following: 6

(i) ***Draw the interfacing diagram of DAC with 8051*** **microcontroller. Write a program to generate triangular wave.**
(Diagram - 3 marks; program - 3 marks)

Ans:



Assembly language program

```
        ORG   0000H
        MOV   A, #00H          ;Load A=00H
UP :    MOV   P1,A             ;Send A to P1
        INC   A                ;Increment A
        NOP                    ;delay
        CJNE  A, # 0FF,UP              ;Compare A with Max. amplitude
UP1 :   MOV   P1,A                     ;Copy A to P1
        DEC   A                        ;Decrement A
        NOP                    ;delay
        CJNE  A, #00H, UP1             ;Compare A with minimum 00H
        SJMP  UP                       ;Repeat
```

OR

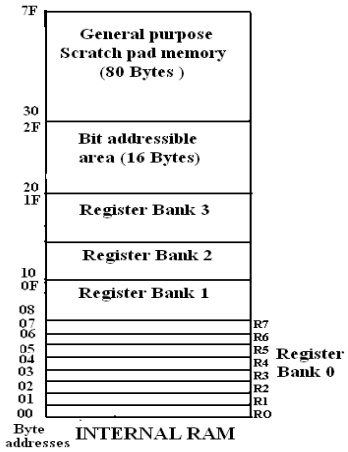
C Program

```
 # include<intel\8052.h>
# include<standard.h>
Unsigned int c ,i=255;
 Void main()
 {
        While(1)
        {
        For(c=0;c<=255;c++)
        {
        P1=c;
        Delay-ms(100);
        If(c==255)
                        {
        For(i=255;i>=0;i--)
        {
                        P1=I;
                        Delay_ms(100);
                        c=0 ;
                        }
                }
        }
   }
}
```

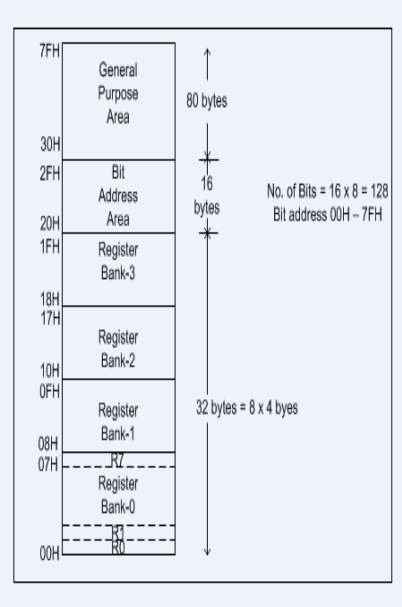
```
void Delay-ms(unsigned int t)
{ unsigned int i,j;
  for (i=0;i<=t;i++)
  for(j=0;j<1275;j++);
}
```

(ii) State the size of internal RAM of 8051 µc. Draw the internal RAM structure and describe it.
(Size of internal RAM – 1 mark; Diagram - 3 marks; description - 2 marks)

Ans:



OR



Internal RAM or Data memory: The 128 bytes of internal RAM is organized as shown in the diagram. It is volatile, so when the 8051 is reset this memory is cleared

1. Four register banks (Bank0, Bank1, Bank2 and Bank3) each of 8-bits (total 32 bytes). The default bank register is Bank0. The remaining Banks are selected with the help of RS0 and RS1 bits of PSW Register.
2. 16 bytes of bit addressable area
3. 80 bytes of general purpose area (Scratch pad memory) as shown in the diagram. This area is also utilized by the microcontroller as a storage area for the operating stack.

2. Attempt any FOUR of the following: 16

a) Explain the following instructions of 8051 µc

(i) SWAPA
(ii) MOVC A, @ A+DPTR
(iii)RRC A
(iv)RETI
(Example - ½ marks; Description - 1/2 marks; each 1 mark)

Ans:

(i) SWAP (swaps LSB with MSB bits or swaps bits 0-3 of the Accumulator with bits 4- 7 of the Accumulator**)**
The SWAP instruction exchanges the low-order and high-order nibbles within the accumulator. No flags are affected by this instruction.
If Accumulator contents is 78H then after execution of swap A
Content of accumulator is 87 H

(ii) MOVC A @ A+DPTR
Move Code / Data Byte to Accumulator MOVC moves a byte from Code Memory into the Accumulator. The Code Memory address from which the byte will be moved is calculated by summing the value of the Accumulator with DPTR

Example: if A contents is 05h and DPTR value is 3000 the code memory forms 3005 and its content will be saved in A after executing this instruction

(iii) RRCA (Rotate Accumulator Right through Carry)

Shifts the bits of the Accumulator to the right. The right-most bit (bit 0) of the Accumulator is loaded into the Carry Flag, and the original Carry Flag is loaded into bit 7. This function can be used to quickly divide a byte by 2

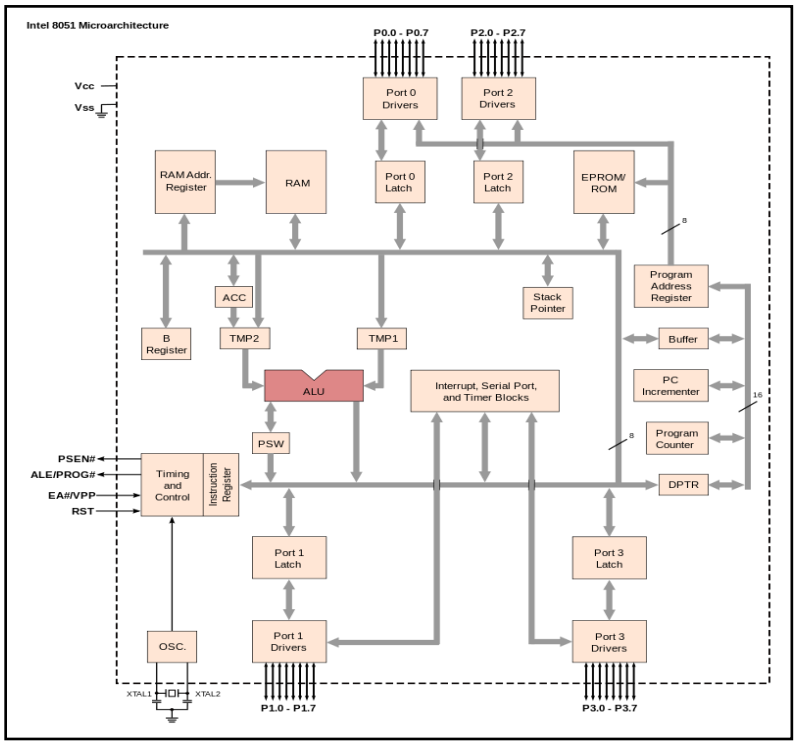
(iv) RETI (return from interrupts)

RETI is used to return from an interrupt service routine. RETI first enables interrupts of equal and lower priorities to the interrupt that is terminating. Program execution continues at the address that is calculated by popping the topmost 2 bytes off the stack. The most-significant-byte is popped off the stack first, followed by the least-significant-byte.
RETI functions identically to RET if it is executed outside of an interrupt service routine.

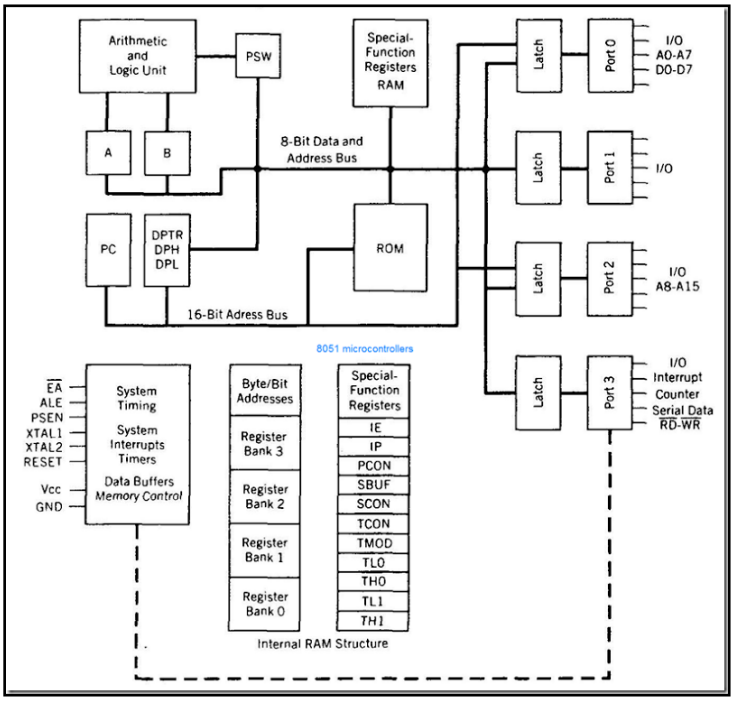
b) Draw the labeled architecture of 8051 µc.

(Labeling - 2 marks; diagram - 2 marks)

Ans:



OR



c) Write an assembly language program to multiply two 8 bit numbers stored in internal RAM locations 20H and 21H. Store the result at 22H and 23H.
(Program - 4 marks)

[NOTE: Comments are not compulsory**]**

Ans:

```
ORG 000H
MOV A, 20H ;  COPY CONTENT OF 20H TO A
MOV B, 21H ; COPY CONTENT OF 21H TO B

MUL AB; MULTIPLY THE CONTENT A WITH B
MOV 22H, A; STORE  LSB TO 22 RAM LOCATION
MOV 23H,B ;  STORE MSB TO 23 RAM LOCATION
```

END

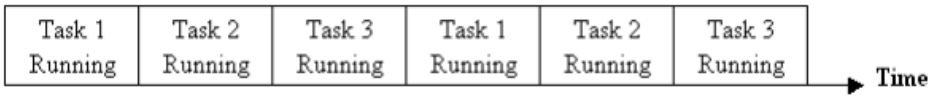
[Note: consider another methods if program is correct by another method **]**

d) Describe round robin scheduling algorithm with suitable diagram.

(Diagram - 1 mark; Description - 3 marks)

Ans:

Round-robin Scheduling Algorithm



Round-robin Scheduling 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 diagram if their tasks 1,2and 3 are waiting in the queue, the CPU first executes task1 then task2 then task3 and then task1.

The kernel gives control to the next task if

a. The current task has completed its work 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 note that there are no priorities for any task. All tasks are considered of equal importance. If time critical operations are not involved then this algorithm will be sufficient. Digital multi meters and microwave ovens use this scheduling algorithm.

e) Draw the format of TMOD register of 8051 microcontroller and state function of each bit.

(Format - 2 marks; function - 2 marks)

Ans:

TMOD Format:

Bit number	D7	D6	D5	D4	D3	D2	D1	D0
Type	**GATE1**	**C/T1**	**M1**	**M0**	**Gate0**	**C/T0**	**M1**	**M0**

Bits of this register have the following function:

- **GATE1** 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/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

- **GATE0** enables and disables Timer 1 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/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.

f) State the advantages of embedded system.

(Any 4; each 1 mark)

Ans:

Advantages of ES

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.

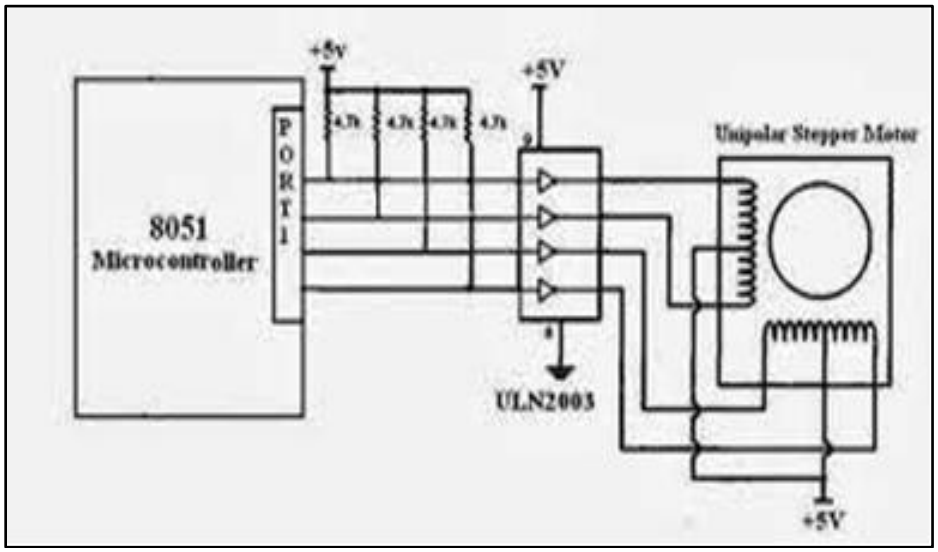
2. **Cost:**
   The streamline make-up of most embedded system allows their parts to be smaller less expensive to produce.
3. **Accessibility:**
   If something goes wrong with certain embedded systems they can be too inaccessible to repair. This problem is addressed in the design stage, so by programming 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

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

a) Draw an interfacing diagram of stepper motor with 8051 microcontroller. Write a program to rotate it in clockwise direction
(Diagram - 2 marks; program - 2 marks)

[NOTE: Program change. Student can also use the other logic. Please check the logic and understanding of students. Codes can be 03h, 09h, 0CH, 06H instead of codes given below**]**

Ans:



Program written in C or in Assembly should be given marks.

```
     ORG 0000h
     MOV A, #33h
L1: MOV P1, A
    RRA….For clockwise direction
    ACALL delay
   SJMP L1
```

OR

```
ORG 0000h
 L1:   MOV A, #33h
       MOV P1, A
       ACALL delay
       MOV A, #99h
       MOV P1, A
       ACALL delay
      MOV A, #0CCh
      MOV P1, A
       ACALL delay
      MOV A, #66h
      MOV P1, A
       ACALL delay
       SJMP L1
delay:  MOV R2,#255
L1: MOV R3,#255
L2: DJNZ R3,L2
     DJNZ R2,L1
    RET
```

OR

```
#include<reg51.h>
void delay(unsigned int);
void main()
{ while(1)
 { P1=0x03;
 delay(100);
```

```
P1=0x09;
 delay(100);
P1=0x0C;
 delay(100);
P1=0x06;
 delay(100);
}
}
Void delay(unsigned int t)
{ unsigned int i,j;
   for (i=0;i<=t;i++)
   for(j=0;j<1275;j++);
}
```

b) List the software development tools used in embedded system. State the function of any two ***(List - 2 marks; explanation each - 1 mark)***

Ans*:*

Various software development tools in microcontroller based systems are:-

1. Compiler
2. Cross assembler
3. Cross compiler
4. Locators
5. Simulators
6. Debugger
7. Editor
8. Assembler

1. **Compiler:-** Instructions in assembly language are represented in the form of executable code by the 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. **Cross assembler:-**It is useful to convert object codes for microcontrollers or processor to other codes for another microcontrollers or processor and vice versa.
3. **Cross compiler: -** It is used to create executable code other than one on which the compiler is run. They are used to generate executables for embedded systems or multiple platforms.
4. **Linker/Locator:-** It is used for relocation process . It is done during compilation also it can

be done at run time by a relocating loader. It is a program that takes one or more objects generated by compiler and combines them into a single executable program also generate .abs file.

5. **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
6. **Editor:** An editor is a program which helps you to construct your assembly language program in right format so that the assembler will translate it correctly to machine language. So, you can type your program using editor. This form of your program is called as source program and extension of program must be .asm or .src depending on which assembler is used.
   The DOS based editor such as EDIT, WordStar, and Norton Editor etc. can be used to type your program.
7. **Assembler:** An assembler is programs that translate assembly language program to the correct binary code for each instruction i.e. machine code and generate the file called as Object file with extension .obj and list file with extension
   .lst extension.
   Some examples of assembler are ASEM-51, Keil's A51, AX 51 and C51, Intel PL/M-51 etc. Some examples of linker are ASEM-51 BL51, Keil uVision Debugger, LX 51 Enhanced Linker etc.

c) Write a program to sent message "WELCOME" serially at 9600 baud rate continuously using assembly or C language.
(Program - 4 marks)
(Calculations not compulsory)

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 = -3
Hex count=255-3=252=FDh

```
ORG 0000h
MOV TMOD, #20h
MOV TH1,#0FDh
MOV SCON, #50h
SETB TR1
REPEAT : MOV DPTR, #mydata
MOV R0, #07
Up: CLR A
MOVC a,@A+DPTR
MOV SBUF,A
JNB TI,$
INC DPTR
CLR TI
DJNZ R0,up
SJMP REPEAT
mydata: DB 'WELCOME'
END
```

OR

```
MOV TMOD, #20h
MOV TH1,#0FDh
MOV SCON, #50h
SETB TR1
MOV SBUF, #'W'
JNB TI, $
CLR TI
MOV SBUF, #'E'
JNB TI, $
CLR TI
MOV SBUF, #'L'
JNB TI, $
CLR TI
```

```
MOV SBUF, #'C'
JNB TI, $
CLR TI
MOV SBUF, #'O'
JNB TI, $
CLR TI
MOV SBUF, #'M'
JNB TI, $
CLR TI
MOV SBUF, #'E'
JNB TI, $
CLR TI
SJMP $
END
```

OR

```
#include<reg51.h>
void SERTX(unsigned char);
void main()
{ TMOD=0x20;
TH1=0xFD;
SCON=0x50;
TR1=1;
while(1)
{ SERTX('W');
SERTX('E');
SERTX('L');
SERTX('C');
SERTX('O');
SERTX('M');
SERTX('E');
}
}
void SERTX(unsigned char x)
{SBUF=X;
while(TI==0);
TI=0;
}
```

d) State any four features of 8051 μc.
(Any 4 features - 1 mark each)

Ans:

- It is an 8bit microcontroller.
- 8bit accumulator, 8bit Register and 8bit ALU.
- On chip RAM 128 bites (data memory).
- On chip ROM 4 Kbytes (program memory).
- Two 16bit counter/ timer.
- A 16 bit dptr(data pointer)
- Two levels of interrupt priority.
- 4 byte bi-directional input/ output port.
- Power saving mode (on some derivatives).
- 16bit address bus:-it can access 2^16 memory locations:-64kb (65536) each of RAM and ROM.
- It is an inclusion of Boolean processing system, have an ability to allow logic operations to be carried out on registers and RAM.
- 8bit data bus:-it can access 8bit of data in one operation.
- It also consist of 3 internal and two external interrupts
- UART (this serial communication port makes chip to use simply as a serial communication interface)
- It has four separate Register set. (Each contains 8 Registers (R0 to R7)).

e) Describe the concept of starvation.
(Concept - 4 Marks)

Ans:

Starvation:

- CPU starvation occurs when higher priority tasks use all of CPU execution time & lower priority tasks use all of CPU execution time & lower priority tasks do not get to run.
- The possibility of blocked states is extremely important in real time systems because without blocked states, lower priority tasks could not run.
- If higher priority tasks are not designed to block, CPU starvation can result
- Aging is a technology of gradually increasing the priority of processes that wait in the system for a long time.

Marks

4. a) Attempt any THREE of the following: 12

(i) Describe the addressing modes of 8051 microcontroller with suitable examples.
(Description of any 4 - each 1 mark)
[NOTE: Explanation of examples is not necessary**]**

Ans:

1) Immediate Addressing mode:

Immediate addressing simply means that the operand (which immediately follows the Instruction op. code) is the data value to be used.

For example the instruction: MOV A, #25H ; Load 25H into A

Moves the value 25H into the accumulator The # symbol tells the assembler that the immediate addressing mode is to be used.

2) Register Addressing Mode:

One of the eight general-registers, R0 to R7, can be specified as the instruction Operand. The assembly language documentation refers to a register generically as Rn.

An example instruction using register addressing is :
ADD A, R5 ; Add the contents of register R5 to contents of A (accumulator)

Here the contents of R5 are added to the accumulator. One advantage of register addressing is that the instructions tend to be short, single byte instructions.

3) Direct Addressing Mode:

Direct addressing means that the data value is obtained directly from the memory location specified in the operand.

For example consider the instruction:
MOV R0, 40H; Save contents of RAM location 40H in R0.

The instruction reads the data from Internal RAM address 40H and stores this in theR0.
Direct addressing can be used to access Internal RAM, including the SFR registers.

4) Register Indirect Addressing Mode:

Indirect addressing provides a powerful addressing capability, which needs to be appreciated. An example instruction, which uses indirect addressing, is as follows:
MOV A, @R0; move contents of RAM location whose address is held by R0 into A
Note the @ symbol indicated that the indirect addressing mode is used. If the data is inside the CPU, only registers R0 & R1 are used for this purpose.

5) Relative Addressing Mode:

This is a special addressing mode used with certain jump instructions. The relative address, often referred to as an offset, is an 8-bit signed number, which is automatically added to the PC to make the address of the next instruction. The 8-bitsigned offset value gives an address range of + 127 to –128 locations.
Consider the following example: SJMP LABEL_X
An advantage of relative addressing is that the program code is easy to relocate in memory in that the addressing is relative to the position in memory.

6) Absolute addressing Mode:
Absolute addressing within the 8051 is used only by the AJMP (Absolute Jump) and ACALL (Absolute Call) instructions.

7) Long Addressing Mode:

The long addressing mode within the 8051 is used with the instructions LJMP and LCALL. The address specifies a full 16 bit destination address so that a jump or a call can be made to a location within a 64KByte code memory space (216 = 64K).
An example instruction is:
LJMP 5000h; full 16 bit address is specified in operand

8) Indexed Addressing Mode:

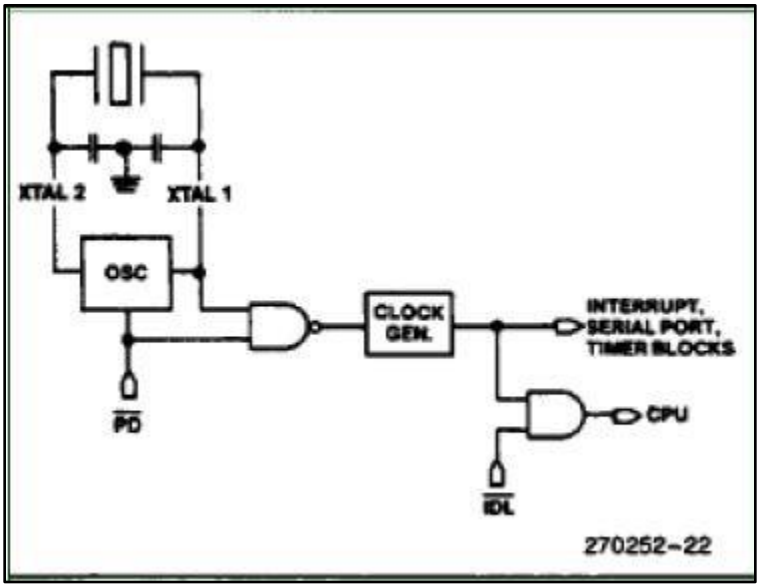
With indexed addressing a separate register, either the program counter, PC, or the data pointer DTPR, is used as a base address and the accumulator is used as an offset address. The effective address is formed by adding the value from the base address to the value from the offset address. Indexed addressing in the 8051 is used with the JMP or MOVC instructions. Look up tables are easy to implement with the help of index addressing.
Consider the example
instruction:
MOVC A, @A+DPTR
MOVC is a move instruction, which moves data from the external code memory space. The address operand in this example is formed by adding the content of the DPTR register to the accumulator value. Here the DPTR value is referred to as the base address and the accumulator value us referred to as the index address.

(ii) Describe the power saving mode of 8051 microcontroller.
(Diagram - 1 mark; explanation - 1.5 marks each)

[NOTE: PCON bit pattern is not necessary**]**

Ans:

***Power Saving Option*:**



IDLE MODE

In the Idle mode, the internal clock signal is gated off to the CPU, but not to the Interrupt, Timer, and Serial Port functions.

The CPU status is preserved in its entirety, the Stack Pointer, Program Counter, Program Status Word, Accumulator, and all other registers maintain their data during Idle. The port pins hold the logical state they had at the time idle mode was activated. ALE and PSEN hold at logic high levels.

There are two ways to terminate the idle mode.

i) Activation of any enabled interrupt will cause PCON.O to be cleared and idle mode is terminated.

ii) Hard ware reset: that is signal at RST pin clears IDEAL bit IN PCON register directly. At this time, CPU resumes the program execution from where it left off.

POWER DOWN MODE

An instruction that sets PCON.1 causes that to be the last instruction executed before going into the Power Down mode. In the Power Down mode, the on-chip oscillator is stopped. With the clock frozen, all functions are stopped, but the on-chip RAM and Special Function Register are

maintained held. The port pins output the values held by their respective SFRS. ALE and PSEN are held low.

Termination from power down mode:
An exit from this mode is hardware reset. Reset defines all SFR but doesn't change on chip RAM

(iii) Write a program to transfer five bytes from source to destination in assembly or C language. Assume source address is 20H onwards and destinations address 30H onwards.
(Program- 4 marks)

Ans:

```
ORG 0000h
MOV R0,#20h
MOV R1,#30h
CLR A
MOV R2,#05h
L1: MOV A,@R0
    MOV @R1,A
INC R0
INC R1
DJNZ R2,L1
SJMP $
```

(iv) Explain in brief the concept of device driver.
[NOTE: Marks should be given student has explained the concept**]**

Ans:

Device Driver:

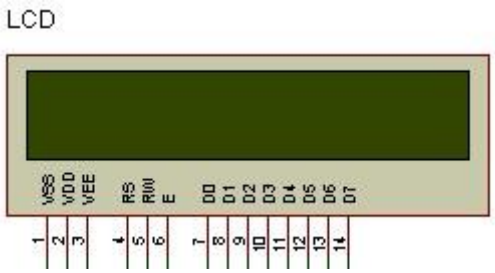
- Embedded system hardware has devices which communicate through serial & parallel ports & buses, There also may be ports for real time voice and video I/Os
- A device access is required for opening, connecting, binding, reading, and writing, disconnecting or closing it. Processor accesses a device using 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.

b) Attempt any ONE of the following: 6

(i) Draw the pin diagram of 16 × 2 LCD display and state the function of following pins :
1) **RS**
2) **R/W**
3) **EN**

(Diagram -1 mark; Function of every pin -1 mark each)

Ans:



Explanation of RS, R/W and EN

(i) 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.

(ii) RW: - R/W gives you the choice between writing and reading. R/W=1, reading is enabled.

R/W=0 then writing is enabled.

(iii)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.

(ii) State the method of task synchronization. Describe semaphore with suitable example.
(List of methods - 2 marks, Explanation of any method - 2 marks)

Ans:

Method of task synchronizations are

- Semaphores
- Message queue,
- Mutual Exclusion
- Dead Lock
- Mailboxes
- Message Queues

o Tasks, along with task-management services, allow developers to design applications for concurrency to meet multiple time constraints and address various design problems inherent to real-time embedded applications.

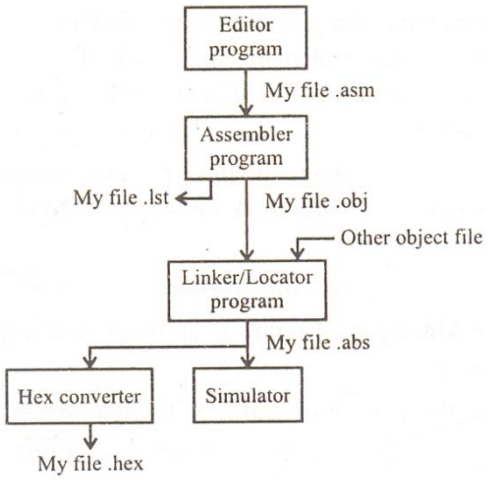
Semaphores:-

- 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
  - Counting
  - Mutex

5. Attempt any FOUR of the following: **16**

a) With the help of flow chart explain the different files created while execution of program using keil compiler.
(Flowchart: 2 marks; Explanation -2 marks)

Ans:



STEP1: We edit assembly language program by using various editors like MSDOS editor, notepad, Keil IDE etc. The extention of the file should be .asm. after typing the program, we must save the file.

STEP2: The .asm. File is fed to an 8051 assembler. The assembler converts assembly instructions into machine code. The assembler produces object file (.obj) and list file (.lst)

STEP3: The main object file and other object file are fed to linker/locator. The linker/locator generates absolute object file (.abs)

STEP4: The absolute object file is fed to simulator to run the program. The simulator simulates 8051 kit. The absolute object files into hex file (.hex). This hex file is actual machine code of 8051.which can be downloaded into program memory.

b) Draw the format of IE and IP register of 8051 µc and describe function of each bit.
(Format of SFR - 1 mark each; explanation of all bits in SFR - 1 mark each for each SFR)

Ans:

IE:

EA	-	ET2	ES	ET1	EX1	ET0	EX0

EX0: External interrupt0 ($\overline{\text{INT0}}$) enable bit
ET0: Timer-0 interrupt enable bit
EX1: External interrupt1 ($\overline{\text{INT1}}$) 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 interrupts if bit=1, otherwise interrupts are disabled if bit=0

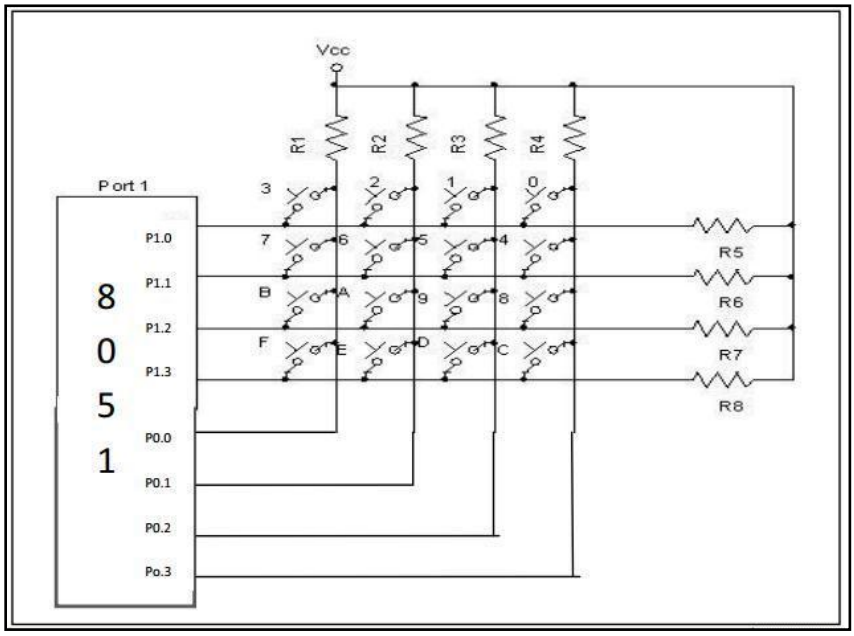
IP:

7	6	5	4	3	2	1	0
——	——	PT2	PS	PT1	PX1	PT0	PX0

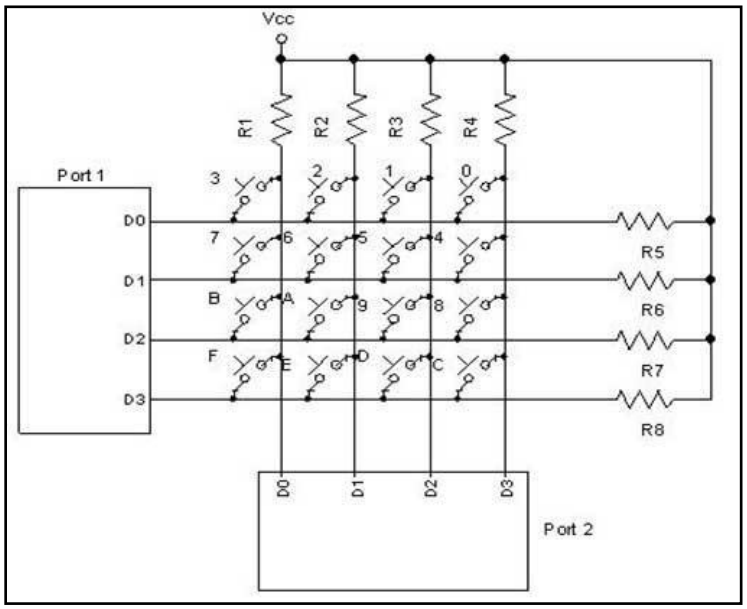
PT2: reserved for future use, **PS:** serial interrupt priority
PT1: timer1 interrupt priority, **PX1:** INT1 interrupt priority
PT0: timer0 interrupt priority, **PX0:** INT0 interrupt priority
Priority may be 1 (highest) or 0 (lowest) by setting or resetting the individual bits.

c) Draw the interfacing of 4 × 4 keyboard with 8051 µc.
(Correct diagram - 4 marks)

Ans:



OR

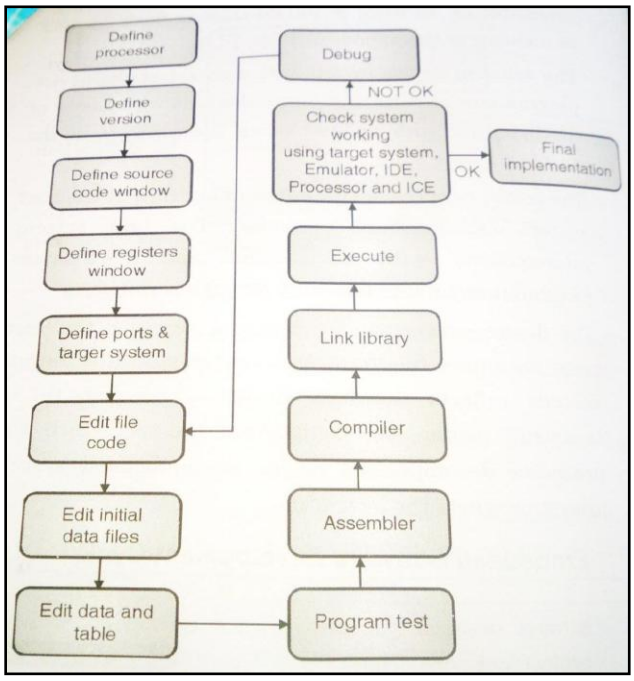


d) With suitable flow chart list the steps involved in the embedded software development cycle.
(Flowchart - 2 marks; listing the steps - 2 marks)
[NOTE: Explanation not compulsory**]**

Ans:

Steps

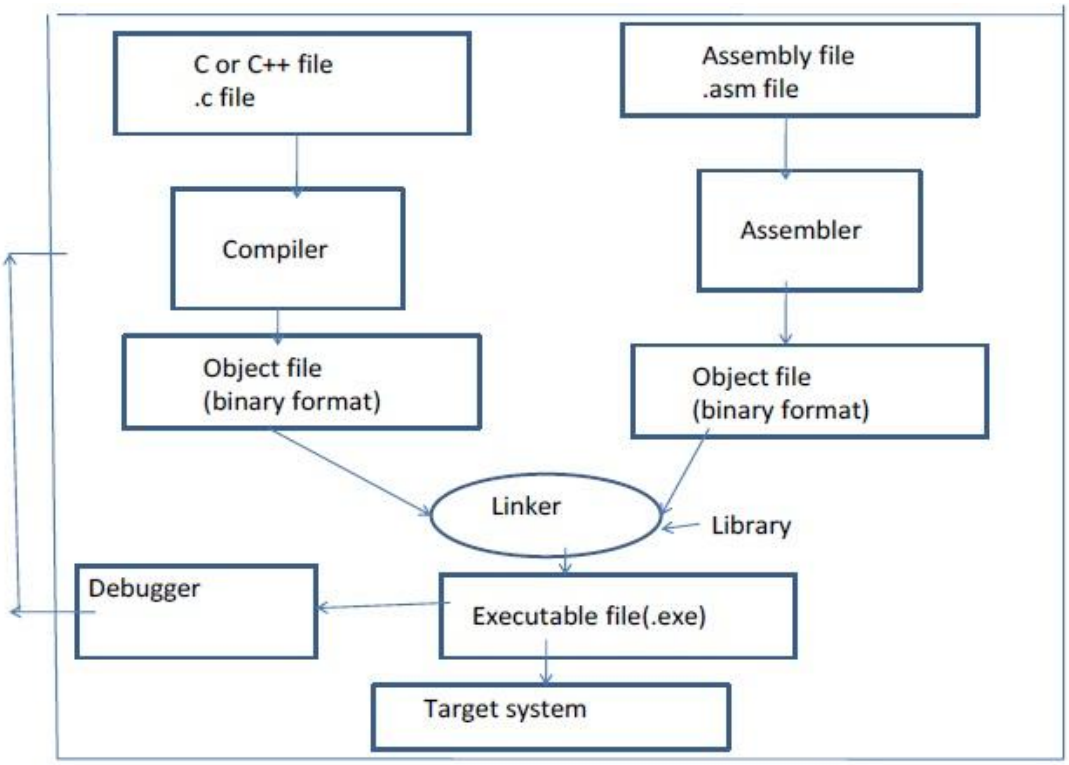
1) Define the processor /processing device (family and version) for the target system.
2) Defining the source code window with labels and symbolic arguments as execution goes on for each single step.
3) Define the processor registers for each step /module.
4) Define details of ports and target system.
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.



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



e) Describe any four specifications of RTOS.
(Any four specifications – 1 mark each)

Ans:

- **Reliability**

  Embedded systems must be reliable. Depending on the application, the system might need to operate for long periods without human intervention.

  Different degrees of reliability may be required. For example, a digital solar-powered calculator might reset itself if it does not get enough light, yet the calculator might still be considered acceptable. On the other hand, a telecom switch cannot reset during operation without incurring high associated costs for down time. The RTOS in these applications require different degrees of reliability.

  Note that the RTOS by itself is not what is measured to determine system reliability. It is the combination of all system elements-including the hardware, BSP, RTOS, and application-that determines the reliability of a system.

- **Predictability**

  The RTOS used in this case needs to be predictable to a certain degree. The term deterministic describes RTOSes with predictable behavior, in which the completion of operating system calls occurs within known timeframes.

  Developers can write simple benchmark programs to validate the determinism of an RTOS. The result is based on timed responses to specific RTOS calls.

  In a good deterministic RTOS, the variance of the response times for each type of system call is very small.

- **Performance**

  This requirement dictates that an embedded system must perform fast enough to fulfill its timing requirements.

  Typically, the more deadlines to be met-and the shorter the time between them-the faster the system's CPU must be. Although underlying hardware can dictate a system's processing power, its software can also contribute to system performance.

  Typically, the processor's performance is expressed in million instructions per second (MIPS).

  Throughput also measures the overall performance of a system, with hardware and software combined.

- **Compactness**

  Application design constraints and cost constraints help determine how compact an embedded system can be. For example, a cell phone clearly must be small, portable, and low cost. These design requirements limit system memory, which in turn limits the size of the application and operating system.

  In such embedded systems, where hardware is limited due to size and costs, the RTOS clearly must be small and efficient. In these cases, the RTOS memory footprint can be an important factor. To meet total system requirements, designers must understand both the static and dynamic memory consumption of the TOS and the application that will run on it.

- **Scalability**

  Because RTOSes can be used in a wide variety of embedded systems, they must be able to scale up or down to meet application-specific requirements.

  Depending on how much functionality is required, an RTOS should be capable of adding or deleting modular components, including file systems and protocol stacks.

6. Attempt any FOUR of the following : 16

a) Write an assembly or C language program to generate a square wave of $1kH_2$ on pin P1.5 using Timer 0 in mode 1. Assume crystal frequency as 12 MHz.
(Calculation - 1 mark; correct program - 3 marks)
[NOTE: Comments not compulsory**]**

Ans:

The period of the square wave = 1 /1 KHz Hz = 1 ms

- The T_{ON} (high time) or T_{OFF} (low time) of the square wave = 1ms/2 = 0.5 ms
- Crystal is 12MHz, thus T_{MC} = 1µs
- No. of machine cycles = 0.5 ms / 1µs = 500
- Count to be put in registers: 65536 – 500 = 65036 in decimal = FE0CH
  TL0 = 0CH and TH0 = 0FEH

//C language program

```
#include <8051.h>
Void T0M1delay (void);          //Timer 0, Mode 1(16 bit timer)
SBIT OUTPUT P1^5;               // Initialize Port pin P1.2 as output
Void main ( )
{
        While (1)
{
OUTPUT= ~ OUTPUT;               // toggle P1.5
T0M1delay ();                   // delay of 500 µs i.e. 0.5ms
}
}

Void T0M1delay ( )
{
TMOD = 0x01;                    // Timer 0, Mode 1(16 bit timer)
TL0 = 0x0C;                     //Load TL0 = 0Ch
TH0 = 0xFE;                     //Load TL0 = FEh
TR0 = 1;                        //Run the timer 0
while (TF0 = = 0);              // Wait for TF0 to overflow
TR0 = 0;                        //Stop the timer 0
TF0 = 0;                        //Clear TF0
}
```

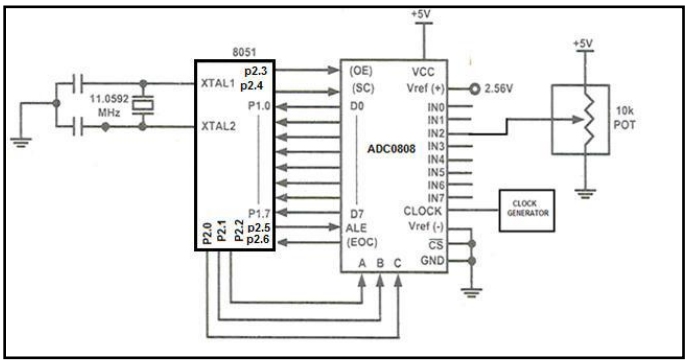
OR

//assembly program

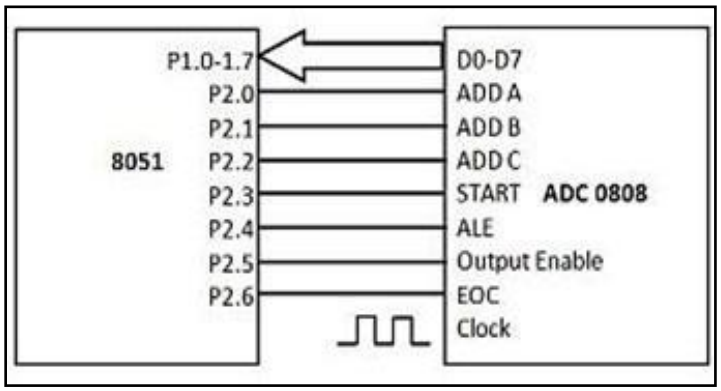
```
Start: MOV TMOD, #01H
UP:  MOV TL0, # 0CH
        MOV TH0, #0FEH
CPL P1.5
        SETB TR0
BACK: JNB TF0 , BACK
CLR TF0
CLR TR0
SJMP UP
```

b) Draw the interfacing diagram of ADC with 8051 microcontroller.
(Correct diagram - 4 marks)

Ans:

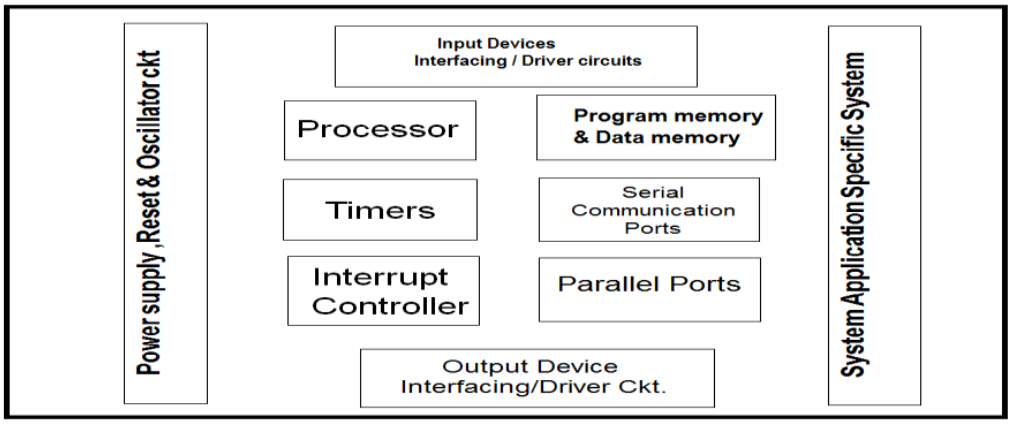


OR

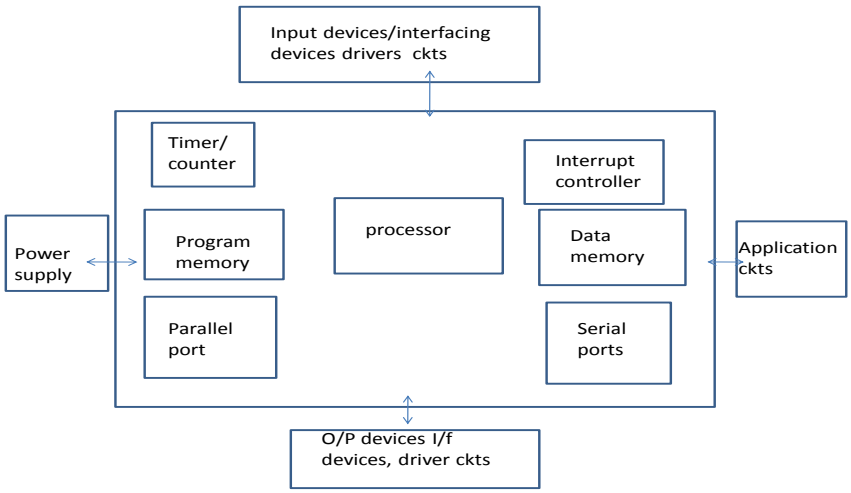


c) Draw the block diagram of embedded system and describe the hardware units of an embedded system.
(Diagram - 2 marks; explanation of any 4 - 2 marks)

Ans:



OR



Various hardware units of embedded system are:

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 An embedded system processor chip can be one of the following
2. **Power supply, reset & oscillator circuit:**
   Most of the systems have their own power supply. Some embedded system do not have their own power supply. These embedded systems are powered by external power supply e.g. USB based embedded system, network interface card, Graphics Accelerator etc. are powered by PC power supply.
   Reset means that processor begins processing of instructions from starting address set by default in program counter on power up.
   Watchdog Timer reset.
   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.

d) Describe the concept of mutual exclusion and deadlock.
(Mutual exclusion - 2 marks; starvation -2 marks)

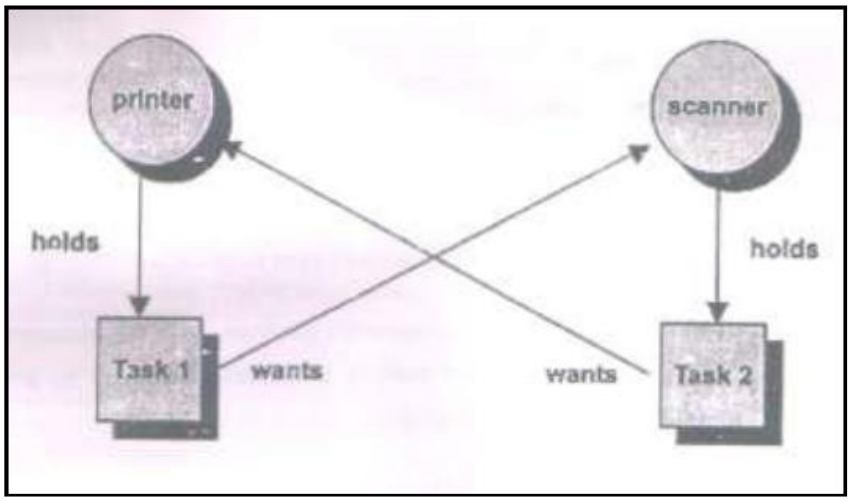
Ans:

Mutual exclusion: A resource can be accessed by only one task at a time, i.e., exclusive access mode. The easiest way for threads to communicate with each other is through shared data structures. This is especially easy when all threads exist in a single address space and can reference global variables, pointers, buffers, linked lists, FIFOs, etc. Although sharing data simplifies the exchange of information, you must ensure that each thread has exclusive access to the data to avoid contention and data corruption. The most common methods of obtaining exclusive access to shared resources are:

- Disabling interrupts,
- Performing test -and-set operations,
- Disabling scheduling, and
- Using semaphores.

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.Neither thread can continue.They are deadlocked.



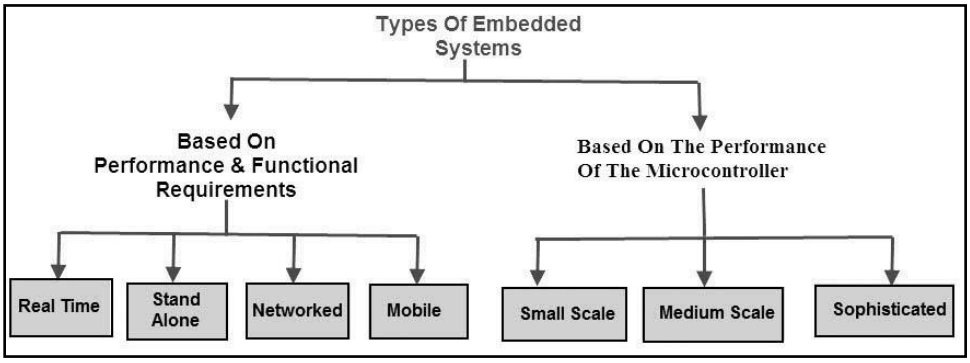
The simplest way to avoid a deadlock is for threads to:

- 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.

e) Explain the classification of embedded system.
(1 mark can be given even if only diagram is drawn.)

Ans:



Brief explanation:

(a) Real Time Embedded Systems

A real time embedded system is defined as, a system which gives a required o/p in a particular time. These types of embedded systems follow the time deadlines for completion of a task. Real time embedded systems are classified into two types such as soft and hard real time systems.

(b) Stand Alone Embedded Systems

Stand-alone embedded systems do not require a host system like a computer, it works by itself. It takes the input from the input ports either analog or digital and processes, calculates and converts the data and gives the resulting data through the connected device-Which either controls, drives or displays the connected devices. Examples for the stand alone embedded systems are mp3 players, digital cameras, video game consoles, microwave ovens and temperature measurement systems.

(c) Networked Embedded Systems

These types of embedded systems are related to a network to access the resources. The connected network can be LAN, WAN or the internet. The connection can be any wired or wireless. This type of embedded system is the fastest growing area in embedded system applications. The embedded web server is a type of system wherein all embedded devices are connected to a web server and accessed and controlled by a web browser. Example for the LAN networked embedded system is a home security system wherein all sensors are connected and run on the protocol TCP/IP

(d) Mobile Embedded Systems

Mobile embedded systems are used in portable embedded devices like cell phones, mobiles, digital cameras, mp3 players and personal digital assistants, etc. The basic limitation of these devices is the other resources and limitation of memory.

(e) Small Scale Embedded Systems

These types of embedded systems are designed with a single 8 or 16-bit microcontroller that may even be activated by a battery. For developing embedded software for small scale embedded systems, the main programming tools are an editor, assembler, cross assembler and integrated development environment (IDE).

(f) Medium Scale Embedded Systems

These types of embedded systems design with a single or 16 or 32 bit microcontroller, RISCs or DSPs. These types of embedded systems have both hardware and software complexities. For developing embedded software for medium scale embedded systems, the main programming tools are C, C++, and JAVA, Visual C++, and RTOS, debugger, source code engineering tool, simulator and IDE.

(g) Sophisticated Embedded Systems

These types of embedded systems have enormous hardware and software complexities that may need ASIPs, IPs, PLAs, scalable or configurable processors. They are used for cutting edge applications that need hardware and software Co-design and components which have to assemble in the final system.