Q.1 Attempt any FIVE of the following : [10]

Q.1(a) Draw internal RAM organisation of 89C51 microcontroller. Explain register banks in it.

Ans.:

The 89c51 has 128 byte internal RAM and divided into 3 parts/area.

In first part:
32 bytes from addresses 00H to 1FH i.e. 32 registers organised in 4 banks. Each bank contains 8 registers (each of 8 bit) and available at each address.

Registers are named as:
- Bank 0 – R0, R1, R2, R3, R4, R5, R6, R7
- Bank 1 – R0, R1, R2, R3, R4, R5, R6, R7
- Bank 2 – R0, R1, R2, R3, R4, R5, R6, R7
- Bank 3 – R0, R1, R2, R3, R4, R5, R6, R7

If register banks are not selected, it can be used as general purpose area.

On reset Bank 0 is selected.

RS1 and RS0 bits (D3 and D4) of PSW register are used to select the register bank.

Note: Others parts of RAM not expected.

Q.1(b) State the function of simulator, linker compiler and debugger. [2]

Ans.: Simulator:
1. It is the software that functions like the hardware without actual hardware.
2. Allows simulation of peripherals and I/O devices.
3. Allows checking of software before the hardware is available to the user.
Linker:
1. Linker is used to link with the library and generation of executable file.
2. It is used for relocation process.
3. It is done during compilation also it can be done at run time by a relocating loader.
4. It is a program that takes one or more objects generated by compiler and combines them into a single executable program.

Compiler:
1. It is program which converts high level language program to machine language.
2. It also indicates the syntax errors in the program if any.
3. It generates object file corresponding to the target device.

Debugger:
1. Debugger is used to find the error and it keeps the control over the system environment and ability to test or follow the execution of the program.
2. Debugger allows the user to load program in to the system memory, executes the program by single stepping and detect logical errors in the program.

Q.1(c) Draw a labelled interfacing diagram of ADC 0808 with 8951 microcontroller. [2]

Ans.:

![Interfacing Diagram](image)

Note: Any labeled diagram showing handshaking signal of ADC (Data bus, Channel Select, ALE, START, end of conversion, output enable) Any port of 89C51 is considered for interfacing.

Q.1(d) Explain pre-emptive scheduling and round-robin scheduling algorithms in RTOS. [2]

Ans.: Preemptive scheduling:
Preemptive scheduling ensures that every task will get CPU time for execution. The allotment of CPU time depends on the preemptive scheduling algorithm. It allows a process to be interrupted in the middle of its execution, taking the CPU away and allocating it to another process. The scheduling algorithm has high overhead. System is costly and complex in design.

Shortest Job first:
Shortest Job first is the example of this preemptive scheduling. If a new process arrives with a CPU burst length less than the remaining time of the current executing process, pre-empt. This scheme is known as the Shortest-Remaining-Time-First (SRTF).

Priority scheduling in pre-emptive:
A preemptive approach will pre-empt the CPU if the priority of the newly-arrived process is higher than the priority of the currently running process. (diagram)
PREEMPTIVE KERNEL

Round-robin scheduling:
In the round robin algorithm, each process gets a small unit of CPU time (a time quantum), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue. If there are 'n' processes in the ready queue and the time quantum is 'q', then each process gets '1/n' of the CPU time in chunks of at most 'q' time units at once. No process waits more than '(n-1)q' time units.

Performance of the round robin algorithm when large then FCFS when small then 'q' must be greater than the context switch time; otherwise, the overhead is too high. One rule of thumb is that 80% of the CPU bursts should be shorter than the time quantum.

OR

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 task 1 then task 2 then task 3 and the again task 1 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.

Q.1(e) Define the terms: RISC and CISC

Ans.: RISC
- It stands for 'Reduced Instruction Set Computer'.
- The RISC processors have a smaller set of instructions with few addressing nodes.

CISC
- It stands for 'Complex Instruction Set Computer'.
- The CISC processors have a larger set of instructions with many addressing nodes.
Q.1(f) Illustrate any two data types used in C with their ranges.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Size in Bits</th>
<th>Data Range/Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned char</td>
<td>8-bit</td>
<td>0 to 255</td>
</tr>
<tr>
<td>(signed) char</td>
<td>8-bit</td>
<td>-128 to +127</td>
</tr>
<tr>
<td>unsigned int</td>
<td>16-bit</td>
<td>0 to 65535</td>
</tr>
<tr>
<td>(signed) int</td>
<td>16-bit</td>
<td>-32,768 to +32,767</td>
</tr>
<tr>
<td>sbit</td>
<td>1-bit</td>
<td>SFR bit-addressable only</td>
</tr>
<tr>
<td>bit</td>
<td>1-bit</td>
<td>RAM bit-addressable only</td>
</tr>
<tr>
<td>sfr</td>
<td>8-bit</td>
<td>RAM addresses 80-FFH only</td>
</tr>
</tbody>
</table>

Q.2 Attempt any THREE of the following:

Q.2(a) A 230 V AC bulb is connected through a relay at P2.2. A light sensor is connected at P3.4. A light sensor produces logic high in dark condition. Write a 'C' program to switch 'ON' the bulb in 'DARK' condition and switch it OFF in 'LIGHT' condition.

Ans.:

```c
#include<reg51.h>
sbit relay = P2^2;
sbit sensor = P3^4;
void main (void)
{
    sensor=1; //P3.4 as input line
    relay = 0; // Initially OFF
    while (1)
    {
        while ( sensor == 0); // check sensor if no dark wait//
        relay = 1; // dark ON bulb//
        while ( sensor == 1); // wait till dark//
    }
}
```

OR

```c
#include<reg51.h>
sbit relay = P2^2;
sbit sensor = P3^4;
void main (void)
{
    sensor=1; //P3.4 as input line
    relay = 0; // Initially OFF
    while (1)
    {
        if ( sensor == 1) // check sensor if dark //
            relay = 1; // ON bulb/
        else
            relay=0;
    }
}
```

Q.2(b) Describe inter task communication in RTOS.

Ans.: Software is the basic building block of RTOS. Task is a simply subroutine. Task must be able to communicate with one another to coordinate their activities or to share the data. Kernel object is used for inter task communication. Kernel objects uses message queue, mail box and pipes, Shared Memory, Signal Function and RPC a for inter task communication.
Message queue:
A message queue is a buffer like data structure, through which tasks and ISRs communicate with each other by sending and receiving messages and synchronize with data. It temporarily stores the message from a sender until the intended receiver is ready to read them. A message queue has queue control block, queue name, unique ID, memory buffers, a queue length. Kernel allocates the memory for message queue, ID, control block etc.

Mail box:
In general, mailboxes are similar to message queues. Mail box technique for inter task communication in RTOS based system used for one way messaging. The task/thread creates mail box to send the message. The receiver task can subscribe the mail box. The thread which creates the mail box is known as mailbox server. The others are known as client. RTOS has function to create, write and read from mail box. No of messages (limited or unlimited) in mail box have been decided by RTOS.

Pipes:
Pipes are kernel objects used for unstructured data exchange between tasks facilities synchronization among tasks. Pipe provides a simple data transfer facility.

Shared Memory:
Shared memory is simplest way of inter process communication. The sender process writes data into shared memory and receiver process reads data.

Signal Function:
Operating system provides the signal function for messaging among task (process).

Remote Procedure Call(RPC) and Sockets:
RPC is a mechanism used by process(task) to call the procedure of another process running on same or different CPU in the network. Sockets are used for RPC communication and establishes full duplex communication between tasks.

Q.2(c) Write a C language program to toggle all bits of P0, P1, P2 and P3 continuously with certain delay. [4]

Ans.:
```
#include <reg51.h>

void Add_delay (unsigned int);  //Declare Delay function
void main (void)
{
    //Main Program
    while(1) //repeat loop
    {
        P0=0xff;      //toggle all bits of port 0
        Add_delay (200);   //add delay
        P0=0x00;      //toggle all bits of port 0
        Add_delay (200);   //add delay
        P1=0xff;      //toggle all bits of port 1
        Add_delay (200);   //add delay
        P1=0x00;      //toggle all bits of port 1
        Add_delay (200);   //add delay
        P2=0xff;      //toggle all bits of port 0
        Add_delay (200);   //add delay
        P2=0x00;      //toggle all bits of port 2
        Add_delay (200);   //add delay
        P3=0xff;      //toggle all bits of port 3
        Add_delay (200);   //add delay
        P3=0x00;      //toggle all bits of port 3
```
Add_delay (200);  //add delay
}

Q.2(d) Draw the 9 pin RS 232C connector and state the significance of DTR and DSR signals.

Ans.: DB-9 Connector:

<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Data Carrier Detect (DCD)</td>
</tr>
<tr>
<td>2</td>
<td>Received Data (RXD)</td>
</tr>
<tr>
<td>3</td>
<td>Transmitted Data (TXD)</td>
</tr>
<tr>
<td>4</td>
<td>Data Terminal Ready (DTR)</td>
</tr>
<tr>
<td>5</td>
<td>Signal Ground (GND)</td>
</tr>
<tr>
<td>6</td>
<td>Data Set Ready (DSR)</td>
</tr>
<tr>
<td>7</td>
<td>Request To Send (RTS)</td>
</tr>
<tr>
<td>8</td>
<td>Clear To Send (CTS)</td>
</tr>
<tr>
<td>9</td>
<td>Ring Indicator (RI)</td>
</tr>
</tbody>
</table>

- **Data Terminal Ready**: When the Data Terminal Equipment (DTE) is turned on, it sends active low signal DTR and indicates that DTE is ready for communication.

- **Data Set Ready**: When modem is turned on, it sends an active low signal DSR and indicates that it is ready for communication.

Q.2(e) State any two applications for each of the following:

(i) Small Scale Embedded System.
(ii) Medium Scale Embedded System.

Ans.: (i) Small Scale Embedded System

- Normally these systems are designed using with an 8 bit or 16 bit microcontroller based system.
- They have very small hardware and software complexities and involved board level design and may be battery operated.
- While developing embedded software for these system, an editor, assembler or cross assembler are used for specific microcontroller or processor used.
- Usually, ‘C’ language is used for the development of the system as C language support machine level programming and C programs compilation is done in assembly.
- Executable codes are then appropriately located into system memory, so software must has to fit into memory available and also keep in mind the need to limit power consumption when system is continuously running.

(ii) Medium Scale Embedded System

- These systems are usually design using single or multiple 16 bit or 32 bit microcontrollers or Digital Signal Processors (DSP or Reduced instruction Set Computers (RISCs).
They have hardware and software complexities, so no board level design is possible. For such complex software design system, the development tools like it real Time Operating System (RTOS), Source code engineering tools, Simulator Debugger and Integrated Development Tools are required. These software tools also provide the solution for the hardware comp so assembler is rarely used. Such type of system may also use readily available Application Specific System Processor (ASSP) that is dedicated to specific task alone provide faster solution and Such type of system may also use readily available Intellectual Property (IP) gives a solution for synthesizing a higher level components that possess gal level sophistication in circuit above that of the counter, registers, floating point operation unit and ALU by configuring FPGA core or VLSI core.

Q.3 Attempt any THREE of the following: [12]

Q.3(a) Write a 'C' program to toggle P2.1 continuously with 100 ms delay. (Use simple delay subroutine).

Ans.: 
```c
#include <reg 51.h>
void add_delay (unsigned int);
sbit data_bit = P2^1;
void main (void)
{
    while (1)
    {
        data_bit =1; // set P2.1 bit
        add_delay (100);
        data_bit =0; // reset P2.1bit
        add_delay (100);
    }
}
void add_delay (unsigned int delay_time)
{
    unsigned int x,y;
    for (x = 0; x <delay_time; x ++)
    for (y =0; y<1275; y++);
}
```

Q.3(b) Compare desktop operating system with RTOS with any four points. [4]

Ans.: 

<table>
<thead>
<tr>
<th>Operating System</th>
<th>RTOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Non-Deterministic time behavior.</td>
<td>Deterministic time behavior.</td>
</tr>
<tr>
<td>2. Used in general desktop computer system.</td>
<td>Used in embedded system</td>
</tr>
<tr>
<td>4. OS services can inject random delays into application software, may cause slow responsiveness of an application at unexpected time.</td>
<td>OS services consumes only known and expected amounts of time regardless the number of services.</td>
</tr>
<tr>
<td>5. Slow context switching</td>
<td>fast context switching</td>
</tr>
<tr>
<td>6. More memory requirements</td>
<td>Less memory requirements</td>
</tr>
<tr>
<td>7. Ex: Windows XP, MS-DOS</td>
<td>Ex: Windows CE, Vx Works</td>
</tr>
</tbody>
</table>
Q.3(c) State any two advantages and disadvantages of embedded system. [4]

Ans.: Advantages:
1. Cost is low.
2. Small in size.
3. Highly reliable.
4. Operation is fast.
5. Easy for mass production.
7. Improves product quality.

Dis-Advantages:
1. Hard for maintenance as it is use and throw device.
2. No technological improvement.
3. Hard to back up of embedded files.
4. Less power supply durability if it is battery operated.

Q.3(d) Write a 'C' language program to rotate stepper motor by 90° clockwise. Assume step angle of 1.8° and 4 step sequence. [4]

Ans.: Minimum angle of rotation is 1.8°

\[
\text{Count} = \frac{\text{Total Angle needed}}{\text{Minimum angle of rotation}} = \frac{45°}{1.8°} = 25 \text{ decimal}
\]

\[
= 19_{\text{H}}
\]

<table>
<thead>
<tr>
<th>Sr. no.</th>
<th>Angle</th>
<th>Decimal</th>
<th>Hexadecimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>45</td>
<td>25</td>
<td>19</td>
</tr>
<tr>
<td>2</td>
<td>90</td>
<td>50</td>
<td>32</td>
</tr>
<tr>
<td>3</td>
<td>180</td>
<td>100</td>
<td>64</td>
</tr>
<tr>
<td>4</td>
<td>270</td>
<td>150</td>
<td>96</td>
</tr>
<tr>
<td>5</td>
<td>360</td>
<td>200</td>
<td>128</td>
</tr>
</tbody>
</table>

Program:
#include <reg51.h>
void delay(unsigned int y);
void rotate( )
{
    P1 = 0x03;   //0011
    delay(30);
    P1 = 0x06;   //0110
    delay(30);
    P1 = 0x0C;   //1100
    delay(30);
    P1 = 0x09;   //1001
    delay(30);
}
void delay(unsigned int y)
{
    int i, j;        //delay subroutine
    for(i = 0; i < y; i++)
        for(j = 0; j < 498; j++)
    }
void main()
{
    for(i=0;i<25;i++)  // For loop for Count of 25 ie 25*1.8 = 90°
Q.4 Attempt any THREE of the following:  

Q.4(a) Draw format of TMOD register. Find the value of TMOD register to operate timer 0 in mode 1.

Ans.: TMOD Format:

<table>
<thead>
<tr>
<th>Gate</th>
<th>C/T</th>
<th>M1</th>
<th>M0</th>
<th>Gate</th>
<th>C/T</th>
<th>M1</th>
<th>M0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Value in TMOD to Operate Timer 0 in mode 1: 01 H

Q.4(b) Write a 'C' program to generate a square wave of 5 kHz. (Operate timer 0 in mode 1).

Ans.: Crystal frequency = 11.0592 MHz

I/P clock = \( \frac{1}{2} \) (crystal frequency) = \( \frac{1}{2} \times 11.0592 \text{ MHz} = 921.6 \text{ kHz} \).

\( T_{in} = 1.085 \mu \text{sec} \)

For 5 KHz square wave

\( F_{out} = 5 \text{ KHz} \)

\( T_{out} = \frac{1}{5 \times 10^3} = 200 \mu \text{sec} \)

\( T_{out} = 200 \mu \text{sec} \)

Consider half of it = \( T_{out} = 100 \mu \text{sec} \)

\( N = \frac{T_{out}}{T_{in}} = \frac{100\mu}{1.085\mu} = 92.16 \)

65536 - 92 = (65444)D = FFA5 H

Program:

```c
#include<reg51.h>
void delay(void);

sbit p = P3>5;

void main(void)
{
    while (1)
    {
        p=~p;
        delay();
    }
}

void delay()
{
```
TMOD=0x01; //set timer 0 in mode 1 i.e. 16 bit number
TL0 = 0xA5H; //load TL register with LSB of count
TH0=0xFFH; //Load register with MSB of count
TR0 = 1; // Start timer 0
While(TF0==0); wait until timer rolls over
TR0=0; Stop timer 0
TF0=0 // Clear timer flag 0
}

Q. 4(c) Compare Zigbee and Bluetooth on the basis of following points:
(i) Modulation Technique. (ii) Communication Range.
(iii) Power Consumption. (iv) IEEE standard.

Ans.:

<table>
<thead>
<tr>
<th>Sr. No. Specifications</th>
<th>ZIGBEE</th>
<th>Bluetooth</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i) Modulation Technique</td>
<td>BPSK</td>
<td>GFSK</td>
</tr>
<tr>
<td>(ii) Communication range</td>
<td>10-100 meters</td>
<td>10-100 meters</td>
</tr>
<tr>
<td>(iii) Power consumption</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>(iv) IEEE Standard</td>
<td>Zigbee is a specification for a set of wireless protocols that have been designed for data transfer by low power devices. It is based on an IEEE 802.15.4 standard. Zigbee has been developed by the Zigbee Alliance</td>
<td>802.11 Bluetooth is a wireless technology through which electronic devices communicate with each other. It is used for the creation of personal networks that operate in the 2.4 GHz unlicensed band and with a range of 10 meters.</td>
</tr>
</tbody>
</table>

Q. 4(d) Draw the interfacing diagram of ADC with 89C51 and explain the function of following pins: SOC, EOC and OE.

Ans.: Waveform Generation - Square Wave

Logic: To generate square wave, first data 00H is given to DAC which will give output as 0v and then data FFH is given to DAC which will give output as 5V. Between these two data, a delay is provided. Depending on Delay:
(i) If delay is same in between both data, the square wave is the output with 50% duty cycle, as shown in the figure.
(ii) If delay is not same in between both data, the pulse is the output with more or less than 50% duty cycle, as shown in the figure.

Fig.: Square wave with different Duty Cycle
Interfacing Diagram

![Interfacing Diagram of DAC](image)

Fig.: Interfacing of DAC

C Program for Square with 50% duty cycle

Duty Cycle calculation
- Delay function for 1ms.
- For example if value = 100 then delay will be 100ms
- Duty Cycle calculation
- Ton = 500ms, Toff = 500ms
- \[ \% \text{ DutyCycle} = \frac{\text{Ton} \times 100}{\text{Ton} + \text{Toff}} \]
- \[ = \frac{500 \times 100}{500 + 500} = 50\% \]

C Program

```c
#include<reg51.h>                           //Define 8051/89c51 Registers
void msdelay(unsigned int a);             //Delay function

void main()
{
    while(1)                               //Loop forever
    {
        P 1  =  0FF;                    //Output FFh data on Port 1
        msdelay (500);               //Delay time for 500ms
        P1 = 0 x 00;                     //Output FFh data on Port 1
        msdelay (500);       //Delay time for 500ms
    }
}

void msdelay(unsigned int value) //delay routine for value x 1ms
{
    unsigned int x,y;
    for(x = 0;x<value; x++)
    for(y = 0;y<1275; y++);
}
```

---

- 11 -
Q.5 Attempt any TWO of the following: [12]

Q.5(a) Distinguish between synchronous and asynchronous communication with any four points.

Ans.:

<table>
<thead>
<tr>
<th></th>
<th>Synchronous</th>
<th>Asynchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Same clock pulse is required at</td>
<td>Different clock pulse is required at</td>
</tr>
<tr>
<td></td>
<td>transmitter and receiver</td>
<td>transmitter and receiver</td>
</tr>
<tr>
<td>2.</td>
<td>Used to transfer group of character</td>
<td>Used to transfer one character at a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>time</td>
</tr>
<tr>
<td>3.</td>
<td>Synchronous character is required</td>
<td>Synchronous character is required.</td>
</tr>
<tr>
<td>4.</td>
<td>No start and stop signals are</td>
<td>Start and stop signals are required.</td>
</tr>
<tr>
<td></td>
<td>required</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>Data transmission rate is greater</td>
<td>Data transmission rate is less than</td>
</tr>
<tr>
<td></td>
<td>or equal to 20Kbps</td>
<td>or equal to 20 Kbps.</td>
</tr>
<tr>
<td>6.</td>
<td>It is less reliable</td>
<td>It is more reliable</td>
</tr>
</tbody>
</table>

Q.5(b) State number of port lines required for a keyboard matrix having following keys: [6]

(i) 16    (ii) 256    (iii) 64    (iv) 144

Ans.:

- Number of port lines required for a keyboard matrix having following keys:
  - (i) 16 - 16 keys can be arranged in a matrix of 4X4, so number of port lines required are 4 + 4 = 8
  - (ii) 256 - 256 keys can be arranged in a matrix of 16X16, so number of port lines required are 16 + 16 =32
  - (iii) 64 - 64 keys can be arranged in a matrix of 8X8, so number of port lines required are 8 + 8 =16
  - (iv) 144 - 144 keys can be arranged in a matrix of 12X12, so number of port lines required are 12 + 12 = 24

Q.5(c) Explain pre-emptive and round robin scheduling in RTOS. [6]

Ans.:

- Pre-emptive Scheduling:
  - This type of scheduling is used when system response is important.
  - The highest priority task ready to run is always given control of the CPU.
  - When a task makes a higher priority task ready to run, the current task is preempted and the higher priority task is immediately given control of the CPU.

- Consider two tasks one low priority task and one high priority task. Initially, the low priority task is running as the high priority task is in waiting state for an external event to occur. After some time, the external event occurs and the high priority task now move to Ready to Run State. Through an interrupt, the ISR is executed to move the high priority task from waiting state to Ready to Run State. Then another ISR is executed to put the high priority task in the Running State. Again after some time, the high priority task may release the CPU and then the low priority task is executed.
- Advantage of preemptive scheduling is that the response is optimum and deterministic.
Round-robin with priority:
- The round-robin algorithm can be slightly modified by assigning priority levels to some or all the tasks. A high priority task can interrupt the CPU so that it can be executed.
- This scheduling algorithm can meet the desired response time for a high priority task. For example, in a bar code scanner, high priority is assigned to the scanning operation.
- The CPU can execute this task by suspending the task that displays the item/price value.
- Soft real-time systems can use this algorithm.

Q.5(d) Write a C language program to generate square waveform of 5KHz on pin P1.5 of 89C51. [6]
Ans.: C Program for Square with 50% duty cycle

Duty Cycle calculation
- Delay function for 1ms.
- For example if value = 100 then delay will be 100ms
- Duty Cycle calculation
- Ton = 500ms, Toff = 500ms
- \[%\text{DutyCycle} = \frac{\text{Ton} \times 100}{(\text{Ton} + \text{Toff})} \]

\[ \frac{500 \times 100}{(500 + 500)} = 50\% \]

C Program
```c
#include<reg51.h>                           //Define 8051/89c51 Registers
void msdelay(unsigned int a);            //Delay function

//------------------main Program

void main()
{
    while(1)                             //Loop forever
    {
        P1 = 0xFF;                   //Output FFh data on Port 1
        msdelay (500);               //Delay time for 500ms
        P1 = 0x00;                     //Output FFh data on Port 1
        msdelay (500);     //Delay time for 500ms
    }
}

void msdelay(unsigned int value) //delay routine for value x 1ms
{
    unsigned int x,y;
    for(x = 0;x<value; x++)
    for(y = 0;y<1275; y++);
}
```

Q.6 Attempt any TWO of the following: [12]
Q.6(a) Draw pin diagram of DB9 connector, stating function of each pin. [6]
Ans.: Pin Diagram:
### Abbreviation | Full Name | Function |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Transmit Data</td>
<td>Serial Data Output (TXD)</td>
</tr>
<tr>
<td>RD</td>
<td>Receive Data</td>
<td>Serial Data Input (RXD)</td>
</tr>
<tr>
<td>CTS</td>
<td>Clear to Send Data</td>
<td>This line indicates that the Modem is ready to exchange data.</td>
</tr>
<tr>
<td>DCD</td>
<td>Data Carrier Detect</td>
<td>When line modem detects a &quot;Carrier&quot; from the modem at the other end of the phone line, this line becomes active.</td>
</tr>
<tr>
<td>DSR</td>
<td>Data Set Ready</td>
<td>This tells the UART dial the modem is ready to establish a link.</td>
</tr>
<tr>
<td>DTR</td>
<td>Data Terminal Ready</td>
<td>This is the opposite to DSR. This tells the Modem that the UART is ready to link.</td>
</tr>
<tr>
<td>RTS</td>
<td>Request To Send</td>
<td>This line informs the Modem that the UART is ready to exchange data.</td>
</tr>
<tr>
<td>RI</td>
<td>Ring Indicator</td>
<td>Goes active when modem detects a ringing signal from the PSTN.</td>
</tr>
</tbody>
</table>

**Q.6(b)** Manipulate the following table for data types used in 'C' language.

<table>
<thead>
<tr>
<th>Data type</th>
<th>Bit size</th>
<th>Data range</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Unsigned char</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>(2) Signed int</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>(3) Sbit</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>(4) Sfr</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

**Ans.:**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Bit size</th>
<th>Data range</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Unsigned char</td>
<td>8</td>
<td>0 to 255</td>
</tr>
<tr>
<td>(2) Signed int</td>
<td>16</td>
<td>-32768 to +32767</td>
</tr>
<tr>
<td>(3) Sbit</td>
<td>1</td>
<td>SFR bit addressable</td>
</tr>
<tr>
<td>(4) Sfr</td>
<td>8</td>
<td>RAM addresses 80 to FF only</td>
</tr>
</tbody>
</table>

**Q.6(c)** Explain in detail the term deadlock and techniques to prevent it.

**Ans.:** 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 reverse 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.

![Deadlock Diagram]

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

**Methods of avoid Deadlock:**
- Mutual exclusion
- Hold & wait or resource holding
- No preemption
- Circular wait

**Starvation:**
- Multiple shared resources have multiple semaphores associated with them.
- The semaphores are all independent of one another. If one task takes semaphore X then another task can take semaphore Y, without blocking.
- But this property can lead to deadlock.
- For example, suppose task 1 calls function to take semaphore X & get it but before it can call the function to take semaphore Y the RTOS stops the execution of task 1 & runs task 2.
  - The task 2 calls the function to take semaphore & gets it.
  - But when task 2 calls the function to take the semaphore X, it is blocked since, the task 1 has already taken the semaphore X.
  - The RTOS now switch back to task 1, which now calls the function to take semaphore Y. Since task 2 has semaphore Y, task 1 is also now blocked.
  - There is no escape from this for either tasks, since now both are blocked, waiting for semaphores that the other has.
  - This problem due to multi-tasking is called Starvation, where a task is denied necessary resources repeatedly without those resources the task will never be completed.

**Q.6(d) Explain in detail any six characteristics of Embedded System.**

**Ans.:** The common critical features and design requirements of an embedded hardware include:
- **Processing power**
  Selection of the processor is based on the amount of processing power to get the job done and also the basis of register width required.
- **Throughput**
  The system may need to handle a lot of data in a short period of time.
- **Response**
  The system has to react to events quickly.
- **Memory**
  Hardware designer must make his best estimate of the memory requirement and must make provision for expansion.
• **Power consumption**
  Systems generally work on battery and design of both software and hardware must take care of power saving techniques.

• **Number of units**
  The no. of units expected to be produced and sold will dictate the Trade-off between production cost and development cost.

• **Expected lifetime**
  Design decisions like selection of components to system development cost will depend on how long the system is expected to run.

• **Program Installation**
  Installation of the software on to the embedded system needs special tools.

• **Testability and Debug ability**
  Setting up test conditions and equipment will be difficult and finding out what is wrong with the software will become a difficult task without a keyboard and the usual display screen.

• **Reliability**
  It is critical if it is a space shuttle or a car but in case of a toy it doesn't always have to work right.