1. Write an assembly language program to transfer "HELLO WORLD" serially at 9600 baud, 8-bit data, 1 stop bit, do this continuously.

```
        MOV TMOD,#20H   ;timer 1,mode 2(auto reload), TMOD = 0010 0000
        MOV TH1,#-3     ;TH1=FDH, 9600 baud rate
        MOV SCON,#50H   ;8-bit, 1 stop, REN enabled
        SETB TR1        ;start timer 1
AGAIN: MOV A,#'H'       ;Load ASCII value of "H" in Accumulator
        ACALL TRANS     ;call transmit subroutine
        MOV A,#'E'      ;transfer "E"
        ACALL TRANS     ;call transmit subroutine
        MOV A,#'L'      ;transfer "L"
        ACALL TRANS     ; call transmit subroutine
        MOV A,#'L'      ;transfer "L"
        ACALL TRANS     ; call transmit subroutine
        MOV A,#'O'      ;transfer "O"
        ACALL TRANS     ; call transmit subroutine
        MOV A,#' '      ;Load ASCII value of space ' '
        ACALL TRANS     ;call transmit subroutine
        MOV A,#'W'      ;transfer "W"
        ACALL TRANS     ; call transmit subroutine
        MOV A,#'O'      ;transfer "O"
        ACALL TRANS     ; call transmit subroutine
        MOV A,#'R'      ;transfer "R"
        ACALL TRANS     ; call transmit subroutine
        MOV A,#'L'      ;transfer "L"
        ACALL TRANS     ; call transmit subroutine
        MOV A,#'D'      ;transfer "D"
        ACALL TRANS     ; call transmit subroutine
        MOV A,#' '      ;Load ASCII value of space ' '
        ACALL TRANS     ;call transmit subroutine

        SJMP AGAIN      ;keep doing it
                        ;serial data transfer subroutine
TRANS: MOV SBUF,A       ;load ASCII value of character into SBUF
HERE: JNB TI,HERE       ;wait for the TI=1 (set)
        CLR TI
        RET
        END
```

2.a Explain full duplex, half duplex and simplex serial data transfer.

Simplex, Half Duplex, and Full Duplex are the mode of transmission in which the data send and receive. In this communication, there is one sender and one receiver to communicate or you can say there is one source and destination to send and receive the data.

1. **Simplex Mode** –
In Simplex mode, the data communication is unidirectional, Only one of the two devices on a link can transmit, the other can only receive.
Example: Keyboard and traditional monitors. The keyboard can only introduce input, the monitor can only give the output.

2. Half-Duplex Mode –
In half-duplex mode, each data can be transmitted and received between two devices (stations), but not at the same time. When one device is sending, the other can only receive, and vice versa. The half-duplex mode is used in cases where there is no need for communication in both directions at the same time.
Example: Walkie-talkie in which message is sent one at a time and messages are sent in both directions.

2. **Full-Duplex Mode** –
In full-duplex mode, both devices can transmit and receive simultaneously.
Example: Telephone Network in which there is communication between two persons by a telephone line, through which both can talk and listen at the same time.

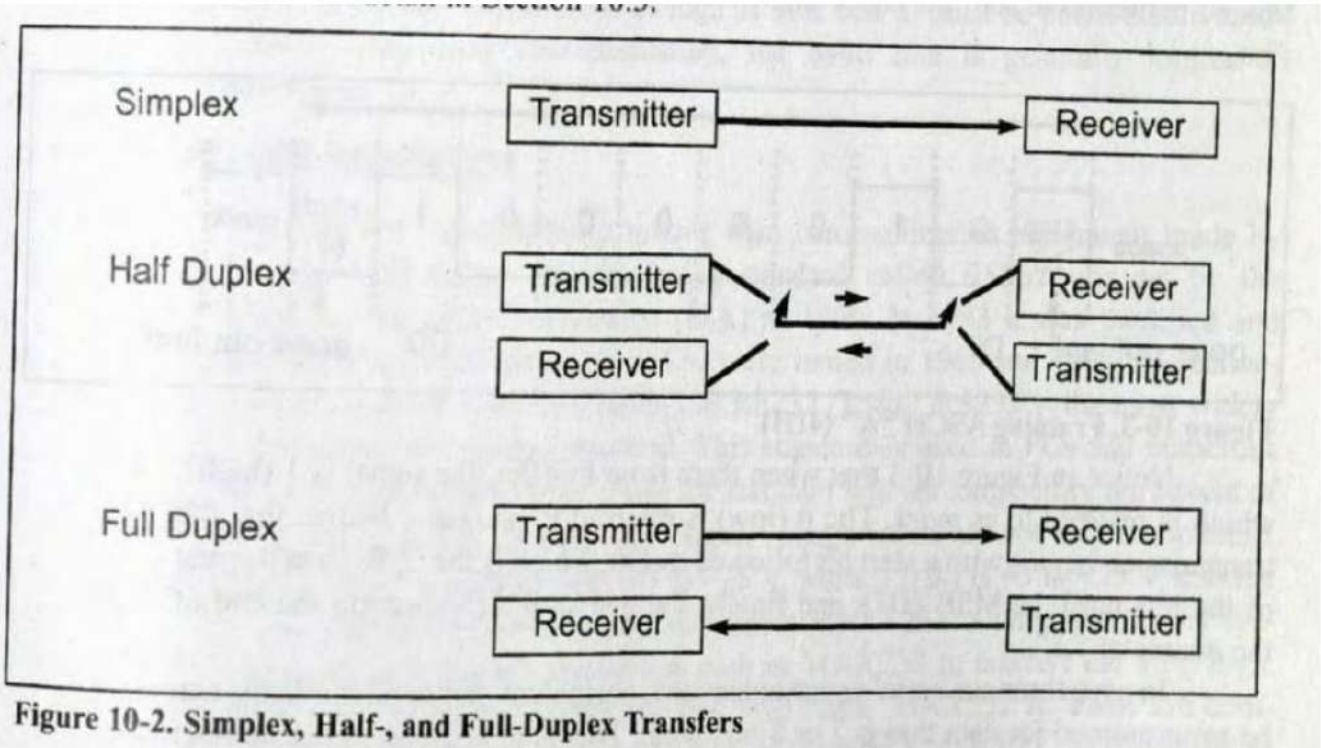


Figure 10-2. Simplex, Half-, and Full-Duplex Transfers

2.b Develop an 8051 C program to toggle the bits of Port 1 to turn OFF and ON LED connected to the port 1.

Solution:

```
#include<reg51.h>
void main(void)
{
        unsigned int x;
        while(1)
        {
                P1=0x55;                    //P1=55H  or 00H
                for(x=0;x<50000;x++);       //Undefined delay
                P1=0xAA;                    //P1=AAH  or FFH
                for(x=0;x<50000;x++);       //Undefined delay
        }
}
```

3. Explain the bit contents of TCON and SCON registers in detail.

TCON (Timer/Counter) Register (Bit-addressable)

D7							D0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1	TCON.7	Timer 1 overflow flag. Set by hardware when timer/counter 1 overflows. Cleared by hardware as the processor vectors to the interrupt service routine
TR1	TCON.6	Timer 1 run control bit. Set/cleared by software to turn timer/counter 1 on/off
TF0	TCON.5	Timer 0 overflow flag. Set by hardware when timer/counter 0 overflows. Cleared by hardware as the processor vectors to the interrupt service routine
TR0	TCON.4	Timer 0 run control bit. Set/cleared by software to turn timer/counter 0 on/off

D7							D0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

IE1	**TCON.3**	**External interrupt 1 edge flag. Set by CPU when the external interrupt edge (H-to-L transition) is detected. Cleared by CPU when the interrupt is processed**
IT1	**TCON.2**	**Interrupt 1 type control bit. Set/cleared by software to specify falling edge/low-level triggered external interrupt**
IE0	**TCON.1**	**External interrupt 0 edge flag. Set by CPU when the external interrupt edge (H-to-L transition) is detected. Cleared by CPU when the interrupt is processed**
IT0	**TCON.0**	**Interrupt 0 type control bit. Set/cleared by software to specify falling edge/low-level triggered external interrupt**

SCON Register:

- SCON is an 8-bit register used to program the start bit, stop bit, and data bits of data framing, among other things

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0	SCON.7	Serial port mode specifier
SM1	SCON.6	Serial port mode specifier
SM2	SCON.5	Used for multiprocessor communication
REN	SCON.4	Set/cleared by software to enable/disable reception
TB8	SCON.3	Transmitted bit 8. Set/cleared by program in mode 2 or 3
RB8	SCON.2	Received bit 8. Bit 8 of received data in modes 2 and 3; stop bit in mode 1. Not used in mode 0.
TI	SCON.1	Transmit interrupt flag. Set to one at the beginning of stop bit in mode 1
RI	SCON.0	Receive interrupt flag. Set to one at halfway through stop bit in mode 1

- SM0, SM1
  - They determine the framing of data by specifying the number of bits per character, and the start and stop bits

SM0	SM1	
0	0	Serial Mode 0
0	**1**	**Serial Mode 1, 8-bit data, 1 stop bit, 1 start bit**
1	0	Serial Mode 2
1	1	Serial Mode 3

Only mode 1 is of interest to us

- SM2
  - This enables the multiprocessing capability of the 8051

- REN (receive enable)
  - It is a bit-adressable register
    - When it is high, it allows 8051 to receive data on RxD pin
    - If low, the receiver is disable
- TI (transmit interrupt)
  - When 8051 finishes the transfer of 8-bit character
    - It raises TI flag to indicate that it is ready to transfer another byte
    - TI bit is raised at the beginning of the stop bit
- RI (receive interrupt)
  - When 8051 receives data serially via RxD, it gets rid of the start and stop bits and places the byte in SBUF register
    - It raises the RI flag bit to indicate that a byte has been received and should be picked up before it is lost
    - RI is raised halfway through the stop bit

4. Assume XTAL = 11.0592 MHz. Use timer 0 interrupt to create the square wave. Write a C program that continuously get single bit of data from P1.7 and sends it to P1.0 while simultaneously creating a square wave of 200 μs period on pin 2.5

Solution:
We will use timer 0 in mode 2 (auto-reload). One half of the period is 100 μs.
100/1.085 μs = 92, and TH0 = 256 - 92 = 164 or A4H.

```
#include<reg51.h>
sbit SW = P1^7;              //define single bit using sbit data type and
sbit LED = P1^0;             //sbit should be declared out of main program (globally)
sbit WAVE = P2^5;

void timer0(void) interrupt 1
{
        WAVE=~WAVE;          //toggle pin
}
void main()
{
    SW=1;                    // Make switch (P1.4) as input
    TMOD=0x02;               //Configure Timer 0 in Mode 2 (8-bit auto reload mode)
    TH0=0xA4;                //Load the initial Count value in TH0
    IE=0x82;                 //Enable the Timer0 Interrupt
    TR0=1;                   //Start Timer 0
    while(1)                 //Repeat Continuously
    {
       LED = SW;             // Read switch and send to LED
    }
}
```

5a. Explain the following, i. Interrupt ii. ISR iii. IVT (Interrupt Vector Table)
List the types of interrupts and their vector locations in 8051.

Solution:

i. **Interrupt:**

An interrupt is an external or internal event that interrupts the microcontroller to inform it that a device needs its service.

Whenever any device needs its service, the device notifies the microcontroller by sending it an interrupt signal. Upon receiving an interrupt signal, the microcontroller interrupts whatever it is doing and serves the device.

ii. ISR:

The program which is associated with the interrupt is called the interrupt service routine (ISR) or interrupt handler.

iii. IVT (Interrupt Vector Table) **:**

When an interrupt is invoked, the microcontroller runs the interrupt service routine. For every interrupt, there is a fixed location in memory that holds the address of its ISR.

The group of memory locations set aside to hold the addresses of ISRs is called interrupt vector table (IVT).

8051 interrupts and their vector locations are listed in the table below.

Interrupt Vector Table for the 8051

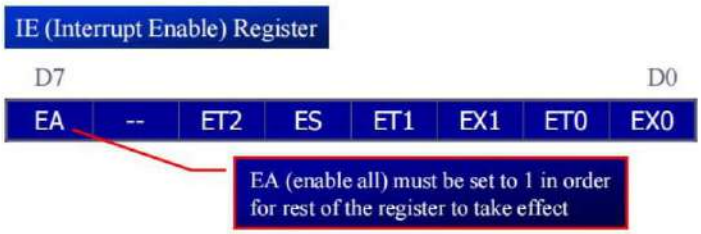
Interrupt	ROM Location (Hex)	Pin	Flag Clearing
Reset	0000	9	Auto
External hardware interrupt 0 (INT0)	0003	P3.2 (12)	Auto
Timer 0 interrupt (TF0)	000B		Auto
External hardware interrupt 1 (INT1)	0013	P3.3 (13)	Auto
Timer 1 interrupt (TF1)	001B		Auto
Serial COM interrupt (RI and TI)	0023		Programmer clears it.

5b. Write the instructions to,

i. enable the serial interrupt, timer 0 interrupt and external hardware interrupt 1,

ii. disable the timer 0 interrupt,

iii. disable all interrupts with a single instruction.



EA	IE.7	Disables all interrupts
--	IE.6	Not implemented, reserved for future use
ET2	IE.5	Enables or disables timer 2 overflow or capture interrupt (8952)
ES	IE.4	Enables or disables the serial port interrupt
ET1	IE.3	Enables or disables timer 1 overflow interrupt
EX1	IE.2	Enables or disables external interrupt 1
ET0	IE.1	Enables or disables timer 0 overflow interrupt
EX0	IE.0	Enables or disables external interrupt 0

Solution:

i. enable the serial interrupt, timer 0 interrupt and external hardware interrupt 1:

```
SETB IE.7     ;Enable all interrupts
SETB IE.4     ; enable the serial interrupt
SETB IE.1     ; enable timer 0 interrupt
SETB IE.2     ; enable external hardware interrupt 1
```

Or, You can write

```
MOV IE,#10010110B;
```

ii. disable the timer 0 interrupt,

```
CLR IE1
```

iii. disable all interrupts with a single instruction.

```
CLR IE.7
```

Or

```
MOV IE,#00H
```

6. a Explain the importance of TI and RI flag in serial communication. Write the programming steps for serial transmission and reception.

Importance of the TI flag

To understand the importance of the role of TI, look at the following sequence of steps that the 8051 goes through in transmitting a character via TxD.

1. The byte character to be transmitted is written into the SBUF register.
2. The start bit is transferred.
3. The 8-bit character is transferred one bit at a time.
4. The stop bit is transferred. It is during the transfer of the stop bit that the 8051 raises the TI flag (TI = 1), indicating that the last character was transmitted and it is ready to transfer the next character.
5. By monitoring the TI flag, we make sure that we are not overloading the SBUF register. If we write another byte into the SBUF register before TI is raised, the untransmitted portion of the previous byte will be lost. In other words, when the 8051 finishes transferring a byte, it raises the TI flag to indicate it is ready for the next character.
6. After SBUF is loaded with a new byte, the TI flag bit must be forced to 0 by the "`CLR TI`" instruction in order for this new byte to be transferred.

Importance of the RI flag bit

In receiving bits via its RxD pin, the 8051 goes through the following steps.

1. It receives the start bit indicating that the next bit is the first bit of the character byte it is about to receive.
2. The 8-bit character is received one bit at time. When the last bit is received, a byte is formed and placed in SBUF.
3. The stop bit is received. When receiving the stop bit the 8051 makes RI = 1, indicating that an entire character byte has been received and must be picked up before it gets overwritten by an incoming character.
4. By checking the RI flag bit when it is raised, we know that a character has been received and is sitting in the SBUF register. We copy the SBUF contents to a safe place in some other register or memory before it is lost.
5. After the SBUF contents are copied into a safe place, the RI flag bit must be forced to 0 by the "`CLR RI`" instruction in order to allow the next received character byte to be placed in SBUF. Failure to do this causes loss of the received character.

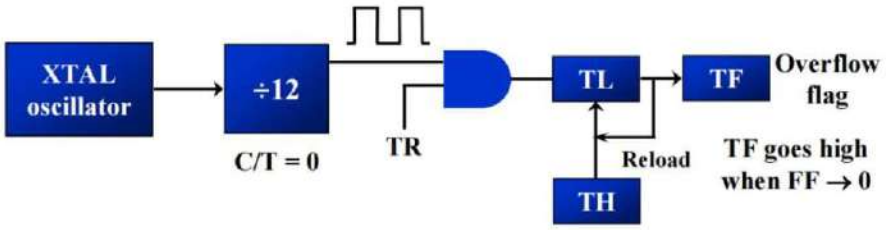
6.B Write the programming steps to program timer 1 in Mode 2.

Solution:

1. Load the TMOD register with 20H to use the timer 1 in mode 2.
2. Load the TH1 registers with the initial count value. The 8051 gives a copy of it to TL1.
3. Start timer using SETB TR1

   After the timer is started, it starts to count up by incrementing the TL1 register. It counts up until it reaches its limit of FFH, When it rolls over from FFH to 00H, it sets high the TF1 (timer 1 flag).

4. Keep monitoring the timer flag (TF1) with the **JNB TF1, target** instruction to see whether it is raised. Get out of the loop when TF1 goes high.
5. Once TL1 register rolls from FFH to 00H and TF is set to 1, TL1 is reloaded automatically with the original value kept by the TH1 register.
6. To repeat the process, clear TF1 and Go back to Step 4, since mode 2 is auto reload.



7a. Write a C program for the 8051 to transfer "YES" serially at 4800 bauds, 8-bit data, 1 stop bit, do this continuously.

```
#include<reg51.h>
void main(void)
{
        TMOD=0x20;                      // TIMER 1 IN MODE 2
        TH1=0xFA;                       //4800 BAUD RATE
        SCON=0X50;                      //8-bit, 1 stop, REN enabled
        TR1=1;                          //Start Timer 1
        while(1)                        //Repeat Continuously
        {
                SBUF='Y';               //Load ASCII value of Y in SBUF
                while(TI==0);           //wait for the TI=1 (set)
                TI=0;                   //Clear TI flag

                SBUF='E';               //Load ASCII value of E in SBUF
                while(TI==0);
                TI=0;

                SBUF='S';               //Load ASCII value of S in SBUF
                while(TI==0);
                TI=0;
        }
}
```

7b. Explain in detail the structure of IE register.

IE (Interrupt Enable) Register

D7							D0
EA	--	ET2	ES	ET1	EX1	ET0	EX0

EA (enable all) must be set to 1 in order for rest of the register to take effect

EA	IE.7	Disables all interrupts
--	IE.6	Not implemented, reserved for future use
ET2	IE.5	Enables or disables timer 2 overflow or capture interrupt (8952)
ES	IE.4	Enables or disables the serial port interrupt
ET1	IE.3	Enables or disables timer 1 overflow interrupt
EX1	IE.2	Enables or disables external interrupt 1
ET0	IE.1	Enables or disables timer 0 overflow interrupt
EX0	IE.0	Enables or disables external interrupt 0

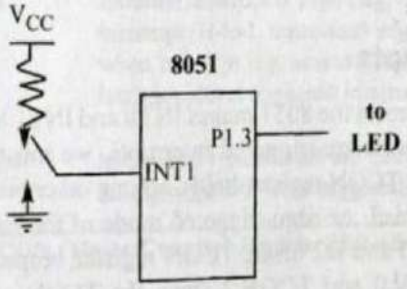
- To enable an interrupt, we take the following steps:
  1. Bit D7 of the IE register (EA) must be set to high to allow the rest of register to take effect
  2. The value of EA
     - If EA = 1, interrupts are enabled and will be responded to if their corresponding bits in IE are high
     - If EA = 0, no interrupt will be responded to, even if the associated bit in the IE register is high

8a. Assume that the INT1 pin is connected to a switch that is normally high. Whenever it goes low, it should turn ON an LED. The LED is connected to P1.3 and is normally OFF. When it is turned on it should stay ON for a fraction of a second. As long as the switch is pressed low, the LED should stay ON.

Solution:

```
          ORG   0000H
          LJMP  MAIN              ;bypass interrupt vector tabl
;--ISR for hardware interrupt INT1 to turn on the LED
          ORG   0013H             ;INT1 ISR
          SETB  P1.3              ;turn on LED
          MOV   R3,#255           ;load counter
BACK:     DJNZ  R3,BACK           ;keep LED on for a while
          CLR   P1.3              ;turn off the LED
          RETI                    ;return from ISR
;--MAIN program for initialization
          ORG   30H
MAIN:     MOV   IE,#10000100B     ;enable external INT1
HERE:     SJMP  HERE              ;stay here until interrupted
          END
```

Pressing the switch will turn the LED on. If it is kept activated, the LED stays on.



8b. What is the use of IP register in 8051 microcontrollers? If interrupts for serial communication, T0 and T1 (Timer0 and Timer1), are activated at the same time and if IP register contains 10H then how the service will be provided to the interrupts.

Solution:

We can alter the sequence of interrupt priority by assigning a higher priority to any one of the interrupts by programming a register called IP (interrupt priority). To give a higher priority to any of the interrupts, we make the corresponding bit in the IP register high.

Interrupt Priority Register (Bit-addressable)

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

--	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

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

If interrupts for serial communication, T0 and T1 (Timer0 and Timer1), are activated at the same time and if IP register contains 10H.

i. e., IP = 00010000B

Here timer 0 and timer 1 have low priority that is '0' in this case interrupts are serviced according to the sequence listed in the below table. That is timer 0 is serviced first and then timer 1 by 8051 microcontroller.

Interrupt Priority Upon Reset

Highest To Lowest Priority	
External Interrupt 0	(INT0)
Timer Interrupt 0	(TF0)
External Interrupt 1	(INT1)
Timer Interrupt 1	(TF1)
Serial Communication	(RI + TI)