Experiment-3

a) **Write a program to demonstrate the operation of UART in 8051 microcontroller by sending and receiving the characters serially.**

Explanation:

First, what does this mean?

1. **UART (Universal Asynchronous Receiver/Transmitter):**
   - It is the hardware inside the 8051 microcontroller that helps in **serial communication** (one bit at a time).
   - Instead of sending 8-bit data (like a whole byte) in parallel, UART sends it bit by bit through one wire → saves pins.

Example: If you want to connect the microcontroller to a PC or another microcontroller, you usually use UART.

2. **Sending and Receiving Characters Serially:**
   - "Sending" means the 8051 will transmit data (like the letter 'A') out through its serial port (TXD pin).
   - "Receiving" means the 8051 will listen on its serial port (RXD pin) and capture whatever character comes in.

So, basically the program will:

- Configure UART → Set baud rate (speed of communication).
- Send a character → Example: 'A'.
- Receive a character → Example: 'B'.
- Store or display it.

Why do we need to write a program?

Because the 8051 **needs to be told**:

- Which baud rate to use (e.g., 9600 bps).
- Which mode of serial communication (8-bit, variable baud rate).
- When to start sending and when to check if a character has been received.

1. Program to transfer data serially with baud rate of 9600

```
ORG 0000H              ; Origin - program starts at address 0000H
MOV TMOD, #20H         ; Timer1 in Mode 2 (8-bit auto-reload mode)
                       ; This mode is used to generate baud rate
MOV TH1, #-3           ; Load TH1 with -3 (FDH).
                       ; This gives baud rate of 9600 when crystal = 11.0592 MHz
MOV SCON, #50H         ; SCON = 0101 0000b
                       ; Mode 1: 8-bit data, variable baud rate
                       ; REN = 1 → Enable Receiver
                       ; SM1=1, SM0=0 → Mode1 selected
SETB TR1               ; Start Timer1 → begins baud rate generation
AGAIN: MOV SBUF, #"A"  ; Move character 'A' into SBUF → this starts
                           transmission
HERE:  JNB TI, HERE    ; Wait until TI (Transmit Interrupt flag) becomes 1
                       ; TI=1 means data is completely transmitted
CLR TI                 ; Clear TI flag manually, ready for next transmission
SJMP AGAIN             ; Repeat the process, send 'A' continuously
END                    ; End of program
```

2. Program to receive data serially with baud rate of 9600

```
ORG 0000H          ; Program start
; ---- UART Initialization ----
MOV TMOD, #20H     ; Timer1 Mode2 (8-bit auto reload) for baud generation
MOV TH1, #0FDH     ; Baud rate = 9600 for 11.0592 MHz crystal
MOV SCON, #50H     ; Serial Control: Mode1 (8-bit UART), REN=1 (Enable RX)
SETB TR1           ; Start Timer1
; ---- Main Program Loop ----
AGAIN: JNB RI, AGAIN  ; Wait until a byte is received (RI=1)
MOV A, SBUF           ; Read received data from serial buffer into Accumulator
MOV P1, A             ; Send ASCII value to Port1 (for LEDs / I/O pins display)
MOV SBUF, A       ; Echo the same data back to Serial Window
```

```
CLR RI          ; Clear Receive flag for next byte
CLR TI          ; Clear Transmit flag (to avoid continuous set condition)
SJMP AGAIN      ; Repeat forever
END
```

How this program works:

Initialize UART (same as TX program)

Timer1 in mode 2, baud rate = 9600.

UART in mode 1 (8-bit, variable baud).

Receiver enabled (REN=1).

Wait for data

The 8051 keeps checking RI (Receive Interrupt flag).

When RI = 1 → it means one full character is received.

Read received data

The character is stored automatically in SBUF (Serial Buffer).

We copy it into A (Accumulator).

Output to Port 1

Whatever character was received is shown on Port 1 pins (P1.0–P1.7).

Example: If you send ASCII 'A' (41h), Port1 outputs 0100 0001.

Clear RI

Must clear RI manually after reading, otherwise UART won't receive next character.

Loop

Keeps receiving forever.

How to Check in Keil

1. Build the program.
2. Go to **View → Serial Windows → UART #1**.
3. Go to **View → Peripherals → I/O Ports → Port1**.
4. Start Debug (Ctrl+F5). Run program (F5).
5. In Serial Window, type A.
    - You will see **A echoed back** in the Serial Window.

- In **Port1 window**, bits will show 0100 0001 (binary of ASCII A).

Try with numbers:

- Type 1 → Serial Window echoes 1 → Port1 = 0011 0001.
- Type B → echoes B → Port1 = 0100 0010.

b) Sequence Generator Using Serial Interface in 8051.

```
ORG 0000H            ; Place program at address 0000H (reset vector)
LJMP START           ; On reset, jump to START (main setup)
; -------------------------
; Program begins
; -------------------------
ORG 8000H                    ; Place the actual program from address 8000H
START:  MOV TMOD,#20H  ; Configure Timer1 in Mode 2 (8-bit auto-reload)
MOV TH1,#0FDH              ; Load TH1 with FDH → generates 9600 baud (with 11.0592
                             MHz crystal)
MOV SCON,#50H  ; Configure serial control: Mode 1 (8-bit UART), REN=1 (receiver
                 enabled)
SETB TR1             ; Start Timer1 (now UART can generate baud rate)
LJMP MAIN          ; Jump to MAIN program
MAIN:   MOV R4,#08     ; Set loop counter = 8 (we want to send 8 characters)
MOV A,#01H         ; Load A = 1 (start from number 1)
NXT_CH: MOV R6,A       ; Save current number into R6 (temporary register)
ADD A,#30H         ; Convert number to ASCII by adding 30H
                   ; e.g., 01H + 30H = 31H → '1'
CJNE A,#3AH,CHECK ; Compare A with 3AH ('9'+1). If not equal, jump to CHECK
SJMP SEND        ; If equal, go directly to SEND
CHECK:  JC SEND        ; If A < 3AH (i.e., '0'–'9'), go to SEND
 ADD A,#07H      ; If A >= 3AH, adjust by +7 to reach ASCII 'A'–'F'
                 ; e.g., 0BH+30H=3BH (';'), +7H → 42H ('B')
```

```
SEND:   LCALL TX_CHAR   ; Call TX_CHAR subroutine to transmit character in A
MOV A,R6        ; Restore the original number from R6
ADD A,#02H      ; Increment number by 2 (next odd number)
DJNZ R4,NXT_CH  ; Decrement R4, if not zero → repeat loop
SJMP $          ; Infinite loop (program ends here safely)
; --------------------------
; Subroutine: TX_CHAR
; Transmits the character in A via UART
; --------------------------
TX_CHAR: MOV SBUF,A     ; Copy A into SBUF (serial buffer register) → starts
                            transmission
WAIT:    JNB TI,WAIT            ; Wait here until TI (Transmit Interrupt flag) becomes 1
                                (done sending)
         CLR TI                 ; Clear TI flag (ready for next transmission)
         RET                    ; Return to main program
END                 ; End of program
```

How it works

- First part sets up UART at **9600 baud**.
- Main loop runs 8 times, each time:
- Convert number → ASCII.
- Send over UART.
- Increment by 2.
- Output will be: 1 3 5 7 9 B D F.