CAT-2 Key **Slot: C1+TC1**

NAME OF THE SCHOOL :School of Electronics Engineering

CONTINUOUS ASSESSMENT TEST - II

FALL SEMESTER 2024-2025

Programme Name & Branch : B.Tech & ECE
Course Code and Course Name :BECE204L Microprocessors and Microcontrollers
Faculty Name(s) :Dr. Sumit Kumar Jindal & Dr. Arun Dev Dhar Dwivedi
Class Number(s) :VL2024250104199, VL2024250104201
Date of Examination :15/10/2024
Exam Duration : 90 minutes Maximum Marks: 50
General instruction(s):
Answer All Questions

M - Max mark; CO – Course Outcome; BL – Blooms Taxonomy Level (1 – Remember, 2 – Understand, 3 – Apply, 4 – Analyse, 5 – Evaluate, 6 – Create)

Course Outcomes

CO4-Deploy the implementation of various peripherals such as general purpose input/

output, timers, serial communication, LCD, keypad and ADC with 8051 microcontroller

Q.1: Assume that two sensors are connected to P3.4 and P3.5 of the 8051 Microcontroller. Sensor connected to P3.4 is placed near entry door of the auditorium and another sensor is connected to P3.5 is placed near the exit door. The maximum capacity of the auditorium is 200. Write an ALP to find the number of people inside the auditorium and send the count to port P0. An LED connected to P1.7 will glow if the auditorium is full.

Ans-1: To monitor the number of people entering and exiting an auditorium with a maximum capacity of 200, and control an LED based on this count, we can develop an Assembly Language Program (ALP) for the 8051 microcontroller. Here's how the program works:

- **P3.4** is connected to the entry sensor.
- **P3.5** is connected to the exit sensor.
- The count of people inside is stored in a register, and it will be updated based on the sensor input.
- The count will be output to **P0**.
- An LED connected to **P1.7** will turn on if the auditorium is full (count = 200).

Here's the ALP code:

```asm
ORG 0000H            ; Origin of program

MOV P0, #00H         ; Initialize count display to 0 on P0
MOV P1, #00H         ; Clear P1 (LED control)
MOV R0, #00H         ; Initialize count register (R0) to 0

; Main program loop
MAIN:
    MOV A, P3        ; Read the current state of Port 3
    JB ACC.4, ENTRY  ; Check if entry sensor (P3.4) is triggered (high)
    JB ACC.5, EXIT   ; Check if exit sensor (P3.5) is triggered (high)
    SJMP MAIN        ; No change, loop back to main
```

```
ENTRY:
    INC R0           ; Increment count for entry
    MOV A, R0        ; Check if count exceeds max capacity
    CJNE A, #200, NOT_FULL
    SETB P1.7        ; Turn on LED (P1.7) if count = 200 (full)
    SJMP DISPLAY

NOT_FULL:
    CLR P1.7         ; Turn off LED if count < 200

DISPLAY:
    MOV P0, R0       ; Update P0 with the current count
WAIT_ENTRY:
    JNB P3.4, MAIN   ; Wait for entry sensor to be released before continuing
    SJMP MAIN
```

```
EXIT:
    MOV A, R0        ; Check if there are people inside to decrement
    JZ MAIN          ; If count is 0, ignore exit signal
    DEC R0           ; Decrement count for exit
    MOV P0, R0       ; Update P0 with the current count
    CLR P1.7         ; Turn off LED (P1.7) since count < 200 after decrement
WAIT_EXIT:
    JNB P3.5, MAIN   ; Wait for exit sensor to be released before continuing
    SJMP MAIN

END
```

Explanation:

1. **Port Setup**:
   - `P0` is used to display the current count of people inside.
   - `P1.7` controls the LED, which turns on when the count reaches 200 (auditorium full).
   - `P3.4` and `P3.5` are used for entry and exit sensors, respectively.
2. **Count Management**:
   - The count of people is stored in register `R0`.
   - When the entry sensor (connected to `P3.4`) is triggered, the count increments by 1.
   - When the exit sensor (connected to `P3.5`) is triggered, the count decrements by 1 (only if there are people inside).
3. **LED Control**:
   - If the count reaches the maximum capacity of 200, the LED connected to `P1.7` turns on.
   - If the count is below 200, the LED remains off.
4. **Sensor Debounce**:
   - The program waits until the entry or exit sensor is released (`JNB P3.4` or `JNB P3.5`) before allowing further changes, ensuring each person is counted only once.

This ALP will continuously monitor the entry and exit sensors, update the count displayed on `P0`, and control the LED based on the auditorium's occupancy status.

Q.2: Two switches (SW1 and SW2) are connected to P1.2 and P1.3 of 8051 microcontroller. Develop an ALP to monitor the status of switches and transmit the following messages with the given baud rate.

S. No.	SW1	SW2	Message to be transmitted	Baud rate
1	OFF	OFF	OFF	1200
2	OFF	ON	SWITCH2	2400
3	ON	OFF	SWITCH1	4800
4	ON	ON	ON	9600

TH1	(Decimal)	(Hex)	SMOD=0	SMOD=1
	-3	FD	9600	19200
	-6	FA	4800	9600
	-12	F4	2400	4800
	-24	E8	1200	2400

S.No	SW1	SW2	Message to be transmitted serially	Baud rate
1	OFF	OFF	OFF	1200
2	OFF	ON	SWITCH2	2400
3	ON	OFF	SWITCH1	4800
4	ON	ON	ON	9600

Ans2:

Here's an Assembly Language Program (ALP) for the 8051 microcontroller to monitor the switches connected to pins P1.2 and P1.3 and transmit the specified messages at the given baud rates using serial communication. For simplicity, I'll use Timer 1 in Mode 2 (8-bit auto-reload) for baud rate generation and a polling mechanism to check switch states.

Assumptions:

1. The crystal oscillator frequency is 11.0592 MHz.
2. The serial port uses Timer 1 for baud rate control in Mode 2.

Code:

```
ORG 0000H ; Start of program
```

```
MAIN:
MOV TMOD, #20H ; Set Timer 1 in Mode 2 (8-bit auto-reload for baud rate)
MOV SCON, #50H ; Configure serial mode 1, 8-bit UART, REN enabled
CHECK_SWITCHES:
MOV A, P1 ; Load P1 to check switch states
JB ACC.2, SW1_ON ; Check if SW1 (P1.2) is ON
JB ACC.3, SW2_ON ; Check if SW2 (P1.3) is ON
SWITCHES_OFF:
; SW1 = OFF, SW2 = OFF
MOV TH1, #-24 ; Set baud rate to 1200
SETB TR1 ; Start Timer 1
MOV DPTR, #OFF_MSG
ACALL SEND_MESSAGE
SJMP CHECK_SWITCHES
SW2_ON:
; SW1 = OFF, SW2 = ON
MOV TH1, #-12 ; Set baud rate to 2400
SETB TR1 ; Start Timer 1
MOV DPTR, #SWITCH2_MSG
ACALL SEND_MESSAGE
SJMP CHECK_SWITCHES
SW1_ON:
JB ACC.3, BOTH_ON ; Check if SW2 is also ON
; SW1 = ON, SW2 = OFF
MOV TH1, #-6 ; Set baud rate to 4800
```

```
SETB TR1 ; Start Timer 1

MOV DPTR, #SWITCH1_MSG

ACALL SEND_MESSAGE

SJMP CHECK_SWITCHES

BOTH_ON:

; SW1 = ON, SW2 = ON

MOV TH1, #-3 ; Set baud rate to 9600

SETB TR1 ; Start Timer 1

MOV DPTR, #ON_MSG

ACALL SEND_MESSAGE

SJMP CHECK_SWITCHES

SEND_MESSAGE:

CLR A

NEXT_CHAR:

MOVC A, @A+DPTR ; Load the character from the message

JZ END_MSG ; If zero, end of message

MOV SBUF, A ; Send character

WAIT_TX:

JNB TI, WAIT_TX ; Wait for transmission to complete

CLR TI ; Clear TI flag

INC DPTR ; Move to next character

SJMP NEXT_CHAR ; Repeat for next character

END_MSG:

RET ; Return to main loop

; Messages
```

```
OFF_MSG: DB 'OFF', 0

SWITCH2_MSG: DB 'SWITCH2', 0

SWITCH1_MSG: DB 'SWITCH1', 0

ON_MSG: DB 'ON', 0

END
```

Explanation

1. **Initial Setup**:
    - `TMOD` is set to `20H` to configure Timer 1 in 8-bit auto-reload mode.
    - `SCON` is set to `50H` to enable Serial Mode 1 (8-bit UART) and enable reception.
2. **Switch Monitoring**:
    - Continuously check `P1.2` and `P1.3` to determine the states of `SW1` and `SW2`.
    - Based on switch conditions, appropriate baud rate is set, and the corresponding message is transmitted.
3. **Baud Rate Calculation**:
    - Baud rates are set using `TH1` based on the oscillator frequency (assumed to be 11.0592 MHz).
4. • Baud rates are configured as follows:
    - 1200: `TH1 = -24`
    - 2400: `TH1 = -12`
    - 4800: `TH1 = -6`
    - 9600: `TH1 = -3`

- **Message Transmission**:

5. The `SEND_MESSAGE` subroutine sends each character in a message until it encounters a zero byte, signaling the end of the message.

Q.3: Identify what the program does and write down the comment lines for the given program

```
ORG 0H
START:
MOV SP, #60H
MOV SCON, #50H
MOV TMOD, #20H
MOV TH1, #0FDH
SETB TR1
MAIN_LOOP:
MOV DPTR, #MESSAGE
MOV R0, #0
SEND_CHAR:
MOVC A, @DPTR
INC DPTR
JNB TI, SEND_CHAR
CLR TI
CJNE A, #00H, SEND_CHAR
SJMP MAIN_LOOP
MESSAGE: DB 'MPMC', 0 ;
END
```

Ans-3:

```
ORG 0H                  ; Set the starting address of the program to 0H

START:
    MOV SP, #60H        ; Set the stack pointer to 60H
    MOV SCON, #50H      ; Configure the serial control register for 8-bit UART mode
    MOV TMOD, #20H      ; Set Timer 1 in Mode 2 (auto-reload mode) for baud rate generati
    MOV TH1, #0FDH      ; Load the timer high register TH1 with 0FDH to set the baud rate
    SETB TR1            ; Start Timer 1 to begin baud rate generation

MAIN_LOOP:
    MOV DPTR, #MESSAGE  ; Load the data pointer register DPTR with the address of the mes
    MOV R0, #0          ; Initialize R0 to 0 (used as an index for characters)

SEND_CHAR:
    MOVC A, @DPTR       ; Move the character at the address in DPTR to accumulator A
    INC DPTR            ; Increment DPTR to point to the next character in MESSAGE
    JNB TI, SEND_CHAR   ; Wait for the transmit interrupt (TI) flag to be set (indicating
    CLR TI              ; Clear the transmit interrupt flag after transmission
    CJNE A, #00H, SEND_CHAR ; Check if the character in A is 0 (indicating the end of the
    SJMP MAIN_LOOP      ; Loop back to MAIN_LOOP to continuously send the message
```

```
MESSAGE:
    DB 'MPMC', 0        ; Define the message "MPMC" with a null terminator (0) to signal

END                     ; End of program
```

ORG 0H ; Set the starting address of the program to 0H

START:

MOV SP, #60H ; Set the stack pointer to 60H

MOV SCON, #50H ; Configure the serial control register for 8-bit UART mode

MOV TMOD, #20H ; Set Timer 1 in Mode 2 (auto-reload mode) for baud rate generation

MOV TH1, #0FDH ; Load the timer high register TH1 with 0FDH to set the baud rate to 9600 bps

SETB TR1 ; Start Timer 1 to begin baud rate generation

MAIN_LOOP:

```
    MOV DPTR, #MESSAGE   ; Load the data pointer register DPTR with the address of the message string
    MOV R0, #0           ; Initialize R0 to 0 (used as an index for characters)

SEND_CHAR:
    MOVC A, @DPTR        ; Move the character at the address in DPTR to accumulator A
    INC DPTR             ; Increment DPTR to point to the next character in MESSAGE
    JNB TI, SEND_CHAR    ; Wait for the transmit interrupt (TI) flag to be set (indicating readiness to send)
    CLR TI               ; Clear the transmit interrupt flag after transmission
    CJNE A, #00H, SEND_CHAR ; Check if the character in A is 0 (indicating the end of the string); if not, continue sending
    SJMP MAIN_LOOP       ; Loop back to MAIN_LOOP to continuously send the message

MESSAGE:
    DB 'MPMC', 0         ; Define the message "MPMC" with a null terminator (0) to signal the end of the message

END                      ; End of program
```

Q.4: You have a gas sensor that outputs analog voltage. Propose a solution for interfacing this sensor with the 8051 using an ADC0804. Develop an ALP outline for reading the ADC values and responding to the gas concentration levels.

Ans-4:

To interface an analog gas sensor with the 8051 microcontroller, we can use the ADC0804, an 8-bit Analog-to-Digital Converter (ADC) with a simple digital output that the microcontroller can read. Here's a solution for setting up the connection and a proposed Assembly Language Program (ALP) outline to read the ADC values and respond to gas concentration levels.

Solution Outline

1. **Connection Setup**:
   - Connect the gas sensor's analog output to the `Vin` pin of the ADC0804.
   - Connect the ADC0804's digital data output pins (`D0-D7`) to Port 1 (`P1.0-P1.7`) of the 8051 to read the 8-bit digital output.
   - Use Port pins on the 8051 (e.g., `P3.2`, `P3.3`, and `P3.4`) to control the ADC0804's `RD` (Read), `WR` (Write), and `INTR` (Interrupt) pins.

2. **Working Principle**:
    - The ADC0804 converts the analog input voltage from the gas sensor into an 8-bit digital value.
    - The ADC is triggered to start a conversion by setting the `WR` pin low, then high.
    - Once the conversion is complete, the `INTR` pin goes low, signaling that the digital output is ready.
    - The microcontroller then reads the digital value from Port 1 to determine the gas concentration.
3. **Gas Concentration Levels**:
    - Based on the ADC reading, the program will define thresholds to indicate different gas concentration levels. For example:
        - **Low concentration**: ADC value < 50
        - **Moderate concentration**: 50 ≤ ADC value < 150
        - **High concentration**: ADC value ≥ 150
    - Responses could involve triggering an alarm, activating a fan, or displaying values on an LCD.

ALP Outline

Here's an ALP outline to interface the ADC0804 with the 8051, read the gas sensor's values, and respond accordingly.

```
; Define Ports
P1_DATA  EQU P1       ; ADC data port
ADC_WR   EQU P3.2     ; ADC0804 Write (WR) pin
ADC_RD   EQU P3.3     ; ADC0804 Read (RD) pin
ADC_INTR EQU P3.4     ; ADC0804 Interrupt (INTR) pin
ALARM    EQU P2.0     ; Alarm pin to indicate high gas concentration
FAN      EQU P2.1     ; Fan pin for ventilation in moderate-high concentration

START:
    CLR ADC_WR        ; Initialize WR as low
    SETB ADC_RD       ; Initialize RD as high
    SETB P2           ; Initialize alarm and fan off
    MOV P1, #00H      ; Clear ADC data port

READ_ADC:
    CLR ADC_WR        ; Start ADC conversion (WR low)
    NOP               ; Small delay
    SETB ADC_WR       ; WR high to complete start of conversion
```

```
WAIT_INTR:
    JB ADC_INTR, WAIT_INTR  ; Wait until INTR goes low (conversion complete)

    CLR ADC_RD          ; Enable ADC data output (RD low)
    MOV A, P1_DATA      ; Read ADC data into accumulator
    SETB ADC_RD         ; Disable ADC data output (RD high)

    ; Check gas concentration levels based on ADC value in accumulator
    MOV R0, A           ; Move ADC value to register R0 for comparison

LOW_LEVEL:
    CJNE R0, #50, MODERATE_LEVEL ; Compare for low concentration
    CLR ALARM           ; Turn off alarm if low concentration
    CLR FAN             ; Turn off fan
    SJMP READ_ADC       ; Loop back to read next value

MODERATE_LEVEL:
    CJNE R0, #150, HIGH_LEVEL ; Compare for moderate concentration
    CLR ALARM           ; No alarm
    SETB FAN            ; Turn on fan for ventilation
    SJMP READ_ADC       ; Loop back to read next value
```

```
HIGH_LEVEL:
     SETB ALARM         ; Activate alarm for high gas concentration
     SETB FAN           ; Turn on fan
     SJMP READ_ADC      ; Loop back to read next value


END
```

Explanation

1. **ADC Initialization and Conversion**:
   - `CLR ADC_WR` and `SETB ADC_WR` initiate the conversion on the ADC0804.
   - The program then waits for `INTR` to go low, signaling the end of conversion.
2. **Reading ADC Value**:
   - The digital output is read from Port 1 (`P1`) into the accumulator (`A`).
   - `SETB ADC_RD` disables the ADC output after reading.
3. **Gas Concentration Levels and Actions**:
   - Based on the ADC value in `R0`, the program compares it to threshold values to determine the concentration level:
     - **Low concentration** (ADC < 50): Turns off both alarm and fan.
     - **Moderate concentration** (50 ≤ ADC < 150): Activates the fan.
     - **High concentration** (ADC ≥ 150): Activates both the fan and alarm.
4. **Repeat**:
   - The program continuously reads new values from the ADC and updates the response based on current gas levels.

This program will help monitor gas concentration levels in real-time and respond with appropriate safety measures based on the thresholds set for each concentration level.

Q.5: Design an assembly language program in 8051 that alters the displayed message on the LCD based on environmental conditions (e.g., displaying "Hot" if the temperature exceeds a threshold) and displays “Cold” if the temperature is below the threshold. Assume threshold temperature to be 30 degrees celsius.

Ans-5:

```
ORG 0000H            ; Origin of the program

; Define Pins and Ports
LCD_CMD  EQU P2      ; LCD control port (RS, RW, EN)
LCD_DATA EQU P0      ; LCD data port (D4-D7)
ADC_DATA EQU P1      ; ADC data port (D0-D7)
ADC_WR   EQU P3.2    ; ADC0804 Write (WR) pin
ADC_RD   EQU P3.3    ; ADC0804 Read (RD) pin
ADC_INTR EQU P3.4    ; ADC0804 Interrupt (INTR) pin

THRESHOLD EQU 0x9BH  ; Threshold ADC value (155 in hex for 30°C)
```

```
; LCD Control Pins
RS EQU 0             ; Register Select (RS) for LCD
RW EQU 1             ; Read/Write (RW) for LCD
EN EQU 2             ; Enable (EN) for LCD

START:
    ; Initialize LCD
    ACALL LCD_INIT
    MOV P1, #00H     ; Clear ADC data port

READ_ADC:
    ; Start ADC Conversion
    CLR ADC_WR       ; Start ADC conversion (WR low)
    NOP              ; Small delay
    SETB ADC_WR      ; WR high to complete start of conversion
```

```
WAIT_INTR:
    JB ADC_INTR, WAIT_INTR  ; Wait until INTR goes low (conversion complete)

    ; Read ADC Value
    CLR ADC_RD       ; Enable ADC data output (RD low)
    MOV A, ADC_DATA  ; Read ADC data into accumulator
    SETB ADC_RD      ; Disable ADC data output (RD high)

    ; Check temperature threshold
    MOV R0, A        ; Move ADC value to register R0
    CJNE R0, THRESHOLD, DISPLAY_COLD  ; Compare R0 with threshold
    JC DISPLAY_COLD  ; Jump if ADC value < threshold (Cold)
```

```
DISPLAY_HOT:
    ACALL CLEAR_LCD  ; Clear LCD display
    MOV DPTR, #HOT_MSG
    ACALL DISPLAY_MSG
    SJMP READ_ADC    ; Repeat reading and checking

DISPLAY_COLD:
    ACALL CLEAR_LCD  ; Clear LCD display
    MOV DPTR, #COLD_MSG
    ACALL DISPLAY_MSG
    SJMP READ_ADC    ; Repeat reading and checking
```

```
; LCD Initialization and Message Display Routines
LCD_INIT:
    ; LCD 4-bit mode initialization sequence
    MOV A, #38H      ; Function set (8-bit interface)
    ACALL LCD_CMD_WR
    MOV A, #0EH      ; Display ON, cursor ON
    ACALL LCD_CMD_WR
    MOV A, #01H      ; Clear Display
    ACALL LCD_CMD_WR
    MOV A, #06H      ; Entry Mode set (increment cursor)
    ACALL LCD_CMD_WR
    RET
```

```
LCD_CMD_WR:
    MOV LCD_CMD, A   ; Send command to LCD
    CLR LCD_CMD.RS   ; RS = 0 for command
    CLR LCD_CMD.RW   ; RW = 0 for write
    SETB LCD_CMD.EN  ; Enable pulse
    NOP
    CLR LCD_CMD.EN
    RET
```

```
LCD_DATA_WR:
    MOV LCD_DATA, A  ; Send data to LCD
    SETB LCD_CMD.RS  ; RS = 1 for data
    CLR LCD_CMD.RW   ; RW = 0 for write
    SETB LCD_CMD.EN  ; Enable pulse
    NOP
    CLR LCD_CMD.EN
    RET
```

```
CLEAR_LCD:
    MOV A, #01H      ; Clear display command
    ACALL LCD_CMD_WR
    RET

DISPLAY_MSG:
    ; Display message pointed to by DPTR
DISP_LOOP:
    CLR A
    MOVC A, @A+DPTR  ; Load character from message string
    JZ DISP_END      ; Stop if null terminator is reached
    ACALL LCD_DATA_WR
    INC DPTR         ; Move to next character
    SJMP DISP_LOOP   ; Continue displaying
DISP_END:
    RET
```

```
; Messages
HOT_MSG: DB 'HOT', 0
COLD_MSG: DB 'COLD', 0

END
```

Explanation

1. **Initialization**:
   - `LCD_INIT` sets up the LCD in 4-bit mode and configures it for displaying messages.
2. **ADC Conversion**:
   - The program initiates ADC0804 conversion by toggling `WR` and waits until `INTR` goes low, signaling the end of the conversion.
   - The digital value from the ADC is read from `ADC_DATA` (Port 1) and stored in `A`.
3. **Temperature Comparison**:
   - The ADC value is compared with `THRESHOLD` (155).
   - If the ADC value is greater than or equal to the threshold, the program displays "Hot."
   - If it's below the threshold, it displays "Cold."
4. **LCD Display**:
   - `DISPLAY_MSG` loads the message from the data pointer `DPTR` and sends each character to the LCD for display.
   - The messages are stored in `HOT_MSG` and `COLD_MSG`, and the display ends when a null terminator (`0`) is reached.

This program will continuously monitor temperature readings and update the LCD display based on the temperature conditions relative to the threshold.