REG.NO.:

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF ELECTRONICS ENGINEERING
CONTINUOUS ASSESSMENT TEST - II
FALL SEMESTER 2024-2025

SLOT: G1+TG1

Programme Name & Branch : **B. Tech & CSE**
Course Code and Course Name : **Microprocessor & Microcontrollers (BECE204L)**

Faculty Name(s) : **Debashish Dash, A D D Dwivedi, G Ragunath, Kalyanbrata Ghosh, Vikas Vijayavargiya, Henridass, Abhishek Tripathi, Naushad H Laskar, Vishal Gupta**

Class Number(s) : **VL2024250104321, VL2024250104315, VL2024250104323, VL2024250104343, VL2024250104331, VL2024250104318, VL2024250104334, VL2024250104328, VL2024250104326**

Date of Examination : **19-Oct-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)
- Assume missing data wherever applicable.

Q. No	Question	M	CO	BL
1.	Write an assembly language program for the 8051 to generate a square wave of frequency 100 Hz on pin P1.0 by using timer 0 mode 2. Assume that XTAL = 11.0592MHz.	10	CO4	BL5
2.	Write an 8051-assembly level program to transmit a sequence of 4 characters ('A', 'B', 'C', 'D') serially. After completing this transmission double the baud rate and then transmit another sequence of 4 characters ('E', 'F', 'G', 'H'). Toggle an output pin P1.0 every time a character is successfully transmitted. **(Crystal frequency of microcontroller is 11.0592 MHz)**	10	CO4	BL5
3.	Write an 8051-assembly program using Timer 0 to generate an interrupt every 120 micro second and External Interrupt 1 (INT1) to trigger on a falling edge. Upon each Timer 0 interrupt, increment a counter and display the counter value on Port 2. When External INT1 occurs, immediately reset the counter and turn ON an LED connected to P1.7. The LED should remain ON for some time after the interrupt. To ensure proper functionality, configure the system such that Timer 0 interrupts do not interfere with the handling of External INT1, which should have the highest priority. **(Crystal frequency of microcontroller is 11.0592 MHz)**	10	CO4	BL5

REG.NO.:

VIT
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF ELECTRONICS ENGINEERING
CONTINUOUS ASSESSMENT TEST - II
FALL SEMESTER 2024-2025

SLOT: G1+TG1

4.	(a) In the following program identify the errors (if any) and re-write the correct instruction. If no error found, write "correct" against the instruction. **(i) ADD AX, 1234H** **(ii) SUB BL, AX** **(iii) MOV DS, 5060H** **(iv) PUSH [BP+04H]** **(v) CMP CX, 5000D**	5	CO2	BL4
	(b) If CS = 3500H and DS = 4000H. Find if there is any overlap between these two segments. If so, what is the size of the overlap?	5		
5.	Write an **8086-assembly language program** to count the number of 1's (set bits) in an 8-bit binary number stored at memory location **2000H**. Store the result (count of 1's) at memory location **3000H**.	10	CO2	BL4

****

Write an assembly language program for the 8051 to generate a square wave of frequency 100 Hz on pin P1.0 by using timer 0 mode 2. Assume that XTAL = 11.0592MHz

Solution :

- The 8051 clock is divided by 12 internally, so the timer clock frequency is:

$$\text{Timer Clock Frequency} = \frac{11.0592\,\text{MHz}}{12} = 921.6\,\text{kHz}$$

- The period of a 100 Hz square wave is:

$$\text{Period} = \frac{1}{100} = 10\,\text{ms}$$

- Since we need to toggle the pin twice per cycle (once high, once low), the half-period is:

$$\text{Half-Period} = \frac{10\,\text{ms}}{2} = 5\,\text{ms}$$

- To calculate the timer reload value, first find the number of timer ticks in 5 ms:

$$\text{Ticks} = 5\,\text{ms} \times 921.6\,\text{kHz} = 4608\,\text{ticks}$$

- Since the timer overflows after 256 counts, the number of overflows required is:

$$\text{Overflows} = \frac{4608}{256} = 18\,\text{overflows}$$

- Thus, we need the timer to count from `(256 - remainder)`, where `remainder` is:

$$\text{Remainder} = 4608 \bmod 256 = 0$$

- So, we will set `TH0` to `0`.

```asm
    ORG 0000H          ; Start from address 0
    MOV TMOD, #02H     ; Set Timer 0 to Mode 2 (8-bit auto-reload)
    MOV TH0, #0        ; Load TH0 with 0 (initial value)
    MOV TL0, #0        ; Load TL0 with 0
    SETB TR0           ; Start Timer 0
    MOV R1, #18        ; 18 overflows for 5 ms delay

TOGGLE_PIN:
    CLR P1.0           ; Clear P1.0 (Make it LOW)
WAIT_LOW:
    JNB TF0, WAIT_LOW ; Wait until Timer 0 overflows
    CLR TF0            ; Clear Timer 0 overflow flag
    DJNZ R1, WAIT_LOW ; Decrement R1 and check for 18 overflows

    MOV R1, #18        ; Reload R1 with 18 for next delay
    SETB P1.0          ; Set P1.0 (Make it HIGH)
WAIT_HIGH:
    JNB TF0, WAIT_HIGH ; Wait until Timer 0 overflows
    CLR TF0             ; Clear Timer 0 overflow flag
    DJNZ R1, WAIT_HIGH ; Decrement R1 and check for 18 overflows

    SJMP TOGGLE_PIN    ; Repeat the process
```

Write an 8051-assembly level program to transmit a sequence of 4 characters ('A', 'B', 'C', 'D') serially. After completing this transmission double the baud rate and then transmit another sequence of 4 characters ('E', 'F', 'G', 'H'). Toggle an output pin P1.0. every time a character is successfully transmitted. (Crystal frequency of microcontroller is 11.0592 MHz

```
    ; Initialization
    MOV TMOD, #20H       ; Timer 1 in Mode 2 (8-bit auto-reload)
    MOV TH1, #-3         ; Set baud rate to 9600 (Reload value = 253)
    MOV TL1, #-3         ; Load TL1 with 253 as well
    SETB TR1             ; Start Timer 1
    MOV SCON, #50H       ; Serial mode 1, 8-bit UART, REN enabled
    SETB TI              ; Set transmit flag to indicate ready

    ; Transmit first sequence: 'A', 'B', 'C', 'D'
    MOV DPTR, #MSG1      ; Point to first message
    CALL TRANSMIT        ; Transmit 'A', 'B', 'C', 'D'

    ; Double the baud rate
    SETB PCON.7          ; Set SMOD bit to double the baud rate

    ; Transmit second sequence: 'E', 'F', 'G', 'H'
    MOV DPTR, #MSG2      ; Point to second message
    CALL TRANSMIT        ; Transmit 'E', 'F', 'G', 'H'

    SJMP $               ; Stop program execution (infinite loop)

; Transmit Subroutine
TRANSMIT:
    MOV R0, #04          ; Transmit 4 characters
TRANSMIT_LOOP:
    CLR A                ; Clear accumulator
    MOVC A, @A+DPTR      ; Get the character from the message
    MOV SBUF, A          ; Load character into SBUF for transmission
WAIT_TX:
    JNB TI, WAIT_TX      ; Wait for transmission to complete
    CLR TI               ; Clear the transmit interrupt flag
    CPL P1.0             ; Toggle P1.0 after each transmission
    INC DPTR             ; Increment DPTR to point to the next character
    DJNZ R0, TRANSMIT_LOOP ; Transmit the next character
    RET                  ; Return from the subroutine

MSG1: DB 'A', 'B', 'C', 'D' ; First sequence of characters
MSG2: DB 'E', 'F', 'G', 'H' ; Second sequence of characters

    END
```

Write an 8051-assembly program using Timer 0 to generate an interrupt every 120 micro second and External Interrupt 1 (INT1) to trigger on a falling edge. Upon each Timer 0 interrupt, increment a counter and display the counter value on Port 2. When External INT1 occurs, immediately reset the counter and turn ON an LED connected to P1.7. The LED should remain ON for some time after the interrupt.

To ensure proper functionality, configure the system such that Timer 0 interrupts do not interfere with the handling of External INT1, which should have the highest priority. (Crystal frequency of microcontroller is 11.0592 MHz)

```
ORG 0000H          ; Start of the program

; Interrupt Vector Table
LJMP MAIN          ; Jump to main program

ORG 0003H          ; Timer 0 Interrupt Vector
LJMP TIMER_ISR     ; Jump to Timer ISR

ORG 0013H          ; External Interrupt 1 (INT1) Vector
LJMP INT1_ISR      ; Jump to INT1 ISR
```

```
MAIN:
    ; Initialization
    MOV TMOD, #01H      ; Timer 0 mode 1 (16-bit timer)
    MOV TH0, #0FFH      ; Load TH0 with high byte of reload value (FFH)
    MOV TL0, #091H      ; Load TL0 with low byte of reload value (91H)
    SETB TR0            ; Start Timer 0

    SETB IT1            ; Configure INT1 for falling edge trigger
    SETB EX1            ; Enable external interrupt 1 (INT1)

    MOV IE, #82H        ; Enable Timer 0 and INT1 interrupts
                        ; Enable global interrupts (EA), INT1 (EX1), and Timer 0 (ET0)

    MOV IP, #04H        ; Set INT1 (EX1) to the highest priority

    CLR P1.7            ; Ensure LED is OFF initially

    MOV P2, #00H        ; Clear Port 2 (initialize counter display)

    SJMP $              ; Infinite loop (waiting for interrupts)
```

```
; Timer 0 Interrupt Service Routine
TIMER_ISR:
    PUSH ACC           ; Save accumulator
    INC P2             ; Increment the counter and display it on Port 2
    MOV TH0, #0FFH     ; Reload TH0 (high byte of 65425)
    MOV TL0, #091H     ; Reload TL0 (low byte of 65425)
    CLR TF0            ; Clear Timer 0 overflow flag
    POP ACC            ; Restore accumulator
    RETI               ; Return from interrupt

; External Interrupt 1 (INT1) Service Routine
INT1_ISR:
    PUSH ACC           ; Save accumulator
    CLR P2             ; Reset counter (clear Port 2)
    SETB P1.7          ; Turn ON LED (P1.7)

    ; Delay to keep LED ON for some time (simple loop)
    MOV R7, #255
LED_DELAY:
    DJNZ R7, LED_DELAY ; Simple delay loop

    CLR P1.7           ; Turn OFF LED
    POP ACC            ; Restore accumulator
    RETI               ; Return from interrupt

    END                ; End of program
```

In the following program identify the errors (if any) and re-write the correct instruction. If no error found, write "correct" against the instruction. (for 8086 Microprocessor) (i) ADD AX, 1234H (ii) SUB BL, AX (iii) MOV DS, 5060H (iv)PUSH [BP+04H] (v) CMP CX, 5000D

(i) ADD AX, 1234H

- **Explanation**: This is a valid instruction. It adds the immediate value 1234H to the AX register.
- **Status**: Correct.

(ii) **SUB BL, AX**

Explanation: This is an error because BL is an 8-bit register and AX is a 16-bit register.

Corrected instruction: SUB BL, AL (Subtract AL from BL, both are 8-bit registers)

(iii) MOV DS, 5060H

Corrected instruction:

MOV AX, 5060H

MOV DS, AX

(iv) PUSH [BP+04H]

Explanation: This is an error because the 8086 microprocessor does not allow pushing memory locations (i.e., operands with indirect addressing) directly onto the stack. You can only push general-purpose registers, segment registers, or immediate values

Corrected instruction:

MOV AX, [BP+04H] ; Load value from memory location [BP+04H] into AX

PUSH AX ; Push the contents of AX onto the stack

(v) CMP CX, 5000D

Status: Correct

(b) If CS = 3500H and DS = 4000H. Find if there is any overlap between these two segments. If so, what is the size of the overlap?

Determine Overlap:

- The **Code Segment** ranges from 35000H to 44FFFH.
- The **Data Segment** ranges from 40000H to 4FFFFH.
- There is an overlap between these segments from 40000H (the start of DS) to 44FFFH (the end of CS).

Size of the Overlap:

- The overlap starts at 40000H and ends at 44FFFH. The size of the overlap is:
  Size of Overlap=44FFFH−40000H+1=5000H (in hexadecimal)=20480 (in decimal)

5. Write an 8086-assembly language program to count the number of 1's (set bits) in an 8-bit binary number stored at memory location 2000H. Store the result (count of 1's) at memory location 3000H.

asm

Copy code

```
    ORG 1000H           ; Origin (start of program)

    MOV SI, 2000H       ; Load the address of the byte (memory 2000H) into SI
    MOV DI, 3000H       ; Load the address where result will be stored (memory 3000H) into

    MOV AL, [SI]        ; Load the byte from memory 2000H into AL
    MOV CL, 08H         ; Set loop counter to 8 (we need to check all 8 bits)
    XOR BL, BL          ; Clear BL to use it as the counter for 1's (set bits)

COUNT_LOOP:
    SHR AL, 1           ; Shift the contents of AL to the right by 1 bit
                        ; The least significant bit (LSB) will be moved into CF
    JNC NO_INCREMENT    ; If CF = 0 (no carry, the bit is 0), skip incrementing
    INC BL              ; Otherwise, increment BL (found a 1)

NO_INCREMENT:
    DEC CL              ; Decrement loop counter (CL)
    JNZ COUNT_LOOP      ; Repeat the loop until all 8 bits are processed

    MOV [DI], BL        ; Store the result (count of 1's) in memory location 3000H

    HLT                 ; Stop execution (optional)

    END                 ; End of program
```