

MES CA-1 ASSIGNMENT

Q.1) Write an 8051 Assembly Language Program (ALP) to generate the last four digits of your PRN using any arithmetic instructions. The program should not directly load the complete PRN number as an immediate value. Instead, it must use appropriate arithmetic operations such as ADD, MUL, or INC to form the number logically. The final result must be stored in the Accumulator register (AX). For example, if a student's PRN is 24070521211, the last four digits are 1211, and the value 1211 should be available in AX at the end of program execution.

The screenshot displays the Proteus 8051 simulator interface. The central window shows the assembly program for the 8051 microcontroller, which is designed to calculate the last four digits of a PRN (1211) using arithmetic instructions. The program is as follows:

```

0000| MOV A, #100
0002| MOV B, #11
0005| MUL AB
0006| ADD A, #66
0008| MOV DPL, A
000A| MOV DPH, B
      END
  
```

The left panel shows the 8051 register set and memory. The PC (Program Counter) is at 0x000D. The right panel shows the I/O devices, including a keypad, a display, and an ADC. The bottom panel shows the hardware interface, including a keyboard, a display, and an ADC. The display shows the value 11111111, and the ADC shows the value 11111111.

Q.2) Execute an 8051-assembly language program for a safety-certified system in which the instructions CJNE, DJNZ, and SUBB are not permitted. Two unsigned numbers are stored in internal RAM locations 50H and 51H. The program must compare these two numbers using only the allowed instruction set (MOV, INC, DEC, JZ, JNZ, CLR, SETB, ANL, ORL) and store the comparison result in a register or memory location such that 01H indicates the value at 50H is greater than the value at 51H, 00H indicates both values are equal, and FFH indicates the value at 50H is less than the value at 51H. The program should be simulated for all three possible cases ($A > B$, $A = B$, $A < B$), and the solution must clearly explain how flag behavior (especially the Zero flag) is utilized to achieve comparison under the given instruction constraints.

The image shows a Proteus 8051 simulator interface. The top section displays the assembly code being executed. The middle section shows the register and memory status. The bottom section shows the hardware components and their values.

Assembly Code:

```

RST Assm Run New Load Save CPY Paste BP
Reset: PC = 0x0000
LOOP:
JZ A_IS_ZERO
DEC A
XCH A, B
JZ B_IS_ZERO
DEC A
XCH A, B
SJMP LOOP
A_IS_ZERO:
XCH A, B
JZ EQUAL
MOV R7, #0FFH
SJMP DONE
B_IS_ZERO:
MOV R7, #01H
SJMP DONE
EQUAL:
MOV R7, #00H
DONE:
END

```

Registers and Memory:

R/O	W/O	TH0	TL0	R7	B
0x00	0x00	0x00	0x00	0x00	0x00
RXD	TXD	TMOD	0x00	R6	ACC
1	1	0x00	0x00	0x00	0x00
SCON	0x00	TCON	0x00	R5	PSW
				0x00	0x00
pins	bits	TH1	TL1	R4	IP
0xFF	0xFF	0x00	0x00	0x00	0x00
P3				R3	IE
0xFF	0xFF			0x00	0x00
P2				R2	PCON
0xFF	0xFF			0x00	0x00
P1				R1	DPH
0xFF	0xFF			0x00	0x00
P0				R0	DPL
0xFF	0xFF			0x00	0x00

Data Memory:

addr	0x00	0x00	value
0	00	00	00
1	00	00	00
2	00	00	00
3	00	00	00
4	00	00	00
5	00	00	00
6	00	00	00
7	00	00	00
8	00	00	00
9	00	00	00
A	00	00	00
B	00	00	00
C	00	00	00
D	00	00	00
E	00	00	00
F	00	00	00

Hardware Components:

- DI: 1, LD: 1
- AND Gate Disabled
- Key Bounce Disabled
- Standard
- 8-bit UART @ 4800 Baud
- Rx Reset
- Tx Send
- 0.0 V output
- Scope
- DAC
- BF: 0, AC: 0x00, IR: 0x00, DR: 0x00
- 8888
- 0.0 V input
- 1111111
- ADC
- MAX
- MIN
- Motor Enabled

Q.3) A student claims that two assembly programs are equivalent because both access the same RAM address; however, this claim is incorrect due to the difference in addressing modes. In this case study, write two short assembly programs—one using direct addressing and the other using indirect addressing—such that both reference the same RAM location. Using an appropriate initial RAM configuration, demonstrate a situation where the outputs of the two programs differ even though the base address is the same. Support the observation with register and RAM snapshots from simulation, and explain that the difference arises because direct addressing accesses the data stored at the given address, whereas indirect addressing treats the contents of that address as a pointer to another memory location, leading to different data being fetched and hence different outputs.

The screenshot displays the Proteus 8.10 SP3 simulation environment. The central window shows the assembly code for an 8051 microcontroller:

```

0000| MOV 40H, #50H
0003| MOV 50H, #60H
0006| MOV A, 40H
0008| MOV R0, 40H
000A| MOV A, @R0
  
```

The left panel shows the register and memory status:

- System Clock (MHz):** 12.0
- Update Freq.:** 1
- PC:** 0x0029
- PSW:** 00000000
- SP:** 0x07
- RAM Address:** 0x00, **Value:** 0x00

The right panel shows the I/O devices and their status:

- P0.7:** 1 (Display-select Decoder CS|DAC WR)
- P0.6:** 1 (Keypad Column 2)
- P0.5:** 1 (Keypad Column 1)
- P0.4:** 1 (Keypad Column 0)
- P0.3:** 1 (Keypad Row 3)
- P0.2:** 1 (Keypad Row 2)
- P0.1:** 1 (Keypad Row 1)
- P0.0:** 1 (Keypad Row 0)
- P1.7:** 1 (LED 7|Seg. dp|DAC DB7|LCD DB7)
- P1.6:** 1 (LED 6|Seg. g|DAC DB6|LCD DB6)
- P1.5:** 1 (LED 5|Seg. f|DAC DB5|LCD DB5)
- P1.4:** 1 (LED 4|Seg. e|DAC DB4|LCD DB4)
- P1.3:** 1 (LED 3|... d|..DB3|..DB3|.. RS)
- P1.2:** 1 (LED 2|... c|..DB2|..DB2|LCD E)
- P1.1:** 1 (LED 1|Seg. b|DAC DB1|LCD DB1)
- P1.0:** 1 (LED 0|Seg. a|DAC DB0|LCD DB0)
- P2.7:** 1 (SW 7|ADC DB7)
- P2.6:** 1 (SW 6|ADC DB6)
- P2.5:** 1 (SW 5|ADC DB5)
- P2.4:** 1 (SW 4|ADC DB4)
- P2.3:** 1 (SW 3|ADC DB3)
- P2.2:** 1 (SW 2|ADC DB2)
- P2.1:** 1 (SW 1|ADC DB1)
- P2.0:** 1 (SW 0|ADC DB0)
- P3.7:** 1 (ADC RD|Comparator Output)
- P3.6:** 1 (ADC WR)
- P3.5:** 1 (Motor Sensor)
- P3.4:** 1 (Display-select Input 1)
- P3.3:** 1 (AND Gate Output|Display-se..t 0)
- P3.2:** 1 (ADC INTR)
- P3.1:** 1 (Motor Control Bit 1|Ext. UART Rx)
- P3.0:** 1 (Motor Control Bit 0|Ext. UART Tx)

The bottom panel shows the hardware components and their outputs:

- DI:** 1, **LD:** 1
- AND Gate Disabled:** 1
- Key Bounce Disabled:** 1
- Standard:** 1
- 0.0 V output:** 0.0 V
- Scope:** DAC
- ADC:** 11111111
- Motor Enabled:** 1

Q.4) Write an 8051 Assembly Language Program in which you must use logical instructions to construct a numeric result. Using multiple logical instructions such as ANL, ORL, and CLR, generate the last four digits of your own mobile number through a suitable sequence of operations (you may split the digits and combine them logically as required). Do not directly load the complete 4-digit number as an immediate value. The program should use more than one logical instruction, and at the end of execution the Accumulator (A) must contain the last four digits of your mobile number. Simulate the program and verify that the final value in the Accumulator matches your mobile number's last four digits.

The image displays a Proteus 8051 simulator interface. The top window shows the assembly code for an 8051 microcontroller. The code is as follows:

```

ORG 0000H
0000| CLR A
0001| ORL A, #03H
0003| ORL A, #70H
0005| MOV R0, A

0006| CLR A
0007| ORL A, #10H
0009| ORL A, R0

END

```

The bottom window shows the hardware components of the simulator, including a keyboard, a display, a scope, and a motor. The display shows the value 0000. The scope shows the output of the DAC, which is 0.0 V. The motor is enabled.

Q.5) An embedded logger stores event codes in internal RAM from 40H to 5FH, but due to strict memory limitations the data must be compacted in-place without using any additional RAM or the stack. Write an assembly language program that scans the memory range 40H–5FH using only indirect addressing, removes all occurrences of the value FFH, shifts the remaining valid data bytes to the left to eliminate gaps, and fills the unused memory locations at the end of the range with 00H. Execute the program to show the RAM contents before and after execution, and clearly explain the pointer movement logic used to identify valid data, shift it correctly, and overwrite invalid entries under the given constraints.

The screenshot displays the Proteus simulation environment with the following components:

- Register Window:** Shows the state of registers R0-R7, B, ACC, PSW, IP, IE, PCON, DPH, DPL, SP, and PC. The PC register is highlighted at 8051.
- Data Memory Window:** Displays the contents of internal RAM from address 00 to 70. The range 40H to 5FH is highlighted in blue.
- Assembly Editor:** Contains the following assembly code:


```

      ORG 0000H
      0000 MOV R0, #40H
      0002 MOV R1, #40H
      0004 MOV R2, #20H
      SCAN:
      0006 MOV A, @R0
      0007 CJNE A, #0FFH, KEEP
      000A SJMP NEXT
      KEEP:
      000C MOV @R1, A
      000D INC R1
      NEXT:
      000E INC R0
      000F DJNZ R2, SCAN
      0011 MOV A, #00H
      FILL:
      0013 CJNE R1, #60H, DONE
      0016 MOV @R1, A
      0017 INC R1
      0018 SJMP FILL
      DONE:
      
```
- Peripheral Components:** Includes a keypad, LEDs, an ADC, a motor sensor, and a display.