

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 shows the EDISIM5 software environment. On the left, there is a memory dump window for the 8051 microcontroller. It displays registers like R0 through R7, PSW, and various port pins (P0-P3). The assembly code window shows the following program:

```

RST|Step|Run|New|Load|Save|CPY|Paste|BP| Time: illus - Instructions: 6
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 right side of the interface shows a pin configuration for the 8051. It includes pins P0.0 to P0.7, P1.0 to P1.7, P2.0 to P2.7, and P3.0 to P3.5. Various digital outputs are mapped to external components like a display-select decoder, keypad columns, and LED segments. Below the pin configuration, there is a logic analysis window showing a digital signal labeled "00000000000000000000000000000000".

Memory Dump (Data Memory):

addr	0x00	A	B	C	D	E	F							
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00
60	00	00	00	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Copyright ©2005-2024 James Rogers Remove All Breakpoints

Hardware Setup (Bottom):

The bottom part of the interface shows a simulated breadboard setup. Components include a DAC, ADC, motor driver, and various logic and serial communication modules. A digital display shows the value "8888".

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 screenshot displays the Proteus SIM software interface for an 8051 microcontroller simulation. The top portion shows the assembly code editor with the following assembly listing:

```

System Clock (MHz) 12.0
R/O W/O TH0 TL0 R7 0x00 B 0x00
RxD TXD 1 1 TMOD 0x00 R6 0x00 ACC 0x00
SCON 0x00 TCON 0x00 R5 0x00 PSW 0x00
pins bits TH1 TL1 R1 0x00 IP 0x00
0xFF 0xFF P3 0x00 0x00 R0 0x00 IE 0x00
0xFF 0xFF P2 PC DPH 0x00
0xFF 0xFF P1 0x0000 i PSW 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0xFF P0 0x0000 i PSW 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Modify RAM
Data Memory addr 0x00 0x00 value
0 1 2 3 4 5 6 7 8 9 A B C D E F
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
70 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Copyright ©2005-2024 James Rogers
Remove All Breakpoints

```

The assembly code listing includes the following labels and instructions:

- LOOP:** JZ A_IS_ZERO
- A_IS_ZERO:** DEC A, XCH A, B, JZ B_IS_ZERO
- B_IS_ZERO:** DEC A, XCH A, B, SJMP LOOP
- EQUAL:** JZ EQUAL
- DONE:** MOV R7, #0FFH, SJMP DONE
- B_EQUAL:** MOV R7, #01H, SJMP DONE
- EQUAL:** MOV R7, #00H, SJMP DONE
- DONE:** END

The right side of the interface shows a list of port pins and their corresponding functions:

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

The bottom half of the interface shows the hardware schematic and simulation results. The schematic includes components like a keypad, LCD, ADC, DAC, and various switches and sensors. The simulation results show digital logic levels and analog waveforms for the various ports and sensors.

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 shows a 8051 microcontroller simulation environment. At the top, there's a control bar with buttons for RST, Step, Pause, Run, Stop, Save, CPY, Paste, and BP. Below it, the assembly code window displays the following instructions:

```

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

The Registers window shows the state of various registers:

R/O	W/O	TH0	TL0	R7	0x00	B	0x00
0x00	0x00	0x00	0x00	R6	0x00	ACC	0x60
RXD	TXD	TMOD	0x00	R5	0x00	PSW	0x00
1	1	TCON	0x00	R4	0x00	IP	0x00
SCON	0x00	TCON	0x00	R3	0x00	IE	0x00
pins	bits	TH1	TL1	R2	0x00	PCON	0x00
0xFF	0xFF P3	0x00	0x00	R1	0x00	DPH	0x00
0xFF	0xFF P2	PC		R0	0x50	DPL	0x00
0xFF	0xFF P1					SP	0x07
0xFF	0xFF P0						

The CPU ID is listed as 8051. The RAM window shows a 16x16 grid of memory addresses from 00 to FF. The bottom left corner of the interface displays the copyright notice: Copyright ©2005-2024 James Rogers.

The bottom half of the screen shows the physical hardware setup. It includes a digital input port (DI), a logic level converter (LD), a keypad with AND Gate Disabled and Key Bounce Disabled options, a 8-bit UART module set to 4800 Baud with No Parity, a DAC output, a Scope input, and a motor control section with ADC inputs for MAX and MIN values. The digital output section shows a 4-digit LED display showing "8888".

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 screenshot shows the Proteus 8.8.8 software interface with the following components:

- Top Bar:** RST, Step, Pause, Run, Stop, Break, CPY, Paste, BP.
- System Clock (MHz):** 12.0.
- Update Freq.:** 1.
- Memory View:** Shows registers and memory dump for the 8051 microcontroller.
- Registers:** TH0, TL0, R7, R6, R5, R4, R3, R2, PCON, DPH, DPL, PSW, IE, IP, PSW, and TCON.
- Port Definitions:** RXD, TXD, SCON, pins, bits, TH1, TL1, R1, R0, DPH, DPL, PC, and PSW.
- Display:** Shows the value 8051.
- Data Memory:** A table for modifying RAM values.
- Breakpoints:** Remove All Breakpoints button.
- Bottom Panel Components:**
 - Digital Input (DI) with state i and logic level LD.
 - 7-Segment Display showing 8888.
 - ADC Input with MAX and MIN levels.
 - UART Port settings: 8-bit UART @ 4800 Baud, No Parity, Rx, Tx, Rx Reset, and Tx Send.
 - Keybounce Settings: AND Gate Disabled, Key Bounce Disabled, Standard.
 - Scope and DAC controls.
 - MAX/MIN potentiometer.

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 shows the F1SIM5 development environment. On the left, the 'Data Memory' window displays the initial state of RAM from 40H to 5FH. The memory starts with several FFH values, followed by valid data (e.g., 8051). The 'Modify RAM' section allows changing specific memory cells. The assembly code window on the right shows the assembly language program for compacting the data. The hardware interface at the bottom features various digital and analog input/output components like AND gates, UART, DAC, ADC, and motor control modules.

```

System Clock (MHz) 12.0
RST Step Pause CPY Paste BP
Time: 3lus - Instructions: 24
U + | 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

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:

```