

NAME: Rutva Sawarkar

PRN: 24070521189

SEM: 4th, C

MICROCONTROLLERS AND EMBEDDED SYSTEMS

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 8.09 IDE interface. The top-left pane shows the 8051 microcontroller's register and pin configuration. The top-right pane contains the assembly code for the program. The bottom-left pane shows the hardware simulation, including a keypad, a display, and various control buttons.

Assembly Code:

```
ORG 0000H
0000 MOV A, #08H
0002 MOV B, #10H
0005 MUL AB
0006 ADD A, #09H
0008 MOV R0, A
0009 MOV A, #01H
000B MOV B, #10H
000E MUL AB
000F ADD A, #01H
0011 MOV B, A
0013 MOV A, R0
END
```

Hardware Simulation:

- Keypad:** A 4x4 grid of buttons labeled 0-9, *, and #.
- Display:** A 4-digit 7-segment display showing "1111".
- ADC:** A vertical slider control labeled "ADC" with a value of "11111111".
- Motor:** A circular motor icon labeled "Motor Enabled".
- Other Controls:** Buttons for "AND Gate Disabled", "Key Bounce Disabled", "Standard", "Rx Reset", and "Tx Send".

NAME: Rutva Sawarkar

PRN: 24070521189

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.

CASE1:

The screenshot displays the 8051 simulator interface. On the left, the hardware configuration panel shows various registers and their values. The PC (Program Counter) is set to 0x0000. The PSW (Program Status Word) is 0x0000. The ACC (Accumulator) is 0x00. The R0 register contains 0x09. The R1 register contains 0x04. The R2 register contains 0x00. The R3 register contains 0x00. The R4 register contains 0x00. The R5 register contains 0x00. The R6 register contains 0x00. The R7 register contains 0x00. The B register contains 0x00. The TH0 register contains 0x00. The TL0 register contains 0x00. The TMOD register contains 0x00. The TCON register contains 0x00. The TH1 register contains 0x00. The TL1 register contains 0x00. The P0 register contains 0xFF. The P1 register contains 0xFF. The P2 register contains 0xFF. The P3 register contains 0xFF. The pins are configured as follows: RXD (1), TXD (1), SCON (0x00). The Data Memory panel shows the values at addresses 0x50 and 0x51. The assembly code panel on the right shows the following code:

```
RST Assm Run New Load Save CPY Paste BP
Reset: PC = 0x0000

MOV R0, 50H ; Copy A
MOV R1, 51H ; Copy B
COMPARE:
DEC R0 ; Decrement A c
DEC R1 ; Decrement B c
JZ A_ZERO ; Did R1 just b
MOV A, R0 ; Check R0
JZ B_GREATER ; If R0 is zero
SJMP COMPARE ; Neither is ze

A_ZERO:
MOV A, R0
JZ EQUAL ; Both zero ? e
SJMP A_GREATER ; R1 zero first

A_GREATER:
MOV 52H, #01H
SJMP DONE

B_GREATER:
```

CASE 2:

NAME: Rutva Sawarkar PRN: 24070521189

System Clock (MHz) 12.0 1 Update Freq.

SBUF

R/O	W/O	TH0	TL0	R7	B
0x00	0x00	0x00	0x00	0x00	0x00

RXD	TXD	TMOD	TCON	R6	ACC
1	1	0x00	0x00	0x00	0x00

pins	bits	TH1	TL1	R5	PSW
0xFF	0xFF	0x00	0x00	0x00	0x00

bits	TH1	TL1	R4	IP
0xFF	0xFF	0x00	0x00	0x00

bits	TH1	TL1	R3	IE
0xFF	0xFF	0x00	0x00	0x00

bits	TH1	TL1	R2	PCON
0xFF	0xFF	0x00	0x00	0x00

bits	TH1	TL1	R1	DPH
0xFF	0xFF	0x00	0x00	0x00

bits	TH1	TL1	R0	DPL
0xFF	0xFF	0x00	0x00	0x00

PC 0x0043 8051

Modify RAM

addr	0x51	0x04	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

Copyright ©2005-2024 James Rogers Remove All Breakpoints

RST Step Run New Load Save CPY Paste BP

Time: 75us - Instructions: 63

```
; Load values
0000| MOV A, 50H
0002| MOV R0, A ; A
0003| MOV A, 51H
0005| MOV R1, A ; B

COMPARE:
0006| DEC R0
0007| DEC R1

0008| MOV A, R0
0009| JZ R0_ZERO ; A exhausted

000B| MOV A, R1
000C| JZ R1_ZERO ; B exhausted

000E| SJMP COMPARE

R0_ZERO:
0010| MOV A, R1
```

Case 3:

System Clock (MHz) 12.0 1 Update Freq.

SBUF

R/O	W/O	TH0	TL0	R7	B
0x00	0x00	0x00	0x00	0x00	0x00

RXD	TXD	TMOD	TCON	R6	ACC
1	1	0x00	0x00	0x00	0x00

pins	bits	TH1	TL1	R5	PSW
0xFF	0xFF	0x00	0x00	0x00	0x00

bits	TH1	TL1	R4	IP
0xFF	0xFF	0x00	0x00	0x00

bits	TH1	TL1	R3	IE
0xFF	0xFF	0x00	0x00	0x00

bits	TH1	TL1	R2	PCON
0xFF	0xFF	0x00	0x00	0x00

bits	TH1	TL1	R1	DPH
0xFF	0xFF	0x00	0x05	0x00

bits	TH1	TL1	R0	DPL
0xFF	0xFF	0x00	0x00	0x00

PC 0x0000 8051

Modify RAM

addr	0x50	0x04	value
0	00	05	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

Copyright ©2005-2024 James Rogers Remove All Breakpoints

RST Assm Run New Load Save CPY Paste BP

Reset: PC = 0x0000

```
; Load values
MOV A, 50H
MOV R0, A ; A
MOV A, 51H
MOV R1, A ; B

COMPARE:
DEC R0
DEC R1

MOV A, R0
JZ R0_ZERO ; A exhausted first

MOV A, R1
JZ R1_ZERO ; B exhausted first

SJMP COMPARE

R0_ZERO:
MOV A, R1
```

NAME: Rutva Sawarkar

PRN: 24070521189

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.

Direct Addressing mode

The screenshot displays a 8051 microcontroller simulator interface. The top section shows the assembly code being executed:

```
0000 MOV 30H, #90H
0003 MOV A, 30H
END
```

The register window shows the following values:

Register	Value
R0	0x00
R1	0x00
R2	0x00
R3	0x00
R4	0x00
R5	0x00
R6	0x00
R7	0x00
B	0x00
ACC	0x90
PSW	0x00
IP	0x00
IE	0x00
PCON	0x00
DPH	0x00
DPL	0x00
SP	0x07

The Data Memory window shows the following values:

Address	Value
00	00
01	00
02	00
03	00
04	00
05	00
06	00
07	00
08	00
09	00
0A	00
0B	00
0C	00
0D	00
0E	00
0F	00

The hardware status window shows the following values:

Component	Value
DI	1
LD	1
AND Gate Disabled	1
Key Bounce Disabled	1
Standard	1
U	No Parity
8-bit UART @	4800 Baud
Rx	
Rx Reset	
Tx	
Tx Send	
0.0 V output	
Scope	
DAC	
BF	0
AC	0x00
IR	0x00
DR	0x00
8.8.8.8	
0.0 V input	
ADC	11111111
MAX	
MIN	
Motor Enabled	

NAME: Rutva Sawarkar

PRN: 24070521189

Indirect addressing mode

The screenshot displays the Proteus 8051 simulator interface. The top section shows the assembly code window with the following instructions:

```
0000| MOV 30H, #40H
0003| MOV 40H, #99H
0006| MOV R0, 30H
0008| MOV A, @R0
END
```

The register window shows the following values:

Register	Value
R0	0x40
R1	0x00
R2	0x00
R3	0x00
R4	0x00
R5	0x00
R6	0x00
R7	0x00
B	0x00
ACC	0x99
PSW	0x00
IP	0x00
IE	0x00
PCON	0x00
DPH	0x00
DPL	0x00
SP	0x07

The data memory window shows the following values:

Address	Value
0	00
1	00
2	00
3	00
4	00
5	00
6	00
7	00
8	00
9	00
A	00
B	00
C	00
D	00
E	00
F	00

The hardware components window shows the following components:

- DI: 1, LD: 1
- AND Gate Disabled
- Key Bounce Disabled
- Standard
- 8-bit UART @ 4800 Baud
- No Parity
- Rx Reset
- Tx Send
- 0.0 V input
- ADC
- MAX
- MIN
- Motor Enabled

The bottom section shows the 8051 microcontroller and its internal components, including the 8051 core, 2K RAM, and 2K ROM.

NAME: Rutva Sawarkar

PRN: 24070521189

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 displays the EdSim51DI - Version 2.1.39 simulation environment. The main window is divided into several sections:

- System Clock (MHz):** Set to 12.0.
- SBUF:** A table showing R/O, W/O, TH0, TL0, R7, B, R6, ACC, R5, PSW, R4, IP, R3, IE, R2, PCON, R1, DPH, R0, DPL, and SP registers.
- Data Memory:** A table showing memory addresses from 0 to 70 and their corresponding values.
- Assembly Code:** A list of instructions: `ORG 0000H`, `0000 MOV A, #0FFH`, `0002 ANL A, #70H`, `0004 MOV R1, #08H`, `0006 ORL A, R1`, `0007 MOV B, A`, `0009 CLR A`, `000A ORL A, #60H`, `000C MOV R2, #07H`, `000E ORL A, R2`, and `END`.
- Hardware Interface:** A green panel at the bottom showing various components: a keyboard, a display (showing 8888), a scope, a DAC, a UART interface (8-bit UART @ 4800 Baud), an ADC (showing 1111111), and a motor control section (Motor Enabled).

NAME: Rutva Sawarkar

PRN: 24070521189

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 8.09 IDE interface with an 8051 assembly program and its execution results.

Assembly Program:

```
MOV 40H, #12H
MOV 41H, #0FFH
MOV 42H, #34H
MOV 43H, #0FFH
MOV 44H, #56H
MOV 45H, #78H
MOV 46H, #0FFH
MOV 47H, #9AH
MOV 48H, #0BCH
MOV 49H, #0FFH
MOV 4AH, #0DEH
MOV 4BH, #0FFH
MOV 4CH, #0FFH
MOV 4DH, #0ABH
MOV 4EH, #0CDH
MOV 4FH, #0FFH
MOV 50H, #0EFH
MOV 51H, #23H
MOV 52H, #0FFH
MOV 53H, #45H
MOV 54H, #67H
```

Data Memory:

addr	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
00	00	60	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	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	12	34	56	78	9A	BC	DE	FO	AB	CD	EF	23	45	67	EE	12
50	34	56	78	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

Execution Results:

- System Clock:** 12.0 MHz
- Time:** 398us - Instructions: 271
- Registers:** R0=0x60, R1=0x60, R2=0x00, R3=0x00, R4=0x00, R5=0x00, R6=0x00, R7=0x00, ACC=0x00, PSW=0x00, IP=0x00, IE=0x00, PCON=0x00, DPH=0x00, DPL=0x00, SP=0x07.
- IO Devices:** Keypad, LEDs, ADC, Motor, Display-select, AND Gate, Scope, DAC, UART.