Name : Priya Prasad

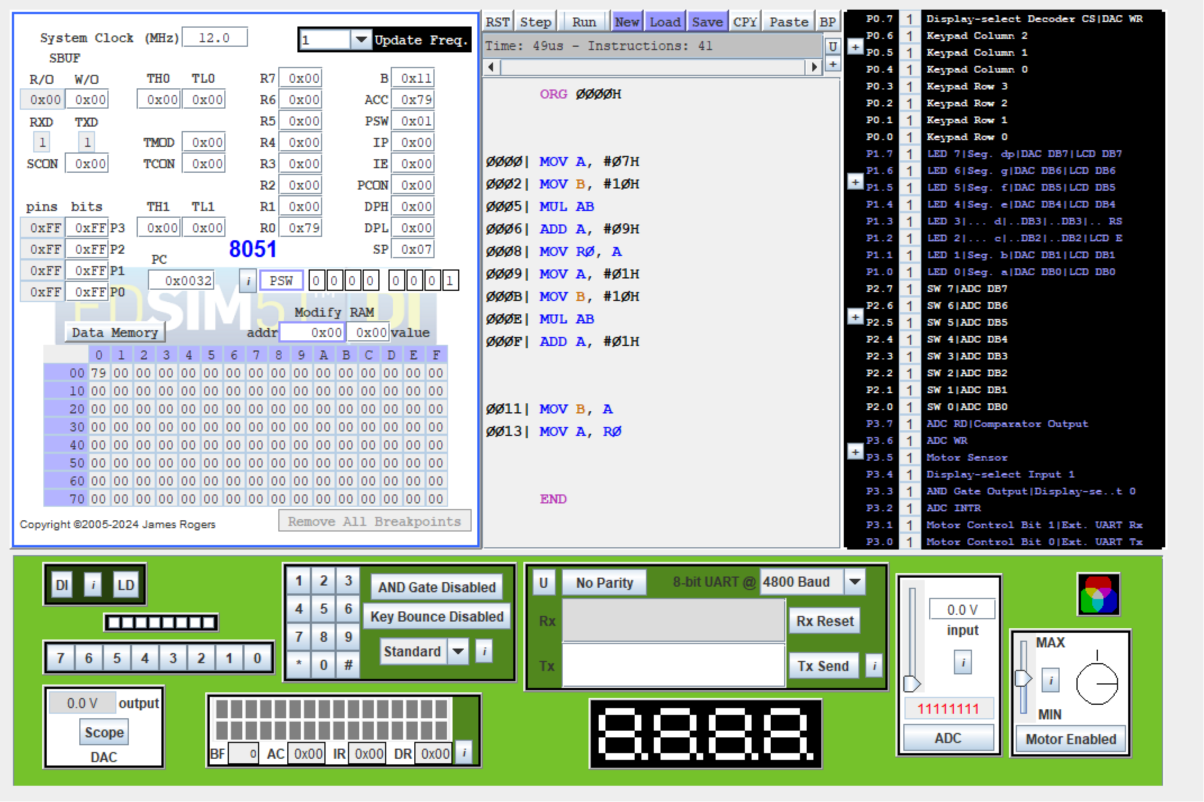
PRN : 24070521179

Sec : C

Sem : 4th

Sub : MES CA1

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.



Time: 49us - Instructions: 41

```
        ORG ØØØØH

ØØØØ| MOV A, #Ø7H
ØØØ2| MOV B, #1ØH
ØØØ5| MUL AB
ØØØ6| ADD A, #Ø9H
ØØØ8| MOV RØ, A
ØØØ9| MOV A, #Ø1H
ØØØB| MOV B, #1ØH
ØØØE| MUL AB
ØØØF| ADD A, #Ø1H


ØØ11| MOV B, A
ØØ13| MOV A, RØ


        END
```

System Clock (MHz) 12.0

R7 0x00, R6 0x00, R5 0x00, R4 0x00, R3 0x00, R2 0x00, R1 0x00, R0 0x79

B 0x11, ACC 0x79, PSW 0x01, IP 0x00, IE 0x00, PCON 0x00, DPH 0x00, DPL 0x00, SP 0x07

PC 0x0032

Data Memory: 00 79

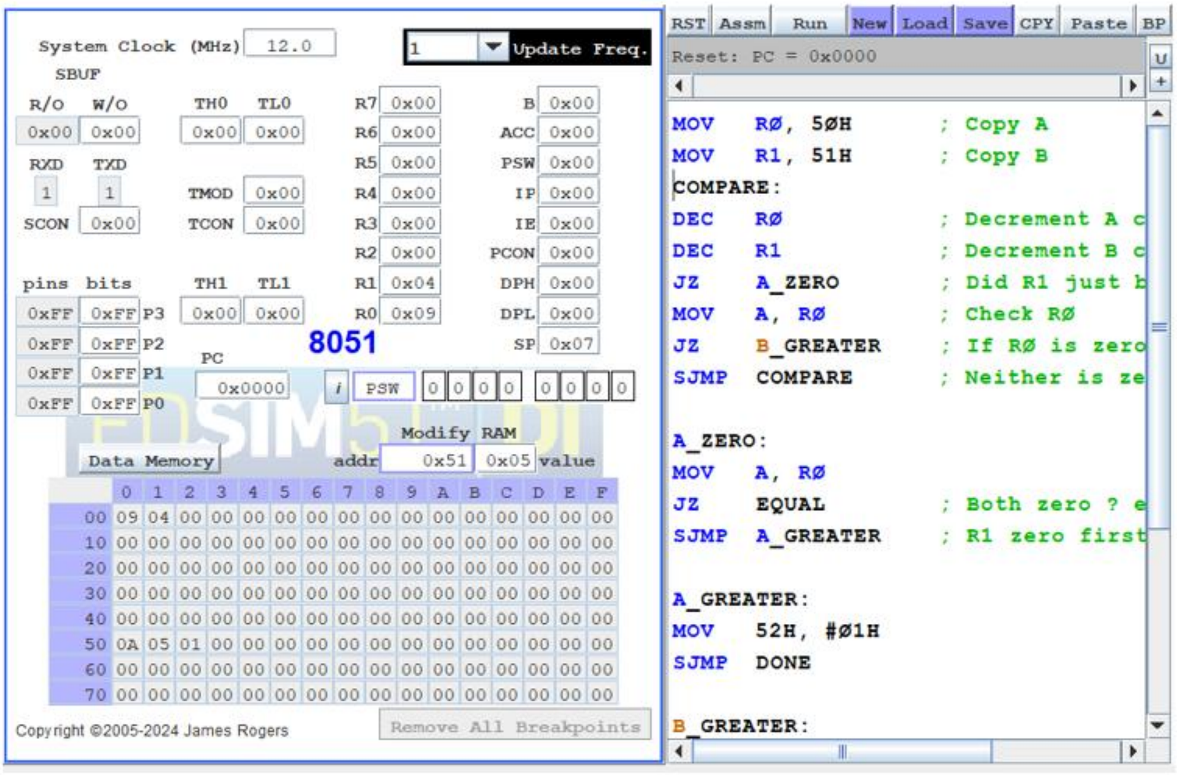
Name : Priya Prasad

PRN : 24070521179

Sec : C

Q2. 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.

CASE I : When A> B. Result is 01H



System Clock (MHz) 12.0

Update Freq. 1

SBUF R/O 0x00 W/O 0x00

RXD 1 TXD 1

SCON 0x00

TH0 0x00 TL0 0x00

TMOD 0x00

TCON 0x00

TH1 0x00 TL1 0x00

R7 0x00
R6 0x00
R5 0x00
R4 0x00
R3 0x00
R2 0x00
R1 0x04
R0 0x09

B 0x00
ACC 0x00
PSW 0x00
IP 0x00
IE 0x00
PCON 0x00
DPH 0x00
DPL 0x00
SP 0x07

pins bits
0xFF 0xFF P3
0xFF 0xFF P2
0xFF 0xFF P1
0xFF 0xFF P0

PC 0x0000

PSW 0 0 0 0 0 0 0 0

Modify RAM addr 0x51 0x05 value

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	09	04	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	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	0A	05	01	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

Reset: PC = 0x0000

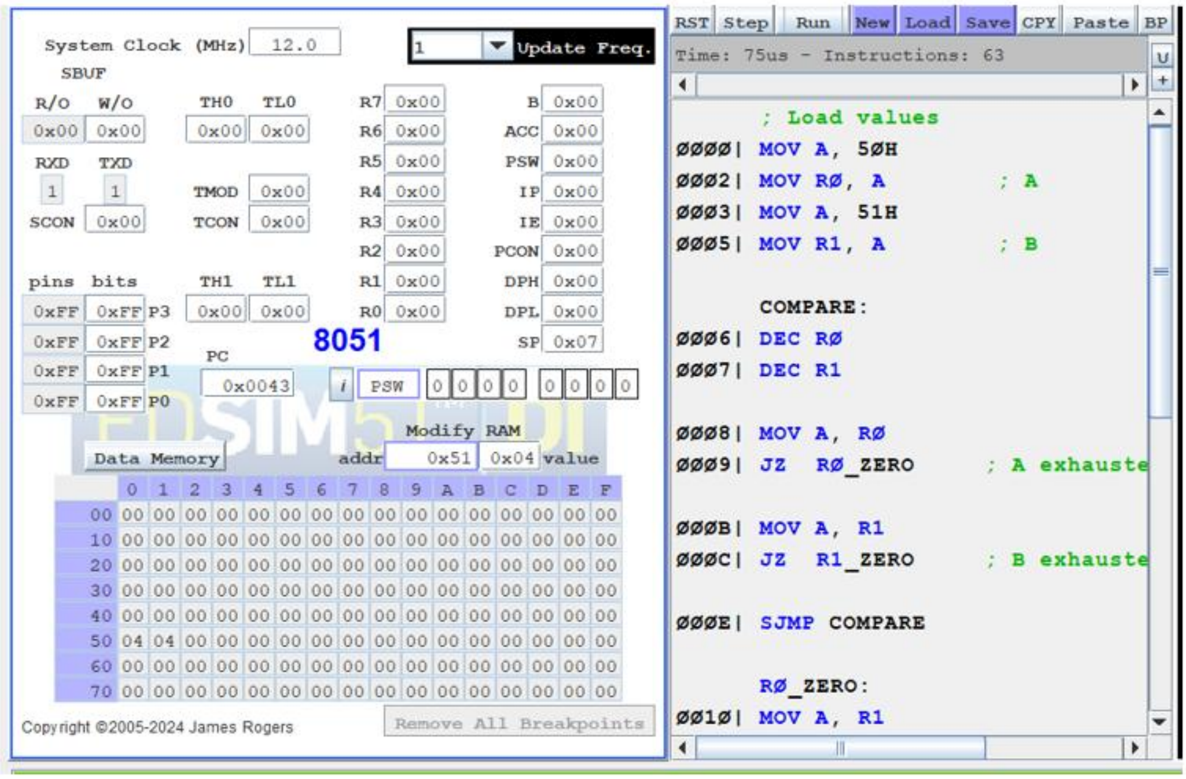
```
MOV   RØ, 5ØH          ; Copy A
MOV   R1, 51H          ; Copy B
COMPARE:
DEC   RØ               ; Decrement A c
DEC   R1               ; Decrement B c
JZ    A_ZERO           ; Did R1 just b
MOV   A, RØ            ; Check RØ
JZ    B_GREATER        ; If RØ is zero
SJMP  COMPARE          ; Neither is ze

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

A_GREATER:
MOV   52H, #Ø1H
SJMP  DONE

B_GREATER:
```

Name : Priya Prasad

PRN : 24070521179

Sec : C

CASE II : When A = B. Result is 00H



```
System Clock (MHz) 12.0      1
SBUF
R/O   W/O      TH0   TL0     R7 0x00        B 0x00
0x00  0x00     0x00  0x00    R6 0x00      ACC 0x00
RXD   TXD                    R5 0x00      PSW 0x00
1     1        TMOD 0x00     R4 0x00       IP 0x00
SCON 0x00      TCON 0x00     R3 0x00       IE 0x00
                             R2 0x00     PCON 0x00
pins  bits     TH1   TL1     R1 0x00      DPH 0x00
0xFF  0xFF P3  0x00  0x00    R0 0x00      DPL 0x00
0xFF  0xFF P2                              SP 0x07
0xFF  0xFF P1  PC
0xFF  0xFF P0  0x0043    i  PSW  0 0 0 0  0 0 0 0
                         Modify RAM
                         addr 0x51  0x04 value
```

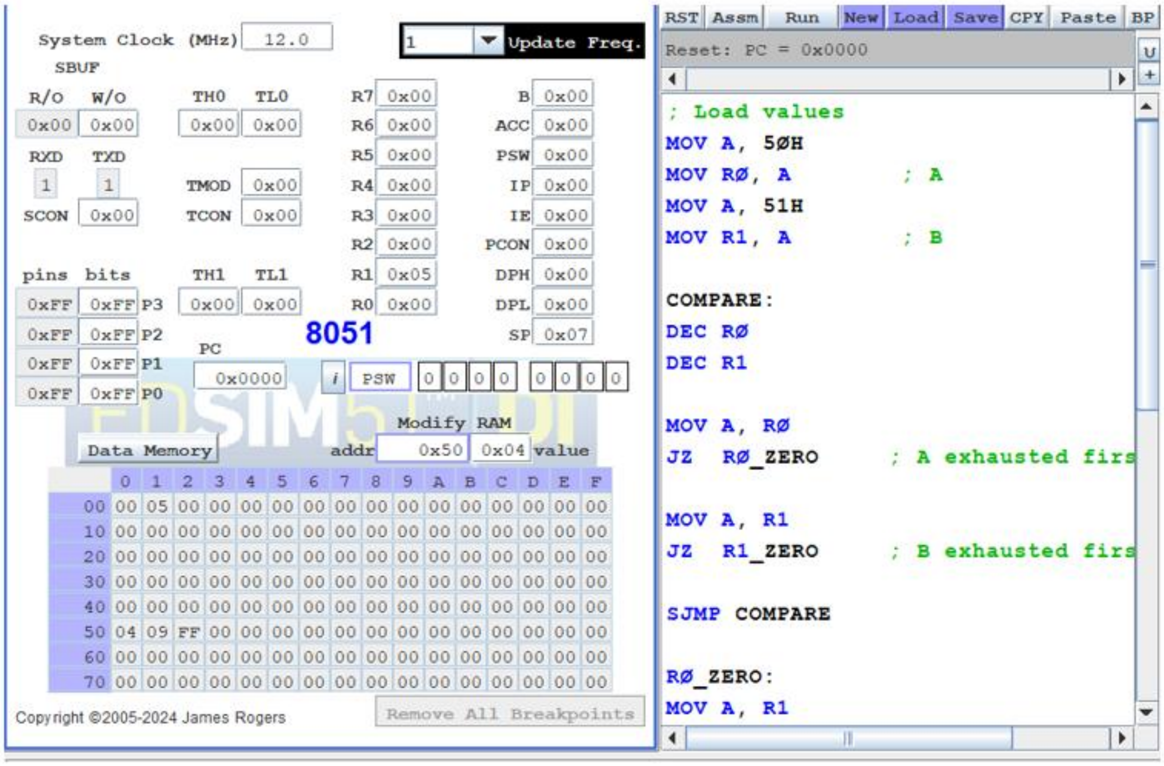
```
    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
50 04 04 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

Time: 75us - Instructions: 63

```
      ; Load values
ØØØØ| MOV A, 5ØH
ØØØ2| MOV RØ, A          ; A
ØØØ3| MOV A, 51H
ØØØ5| MOV R1, A          ; B

      COMPARE:
ØØØ6| DEC RØ
ØØØ7| DEC R1

ØØØ8| MOV A, RØ
ØØØ9| JZ  RØ_ZERO        ; A exhauste

ØØØB| MOV A, R1
ØØØC| JZ  R1_ZERO        ; B exhauste

ØØØE| SJMP COMPARE

      RØ_ZERO:
ØØ1Ø| MOV A, R1
```

CASE III : When A < B. Result is FFH



```
System Clock (MHz) 12.0      1
SBUF
R/O   W/O      TH0   TL0     R7 0x00        B 0x00
0x00  0x00     0x00  0x00    R6 0x00      ACC 0x00
RXD   TXD                    R5 0x00      PSW 0x00
1     1        TMOD 0x00     R4 0x00       IP 0x00
SCON 0x00      TCON 0x00     R3 0x00       IE 0x00
                             R2 0x00     PCON 0x00
pins  bits     TH1   TL1     R1 0x05      DPH 0x00
0xFF  0xFF P3  0x00  0x00    R0 0x00      DPL 0x00
0xFF  0xFF P2                              SP 0x07
0xFF  0xFF P1  PC
0xFF  0xFF P0  0x0000    i  PSW  0 0 0 0  0 0 0 0
                         Modify RAM
                         addr 0x50  0x04 value
```

```
    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00 00 05 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 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
50 04 09 FF 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

Reset: PC = 0x0000

```
; Load values
MOV A, 5ØH
MOV RØ, A          ; A
MOV A, 51H
MOV R1, A          ; B

COMPARE:
DEC RØ
DEC R1

MOV A, RØ
JZ  RØ_ZERO        ; A exhausted firs

MOV A, R1
JZ  R1_ZERO        ; B exhausted firs

SJMP COMPARE

RØ_ZERO:
MOV A, R1
```

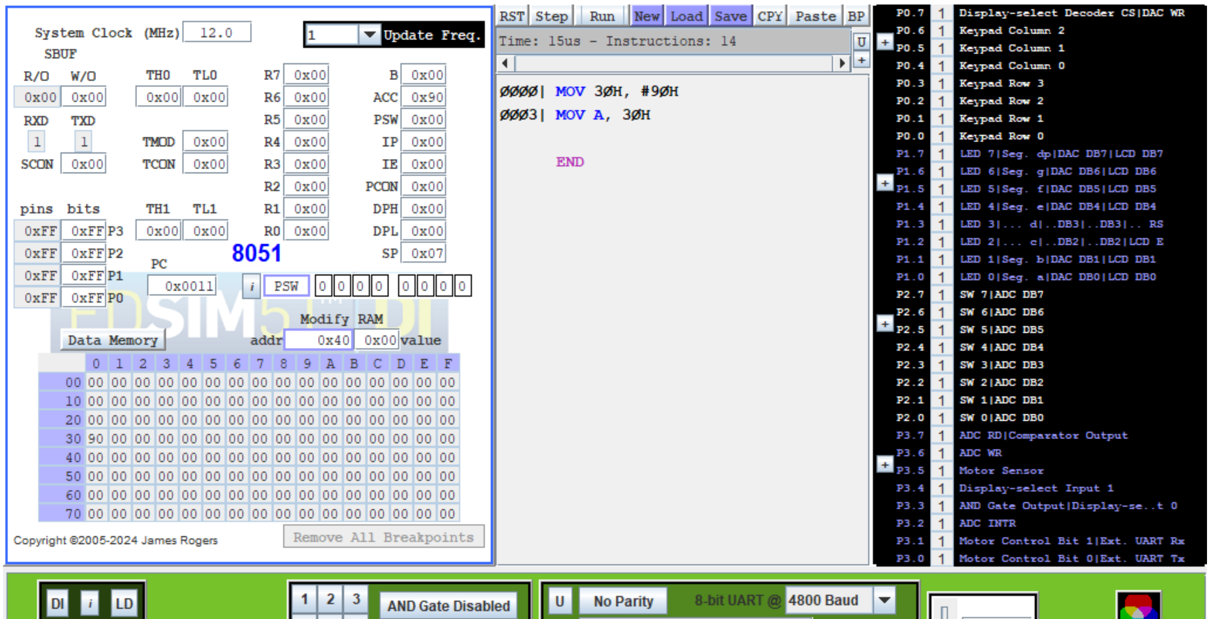
Name : Priya Prasad

PRN : 24070521179

Sec : C

Q3. 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.

CASE 1 Direct



System Clock (MHz) 12.0

Time: 15us - Instructions: 14

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

      END
```

SBUF R/O 0x00 W/O 0x00; RXD 1; TXD 1; SCON 0x00

TH0 0x00; TL0 0x00; TMOD 0x00; TCON 0x00; TH1 0x00; TL1 0x00

R7 0x00; R6 0x00; R5 0x00; R4 0x00; R3 0x00; R2 0x00; R1 0x00; R0 0x00

B 0x00; ACC 0x90; PSW 0x00; IP 0x00; IE 0x00; PCON 0x00; DPH 0x00; DPL 0x00; SP 0x07

pins	bits	
0xFF	0xFF	P3
0xFF	0xFF	P2
0xFF	0xFF	P1
0xFF	0xFF	P0

PC 0x0011; PSW 0 0 0 0 0 0 0 0

Modify RAM addr 0x40 value 0x00

	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	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	90	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

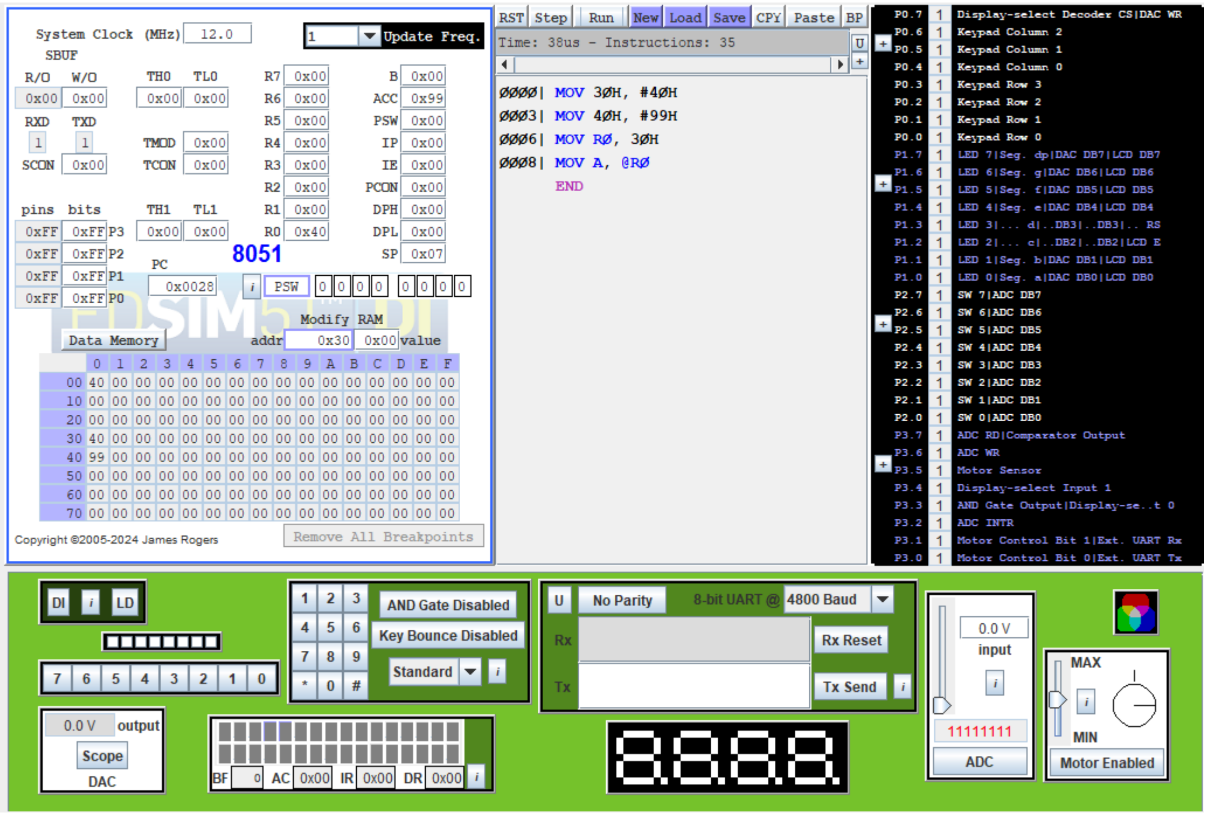
```
P0.7  1  Display-select Decoder CS|DAC WR
P0.6  1  Keypad Column 2
P0.5  1  Keypad Column 1
P0.4  1  Keypad Column 0
P0.3  1  Keypad Row 3
P0.2  1  Keypad Row 2
P0.1  1  Keypad Row 1
P0.0  1  Keypad Row 0
P1.7  1  LED 7|Seg. dp|DAC DB7|LCD DB7
P1.6  1  LED 6|Seg. g|DAC DB6|LCD DB6
P1.5  1  LED 5|Seg. 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
```

Name : Priya Prasad

PRN : 24070521179

Sec : C

CASE 2 Indirect



```
ØØØØ| MOV 3ØH, #4ØH
ØØØ3| MOV 4ØH, #99H
ØØØ6| MOV RØ, 3ØH
ØØØ8| MOV A, @RØ
      END
```

Time: 38us - Instructions: 35

System Clock (MHz) 12.0

Update Freq. 1

SBUF R/O 0x00 W/O 0x00

RXD 1 TXD 1

SCON 0x00

TH0 0x00 TL0 0x00

TMOD 0x00

TCON 0x00

TH1 0x00 TL1 0x00

pins	bits	
0xFF	0xFF	P3
0xFF	0xFF	P2
0xFF	0xFF	P1
0xFF	0xFF	P0

PC 0x0028

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

PSW 0 0 0 0 0 0 0 0

Modify RAM addr 0x30 0x00 value

Data Memory

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	40	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	00
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40	99	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

```
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
```

DAC output 0.0 V

8-bit UART @ 4800 Baud, No Parity

BF 0 AC 0x00 IR 0x00 DR 0x00

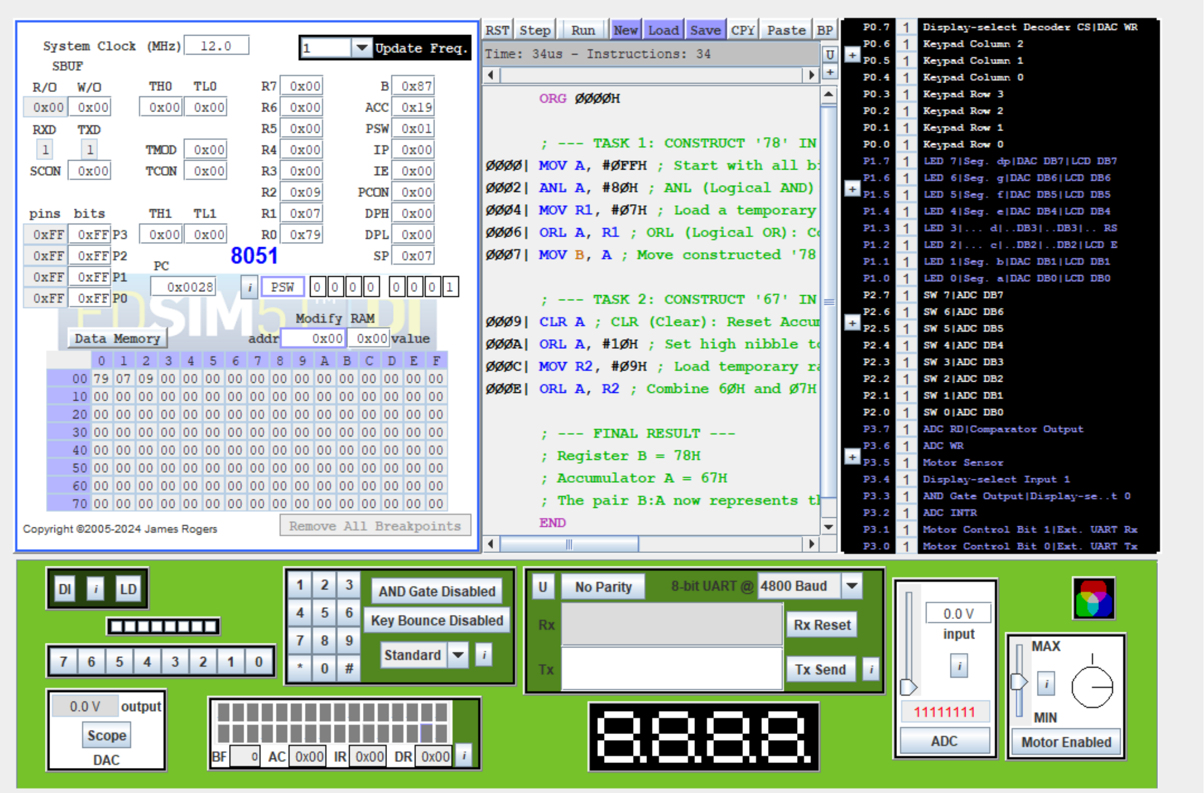
ADC input 0.0 V 11111111

Name : Priya Prasad

PRN : 24070521179

Sec : C

Q4. 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.



```
System Clock (MHz) 12.0        1  Update Freq.
SBUF
R/O   W/O     TH0   TL0     R7 0x00     B    0x87
0x00  0x00    0x00  0x00    R6 0x00     ACC  0x19
RXD   TXD                   R5 0x00     PSW  0x01
1     1       TMOD  0x00    R4 0x00     IP   0x00
SCON  0x00    TCON  0x00    R3 0x00     IE   0x00
                            R2 0x09     PCON 0x00
pins  bits    TH1   TL1     R1 0x07     DPH  0x00
0xFF  0xFF P3 0x00  0x00    R0 0x79     DPL  0x00
0xFF  0xFF P2            8051           SP   0x07
0xFF  0xFF P1   PC
0xFF  0xFF P0   0x0028       PSW 0 0 0 0  0 0 0 1
                       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 79 07 09 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 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

Time: 34us - Instructions: 34

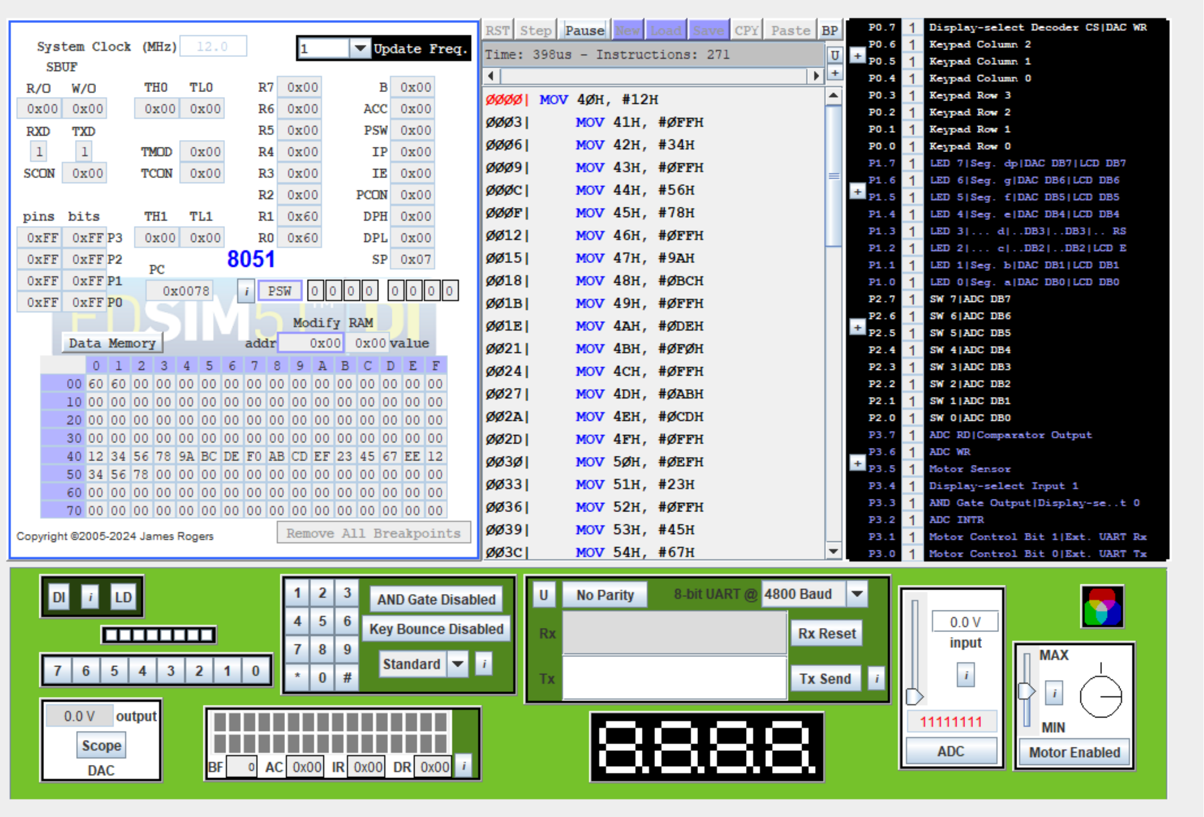
```
      ORG ØØØØH

      ; --- TASK 1: CONSTRUCT '78' IN
ØØØØ| MOV A, #ØFFH ; Start with all bi
ØØØ2| ANL A, #8ØH ; ANL (Logical AND)
ØØØ4| MOV R1, #Ø7H ; Load a temporary
ØØØ6| ORL A, R1 ; ORL (Logical OR): Co
ØØØ7| MOV B, A ; Move constructed '78

      ; --- TASK 2: CONSTRUCT '67' IN
ØØØ9| CLR A ; CLR (Clear): Reset Accum
ØØØA| ORL A, #1ØH ; Set high nibble to
ØØØC| MOV R2, #Ø9H ; Load temporary ra
ØØØE| ORL A, R2 ; Combine 6ØH and Ø7H

      ; --- FINAL RESULT ---
      ; Register B = 78H
      ; Accumulator A = 67H
      ; The pair B:A now represents th
      END
```

Name : Priya Prasad

PRN : 24070521179

Sec : C

Q5. 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.



```
Time: 398us - Instructions: 271

ØØØØ| MOV 4ØH, #12H
ØØØ3|     MOV 41H, #ØFFH
ØØØ6|     MOV 42H, #34H
ØØØ9|     MOV 43H, #ØFFH
ØØØC|     MOV 44H, #56H
ØØØF|     MOV 45H, #78H
ØØ12|     MOV 46H, #ØFFH
ØØ15|     MOV 47H, #9AH
ØØ18|     MOV 48H, #ØBCH
ØØ1B|     MOV 49H, #ØFFH
ØØ1E|     MOV 4AH, #ØDEH
ØØ21|     MOV 4BH, #ØFØH
ØØ24|     MOV 4CH, #ØFFH
ØØ27|     MOV 4DH, #ØABH
ØØ2A|     MOV 4EH, #ØCDH
ØØ2D|     MOV 4FH, #ØFFH
ØØ3Ø|     MOV 5ØH, #ØEFH
ØØ33|     MOV 51H, #23H
ØØ36|     MOV 52H, #ØFFH
ØØ39|     MOV 53H, #45H
ØØ3C|     MOV 54H, #67H
```

```
System Clock (MHz) 12.0          1  Update Freq.
SBUF
R/O    W/O        TH0    TL0      R7 0x00       B 0x00
0x00   0x00       0x00   0x00     R6 0x00     ACC 0x00
RXD    TXD                        R5 0x00     PSW 0x00
 1      1         TMOD   0x00     R4 0x00      IP 0x00
SCON   0x00       TCON   0x00     R3 0x00      IE 0x00
                                  R2 0x00    PCON 0x00
pins   bits       TH1    TL1      R1 0x60     DPH 0x00
0xFF   0xFF P3    0x00   0x00     R0 0x60     DPL 0x00
0xFF   0xFF P2                8051             SP 0x07
0xFF   0xFF P1    PC
0xFF   0xFF P0    0x0078      PSW 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  60 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 F0 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
Copyright ©2005-2024 James Rogers          Remove All Breakpoints
```