



SYMBIOSIS INSTITUTE OF TECHNOLOGY, NAGPUR

Symbiosis International (Deemed University)

(Established under section 3 of the UGC Act, 1956)

Re-accredited by NAAC with 'A++' Grade | Awarded Category – I by UGC

Founder: Prof. Dr. S. B. Mujumdar, M. Sc., Ph. D. (Awarded Padma Bhushan and Padma Shri by President of India)

MES CA-1

NAME: Anaya Shende

PRN: 24070521213

SECTION: D

SEMESTER: 4th

Q.1. Write an 8051 Assembly Language Program (ALP) to generate the last four digits of your PRN using arithmetic instructions only. The final result should be available in the Accumulator (A).

The screenshot displays the SIMS 8051 simulator interface. The main window shows an assembly program for an 8051 microcontroller. The program is as follows:

```
ORG 0000H
MOV A, #06H ; A = 6
MOV B, #06H ; B = 6
ADD A, B ; A = 12 (BCD)
DA A ; Decimal adjust
MOV B, A ; Store 12 in B
INC A ; A = 13
END
```

The hardware interface at the bottom shows the following components:

- System Clock (MHz):** 12.0
- System Memory:** 0x00 to 0xFF (RAM and ROM)
- System I/O:** 0x00 to 0xFF (P0, P1, P2, P3, P4, P5, P6, P7)
- System Registers:** R0 to R7, ACC, PSW, IP, IE, PCON, DPH, DPL, SP
- System Peripherals:** UART, ADC, DAC, Motor, Display, Keypad, Keypad Column, Keypad Row
- System Status:** 0.0 V output, Scope, DAC, Motor Enabled

Q.2. Write an 8051 Assembly Language Program to compare two unsigned numbers stored at internal RAM locations 50H and 51H without using CJNE, DJNZ, or SUBB. Only MOV, INC, DEC, JZ, JNZ, CLR, SETB, ANL, and ORL instructions are allowed. Store the result such that 01H indicates 50H > 51H, 00H indicates equality, and FFH indicates 50H < 51H.

The screenshot displays the Proteus 8051 simulator interface. The main window shows the assembly code for comparing two unsigned numbers stored at internal RAM locations 50H and 51H. The code is as follows:

```

MOV 50H, #06H
MOV 51H, #04H
MOV A, 50H
MOV R0, A ; store first number in R0

MOV A, 51H
MOV R1, A ; store second number in R1

START:
MOV A, R0
JZ CHECK_R1 ; if first becomes zero

MOV A, R1
JZ FIRST_BIG ; if second becomes zero

DEC R0
DEC R1
SJMP START

CHECK_R1:
MOV A, R0
JZ SAME

MOV 50H, #0FFH ; first < second
SJMP END

FIRST_BIG:
MOV 50H, #01H ; first > second
SJMP END

SAME:
MOV 50H, #00H ; equal

END

```

The simulator shows the registers (R0-R7, ACC, PSW, IP, IE, PCON, DPH, DPL, SP) and memory (Data Memory, Modify RAM) windows. The hardware components include a keypad, a 7-segment display, an ADC, and a motor. The display shows the result of the comparison, which is 00H, indicating equality.

Q.3. A student claims that two assembly programs are equivalent because they access the same RAM address. Using one program with direct addressing and another with indirect addressing, demonstrate why this claim is incorrect by showing a case where both reference the same base address but produce different outputs. Explain how the difference in addressing modes causes this behavior.

The image displays two screenshots of the Proteus simulator, illustrating the difference in behavior between direct and indirect addressing modes in assembly programming.

Top Screenshot (Direct Addressing):

- Assembly Code:**

```

ORG 0000H
0000: MOV A, 30H ; da
END

```
- Register Window:** Shows the PC (Program Counter) at 0x007D and the PSW (Program Status Word) at 0x0001.
- Data Memory:** Shows the value 0x00 at address 0x00.
- Output:** The 7-segment display shows 0000.

Bottom Screenshot (Indirect Addressing):

- Assembly Code:**

```

ORG 0000H
0000: MOV R0, #30H
0002: MOV A, @R0 ; ida
END

```
- Register Window:** Shows the PC (Program Counter) at 0x0025 and the PSW (Program Status Word) at 0x0001.
- Data Memory:** Shows the value 0x00 at address 0x00.
- Output:** The 7-segment display shows 11111111.

The difference in behavior is caused by the addressing mode used in the MOV instruction. In the top program, the MOV instruction uses direct addressing, where the operand is the memory address 30H. In the bottom program, the MOV instruction uses indirect addressing, where the operand is the register R0, which contains the memory address 30H. This results in the bottom program accessing the memory location at 30H, which contains the value 0x00, and displaying it as 11111111 on the 7-segment display.

Q.4. Write an 8051 Assembly Language Program to generate the last four digits of your mobile number using logical instructions (ANL, ORL, CLR). Do not directly load the complete four-digit number. The final result must be available in the Accumulator (A).

The screenshot displays the Proteus 8051 simulator interface. The central window shows the following assembly code:

```

00001 MOV A, #40
00002 ORL A, #05 ; A = 45
00004 MOV R0, A
00005 MOV A, #70
00007 ORL A, R0 ; A = 4570 (lower 4 digits)

```

The register window on the left shows the Accumulator (A) at 0x6F, R0 at 0x2D, and the Program Counter (PC) at 0x0157. The data memory window shows a table of values for addresses 00 to 70. The hardware window at the bottom shows a breadboard with a keypad, a 4-digit 7-segment display showing '8888', an ADC module, and a motor sensor.

Q.5. Write an 8051 Assembly Language Program to compact data stored in internal RAM locations 40H–5FH using only indirect addressing. Remove all occurrences of FFH, shift the remaining data left to eliminate gaps, and fill the unused locations at the end with 00H, without using additional RAM or the stack.

The screenshot displays the Proteus 8051 simulator interface. The main window shows the assembly code for an 8051 microcontroller. The code is as follows:

```

; start address
0002| MOV R0, #40H
0002| MOV R1, #40H
; store from 1

LOOP:
0004| MOV A, @R0
0005| CJNE A, #0FFH, SAVE
0008| SJMP SKIP
; if FF skip

SAVE:
000A| MOV @R1, A
000B| INC R1
; save value
; next place

SKIP:
000C| INC R0
000D| CJNE R0, #60H, LOOP
; next address

FILL:
0010| MOV A, R1
0011| CJNE A, #60H, ZERO
0014| SJMP END

```

The Data Memory window shows the internal RAM locations 40H to 5FH. The values are as follows:

Address	Value
40H	00
41H	00
42H	00
43H	00
44H	00
45H	00
46H	00
47H	00
48H	00
49H	00
4AH	00
4BH	00
4CH	00
4DH	00
4EH	00
4FH	00
50H	00
51H	00
52H	00
53H	00
54H	00
55H	00
56H	00
57H	00
58H	00
59H	00
5AH	00
5BH	00
5CH	00
5DH	00
5EH	00
5FH	00

The I/O window shows the status of various I/O devices. The UART is configured for 8-bit, No Parity, 4800 Baud. The ADC is configured for 0.0 V input and 11111111 output. The Motor is configured for Motor Enabled.