Aditi Tapariya
U21EE081

MICROCONTROLLER AND MICROPROCESSOR LAB

EXPERIMENT 4

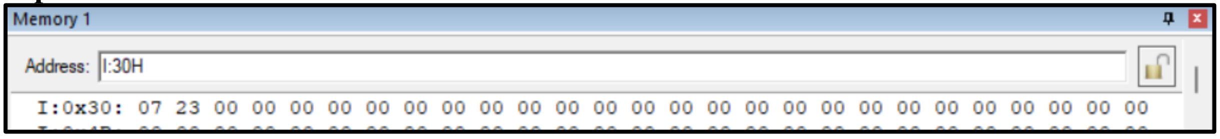
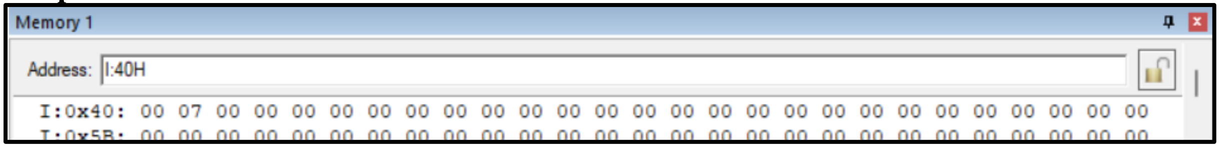
AIM: To perform division operations for 8-bit and 16-bit numbers in 8051.

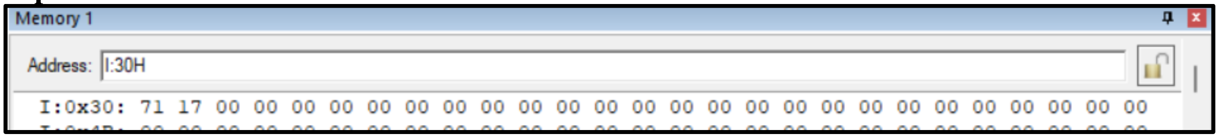
SOFTWARE USED: Keil uVision5

Question 1: Write an assembly language problem to divide two 8-bit numbers. Data 1 is stored at internal memory locations 30h and data 2 at 31h. The quotient should be stored at internal memory locations at 40h and the remainder at 41h.

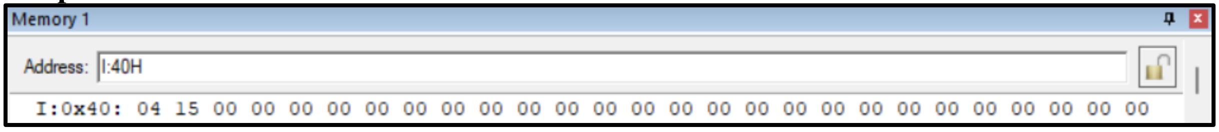
Code:
```
ORG 0000H
        MOV A,30H
        MOV B,31H
        DIV AB
        MOV 40H, A
        MOV 41H, B
        END
```

Algorithm:
1. Set up memory location ORG 0000H.
2. Initialize register A with the value 30H.
3. Initialize register B with the value 31H.
4. Divide the value in register A by the value in register B.
5. Store the quotient of the division operation in register A.
6. Store the remainder of the division operation in register B.
7. Store the quotient (result) in memory location 40H.
8. Store the remainder in memory location 41H.
9. End the program.

Result:
Case 1:
Input:

Memory 1
Address: I:30H
```
I:0x30: 07 23 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```
Output:

Memory 1
Address: I:40H
```
I:0x40: 00 07 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x5B: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Case 2:
Input:

Memory 1
Address: I:30H
```
I:0x30: 71 17 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

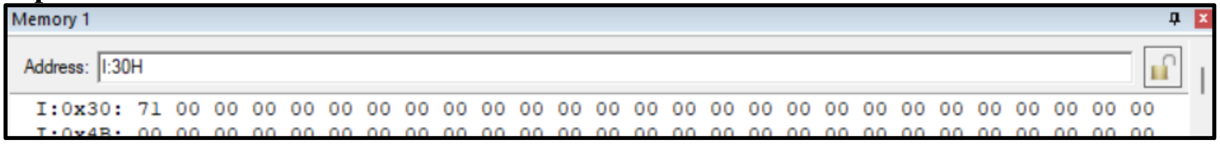
Output:



```
I:0x40: 04 15 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

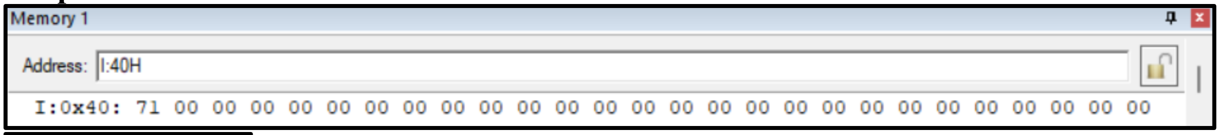
Case 3:

Input:

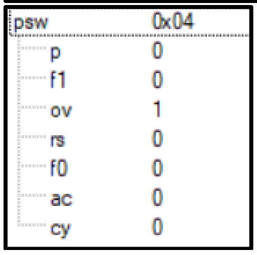


```
I:0x30: 71 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x4B: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Output:



```
I:0x40: 71 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```



```
psw   0x04
  p   0
  f1  0
  ov  1
  rs  0
  f0  0
  ac  0
  cy  0
```

Conclusion: The provided assembly code efficiently divides two 8-bit numbers stored at memory locations 30h and 31h. It handles various scenarios, including division by zero, resulting in an overflow condition, which sets the overflow flag. Additionally, it appropriately stores the quotient and remainder at memory locations 40h and 41h, respectively, ensuring accurate results for different inputs.

Question 2: Write an assembly language problem to divide two 16-bit numbers. Data 1 is stored at internal memory locations at 30h and 31h and data 2 at 32h and 33h. The quotient should be stored at internal memory locations 40hand 41h and the remainder at 42h and 43h.

Code:

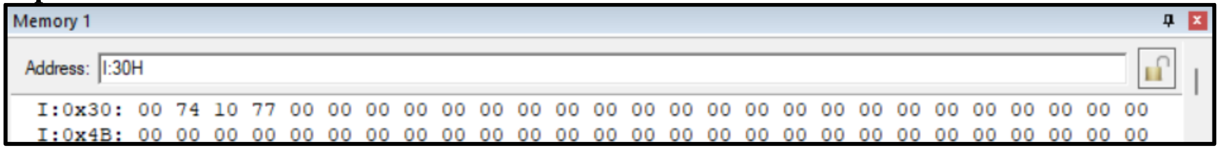
```
ORG 0000H
        MOV R0,30H
        MOV R1,31H
        MOV R2,32H
        MOV R3,33H
        MOV R4, #00H
        MOV R5, #00H
        HERE: MOV A, R1
        CLR C
        SUBB A, R3
        JC NEXT
        JNZ NEXT1
        MOV A, R0
        SUBB A, R2
        JC NEXT
        NEXT1: MOV A, R0
```

```
CLR C
SUBB A, R2
MOV R0, A
MOV A, R1
SUBB A, R3
MOV R1, A
MOV A, #01H
ADD A, R4
MOV R4, A
JNC STEP
INC R5
STEP: SJMP HERE
NEXT: MOV 40H, R4
MOV 41H, R5
MOV 42H, R0
MOV 43H, R1
END
```

Algorithm:

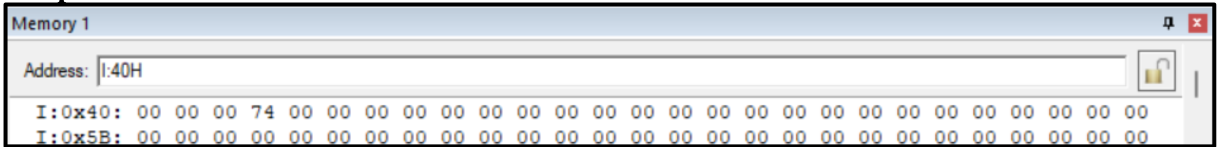
1. Set up memory location ORG 0000H.
2. Load values from memory locations 30H, 31H, 32H, and 33H into registers R0, R1, R2, and R3 respectively.
3. Initialize registers R4 and R5 to 00H.
4. Start a loop labeled HERE.
5. Subtract R3 from R1 with borrow and check carry.
6. If carry, jump to NEXT.
7. If the result not zero, jump to NEXT1.
8. Subtract R2 from R0 with borrow.
9. If carry, jump to NEXT.
10. Label NEXT1: Subtract R2 from R0 with borrow.
11. Store results in R0 and R1.
12. Increment R4.
13. If no carry, increment R5.
14. Jump back to HERE.
15. Label NEXT: Store values in R4, R5, R0, and R1 into memory locations 40H, 41H, 42H, and 43H respectively.
16. End the program.

Result:
Case 1:
Input:



```
Address: I:30H
I:0x30: 00 74 10 77 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x4B: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

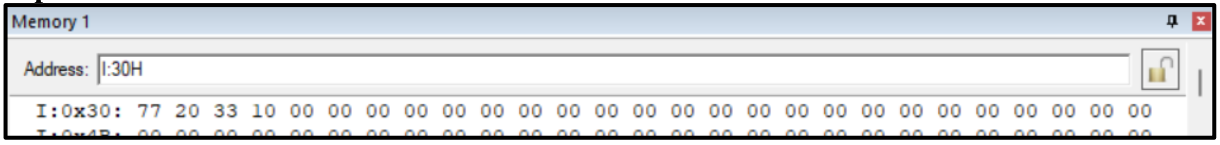
Output:



```
Address: I:40H
I:0x40: 00 00 00 74 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x5B: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Case 2:

Input:

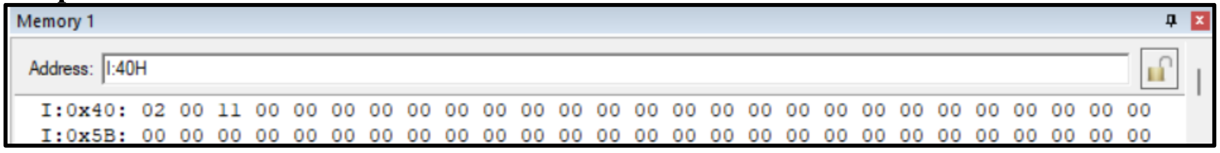


Memory 1

Address: I:30H

```
I:0x30: 77 20 33 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x4B: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Output:



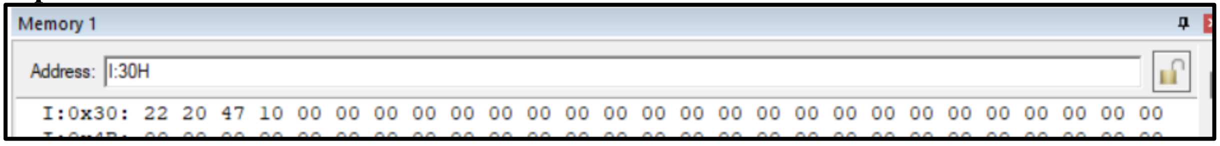
Memory 1

Address: I:40H

```
I:0x40: 02 00 11 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x5B: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Case 3:

Input:

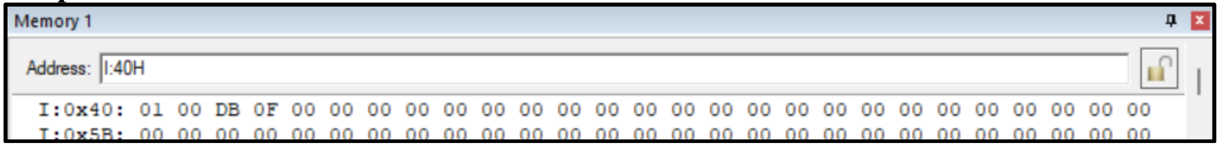


Memory 1

Address: I:30H

```
I:0x30: 22 20 47 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x4B: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Output:



Memory 1

Address: I:40H

```
I:0x40: 01 00 DB 0F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
I:0x5B: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Conclusion:

The assembly code successfully divides two 16-bit numbers stored at memory locations 30h-33h, storing the quotient at 40h-41h and the remainder at 42h-43h. It utilizes a loop to perform the division operation and handles carry conditions to ensure accurate results.