Aditi Tapariya
U21EE081

MICROCONTROLLER AND MICROPROCESSOR LAB

EXPERIMENT 2

AIM: Assembly language program for multiplication of numbers.

SOFTWARE USED: Keil uVision5

Question-1: To perform multiplication of 8 and 16 bits in 8051.
Case-1: Write an assembly language problem to multiply two 8-bit numbers(non-zeros). Data 1 is stored at an internal RAM location at 30h and data 2 is stored at 31h. The final result should be stored at internal memory locations 40h and 41h.

a) Using MUL instructions:

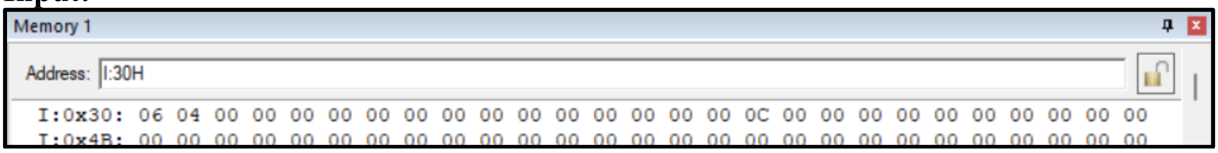
Code:

```
ORG 0000H
        MOV A,30H
        MOV B,31H
        MUL AB
        MOV 40H, A
        MOV 41H, B
        END
```

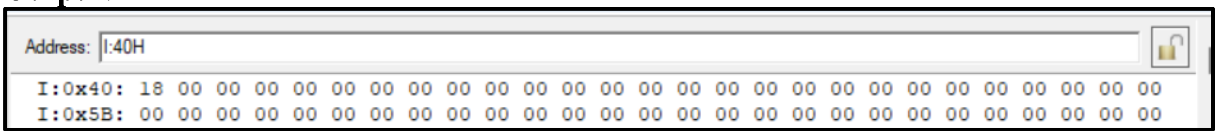
Algorithm:

1. Initialize the origin of the code to address 0000H.
2. Move the hexadecimal value 30H (48 in decimal) into register A.
3. Move the hexadecimal value 31H (49 in decimal) into register B.
4. Multiply the content of registers A and B.
5. Store the result of the multiplication in register A.
6. Move the content of register A to memory location 40H.
7. Move the content of register B to memory location 41H.
8. End the program.

Result:
Input:



Output:



Conclusion:
The assembly code accurately multiplies two 8-bit non-zero numbers using the MUL instruction, storing the result in designated memory locations, and demonstrating efficient arithmetic operations in assembly language programming.

b) Without using MUL instruction.

Code:

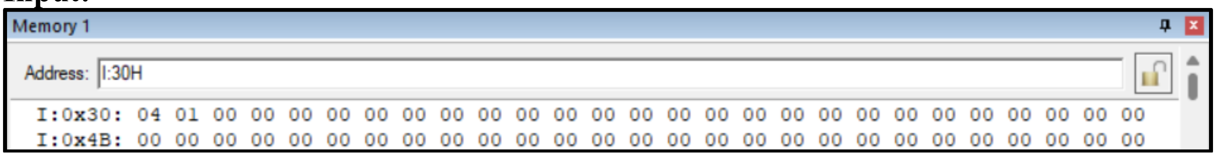
```
ORG 0000H
        MOV A, #00H
        MOV R1,30H
        MOV R2,31H //used as counter
        MOV R3, #00H // store higher byte of result
        UP: ADD A, R1
        JNC NEXT
        INC R3
        NEXT: DJNZ R2, UP
        MOV 40H, A
        MOV 41H, R3
        END
```

Algorithm:

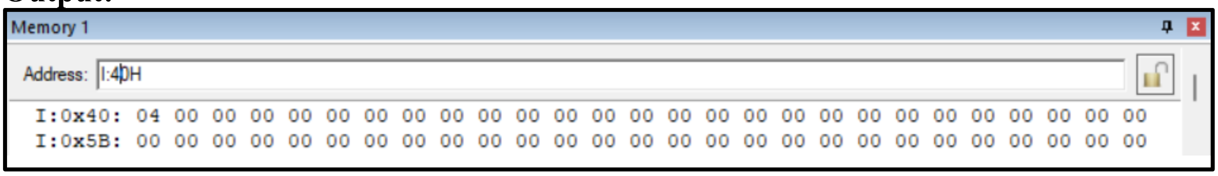
1. Initialize the origin of the code to address 0000H.
2. Move the value 00H (0 in decimal) into register A.
3. Move the hexadecimal value 30H (48 in decimal) into register R1.
4. Move the hexadecimal value 31H (49 in decimal) into register R2, which will serve as a counter.
5. Move the value 00H (0 in decimal) into register R3, which will store the higher byte of the result.
6. Start a loop labeled as "UP."
7. Add the content of register R1 to register A, storing the result in register A.
8. Check the carry flag (CF). If CF is not set, proceed to the "NEXT" step.
9. Increment the content of register R3 to account for the carry in the higher byte.
10. Label the next step as "NEXT."
11. Decrement the content of register R2 (counter) by 1 using the DJNZ (Decrement Jump if Not Zero) instruction. If R2 is not zero, jump back to the "UP" step; otherwise, continue to the next instruction.
12. Move the content of register A to memory location 40H.
13. Move the content of register R3 to memory location 41H.
14. End the program.

Result:

Input:



Output:



Conclusion:

The assembly code effectively multiplies two 8-bit non-zero numbers without using the MUL instruction, employing addition and carry flag handling, ensuring accurate multiplication and storage of the result.

Case-2: Write an assembly language problem to multiply 16-bit data stored at the internal RAM location 30h and 31h and 8-bit data stored at memory location 32h.The final result should be stored at the internal RAM locations 40h,41h, and 42h.

Code:

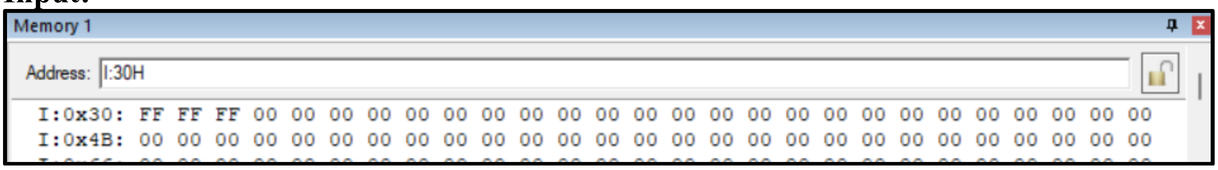
```
ORG 0000H
        MOV A,30H
        MOV B,32H
        MUL AB
        MOV 40H, A
        MOV R1, B
        MOV A,31H
        MOV B,32H
        MUL AB
        ADD A, R1
        JNC NEXT
        INC B
        NEXT: MOV 41H, A
        MOV 42H, B
        END
```

Algorithm:

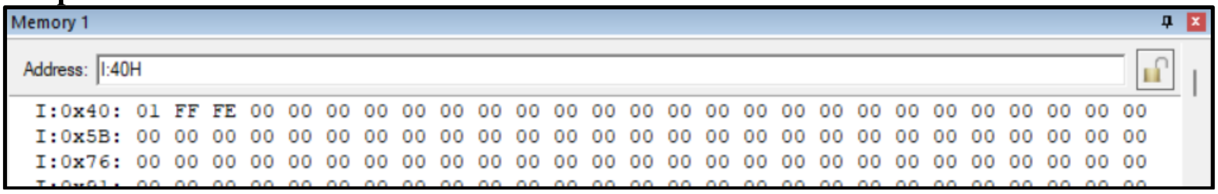
1. Initialize the origin of the code to address 0000H.
2. Move the hexadecimal value 30H (48 in decimal) into register A.
3. Move the hexadecimal value 32H (50 in decimal) into register B.
4. Multiply the content of registers A and B.
5. Store the lower byte of the result in register A.
6. Move the content of register B to register R1 for later use.
7. Move the hexadecimal value 31H (49 in decimal) into register A.
8. Move the hexadecimal value 32H (50 in decimal) into register B.
9. Multiply the content of registers A and B.
10. Add the result of the multiplication to register A.
11. Check the carry flag (CF). If CF is not set, proceed to the "NEXT" step.
12. Increment the content of register B to account for the carry.
13. Label the next step as "NEXT."
14. Move the content of register A to memory location 41H.
15. Move the content of register B to memory location 42H.
16. End the program.

Result:

Input:



Output:



Conclusion:
The assembly code efficiently multiplies a 16-bit number stored across two registers and an 8-bit number, handling carry and ensuring accurate multiplication, storing the result in designated memory locations.

Case-3: Write an assembly language problem to find the square of a 16-bit number. The data is stored in an internal memory location 30h and 31h. Store the final result at the internal RAM location 40h,41h,42h, and 43h.

Code:
```
ORG 0000H
        MOV A,30H
        MOV B,30H
        MUL AB
        MOV 40H, A
        MOV R1, B
        MOV A,31H
        MOV B,30H
        MUL AB
        ADD A, R1
        MOV R2, A
        JNC NEXT
        INC B
        NEXT: MOV R3, B
        MOV A,30H
        MOV B,31H
        MUL AB
        ADD A, R2
        MOV 41H, A
        MOV A, B
        ADDC A, R3
        MOV R4, A
        JNC NEXT1
        INC R5
        NEXT1: MOV A,31H
        MOV B,31H
        MUL AB
        ADD A, R4
        MOV 42H, A
        MOV A, B
        ADDC A, R5
        MOV 43H, A
        END
```

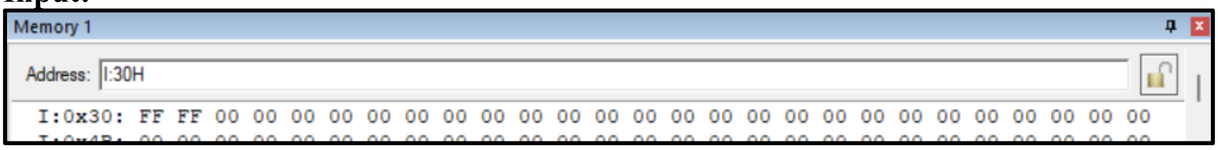
Aditi Tapariya
U21EE081

Algorithm:

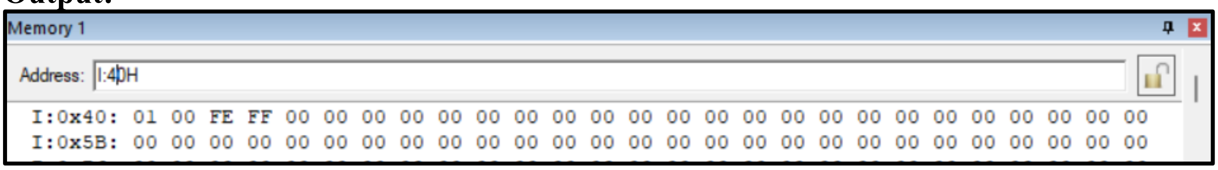
1. Initialize the origin of the code to address 0000H.
2. Load the hexadecimal value 30H (48 in decimal) into register A.
3. Load the hexadecimal value 30H (48 in decimal) into register B.
4. Multiply the content of registers A and B.
5. Store the lower byte of the result in register A and the upper byte in register B.
6. Move the content of register A to memory location 40H.
7. Move the content of register B to register R1 for later use.
8. Load the hexadecimal value 31H (49 in decimal) into register A.
9. Load the hexadecimal value 30H (48 in decimal) into register B.
10. Multiply the content of registers A and B.
11. Add the result of the multiplication to register A.
12. Move the content of register A to register R2.
13. Check the carry flag (CF). If CF is not set, proceed to the "NEXT" step.
14. Increment the content of register B to account for the carry.
15. Label the next step as "NEXT."
16. Move the content of register B to register R3.
17. Load the hexadecimal value 30H (48 in decimal) into register A.
18. Load the hexadecimal value 31H (49 in decimal) into register B.
19. Multiply the content of registers A and B.
20. Add the result of the multiplication to register A.
21. Move the content of register A to memory location 41H.
22. Move the content of register B to register A.
23. Add with carry the content of register R3 to register A.
24. Move the result to register R4.
25. Check the carry flag (CF). If CF is not set, proceed to the "NEXT1" step.
26. Increment register R5 to account for the carry.
27. Label the next step as "NEXT1."
28. Load the hexadecimal value 31H (49 in decimal) into register A.
29. Load the hexadecimal value 31H (49 in decimal) into register B.
30. Multiply the content of registers A and B.
31. Add the result of the multiplication to register A.
32. Move the content of register A to memory location 42H.
33. Move the content of register B to register A.
34. Add with carry the content of register R5 to register A.
35. Move the result to memory location 43H.
36. End the program.

Result:

Input:



Output:



Conclusion:

The assembly code efficiently computes the square of a 16-bit number, utilizing multiplication and carry handling, storing the result across designated memory locations, and demonstrating effective arithmetic operations.