Aditi Tapariya
U21EE081

MICROCONTROLLER AND MICROPROCESSOR LAB

EXPERIMENT 6

AIM: To perform conversion from hexadecimal to decimal and vice-versa.

SOFTWARE USED: Keil uVision5

Question-1: Write an ALP for 8051 microcontrollers to convert a 2-digit decimal value stored in a 50h location to a hexadecimal value. Store the result in 53h.

Code:

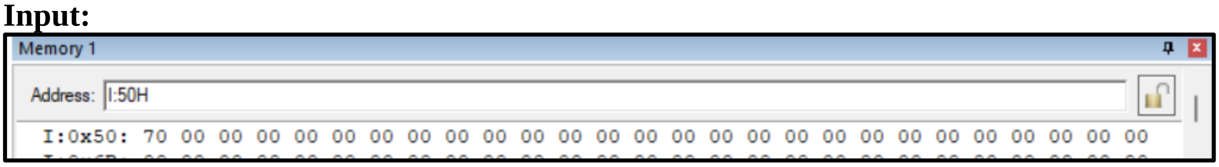
```
ORG 0000H
        MOV A,50H
        SWAP A
        ANL A, #0FH
        MOV B, #0AH
        MUL AB
        MOV R0, A
        MOV A,50H
        ANL A, #0FH
        ADD A, R0
        MOV 53H, A
        END
```

Algorithm:

1. Load decimal value from memory 50h to accumulator.
2. Swap accumulator nibbles to isolate tens and units digits.
3. Clear the upper nibble of the accumulator.
4. Load 0AH (10 decimal) into register B.
5. Multiply the accumulator by register B.
6. Move result (tens * 10) to register R0.
7. Reload decimal value from memory 50h.
8. Clear the upper nibble of the accumulator.
9. Add R0 (tens * 10) to the units' digit.
10. Store the final result in memory location 53h.

Result:

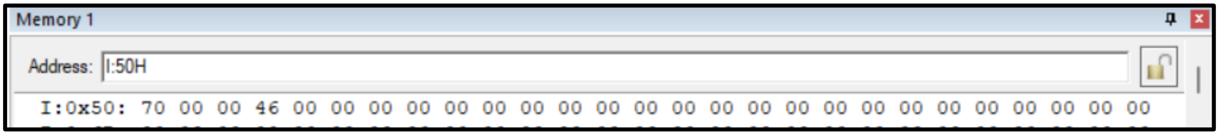
Input:



Address: I:50H

```
I:0x50: 70 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:50H

```
I:0x50: 70 00 00 46 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 provided assembly code efficiently converts a 2-digit decimal value to hexadecimal on an 8051 microcontroller, storing the result in memory, and facilitating numeric conversion operations.

Question-2: Write an ALP for 8051 micro-controllers to convert a 2-digit hexadecimal value stored in a 50h location to a decimal value. Store the result in 53h and 54h.

Code:

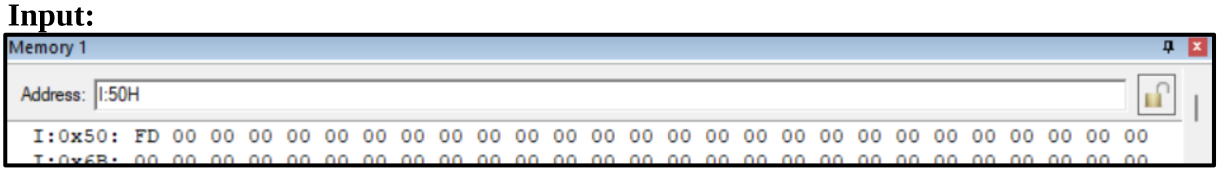
```
 ORG 0000H
        MOV A,50H
        MOV B, #64H
        DIV AB
        MOV 53H, A
        MOV A, B
        MOV B, #0AH
        DIV AB
        SWAP A
        ADD A, B
        MOV 54H, A
        END
```

Algorithm:

1. Load the hexadecimal value from memory location 50h into accumulator A.
2. Load the divisor (100 decimal) into register B.
3. Divide accumulator A by register B.
4. Store the quotient (tens digit) from accumulator A into memory location 53h.
5. Reload accumulator A with the remainder (units digit).
6. Load the divisor (10 decimal) into register B.
7. Divide accumulator A by register B.
8. Swap accumulator A to separate the tens and unit digits.
9. Add the unit digit (after swapping) to the tens digit.
10. Store the final decimal result (tens and units digits combined) in memory locations 53h and 54h.

Result:

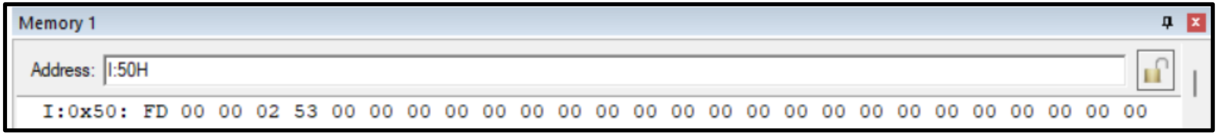
Input:



Address: I:50H

```
I:0x50: FD 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:0x6B: 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:50H

```
I:0x50: FD 00 00 02 53 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Conclusion:

The provided assembly code effectively converts a 2-digit hexadecimal value to decimal on an 8051 microcontroller, storing the result in memory, and facilitating numeric conversion operations.

<u>Question-3</u>: Write an ALP for 8051 micro-controllers to convert a 1-digit hexadecimal value stored in a 50h location to ASCII code. Store the result in 53h.

Code:

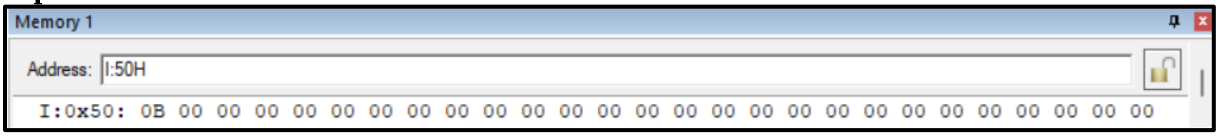
```
ORG 0000H
        MOV A,50H
        CLR C
        SUBB A, #0AH
        JC HERE
        ADD A, #41H
        SJMP RESULT
        HERE: MOV A,50H
        ADD A, #30H
        RESULT: MOV 54H, A
        END
```

Algorithm:

1. Load the 1-digit hexadecimal value from memory location 50h into accumulator A.
2. Clear the carry flag.
3. Subtract 0AH from accumulator A.
4. Check if the carry flag is set (indicates a borrow).
    a. If not, add 41H to accumulator A (convert to ASCII).
    b. If yes, skip conversion and proceed directly to storing the result.
5. Store the result (ASCII code) from accumulator A into memory location 53h.
6. If the carry flag was set, reload the original value from memory 50h into accumulator A.
7. Add 30H to accumulator A to convert it to ASCII.
8. Store the result (ASCII code) from accumulator A into memory location 54h.

Result:

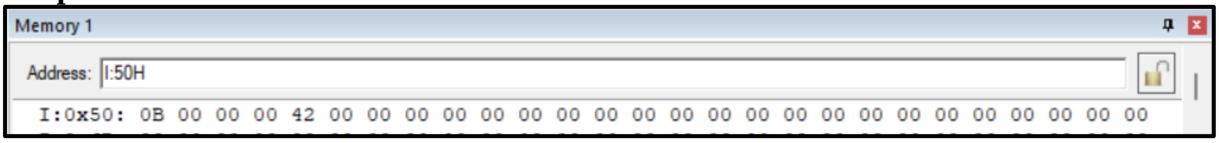
Input:


Memory 1

Address: I:50H

```
I:0x50: 0B 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:50H

```
I:0x50: 0B 00 00 00 42 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Conclusion:

The provided assembly code converts a 1-digit hexadecimal value to its corresponding ASCII code, facilitating character representation on an 8051 microcontroller, stored in designated memory locations.