Aditi Tapariya
U21EE081

MICROCONTROLLER AND MICROPROCESSOR LAB

EXPERIMENT 5

AIM: To perform memory block movement for forward, reverse, and overlapping cases in 8051.

SOFTWARE USED: Keil uVision5

Question-1: Write in 8051 Microcontroller to move a block from memory from the internal memory address 40h-49h to 50h-59h in the forward order.

Code:

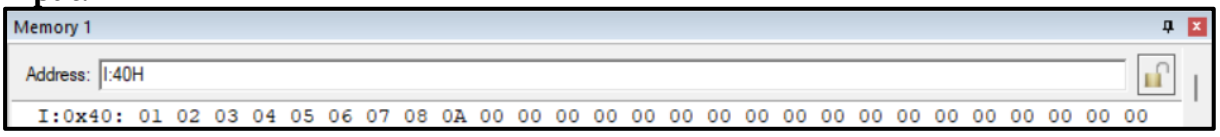
```
ORG 0000H
        MOV R0, #40H
        MOV R1, #50H
        MOV R2, #0AH
        LOC1: MOV A, @R0
        MOV @R1, A
        INC R0
        INC R1
        DJNZ R2, LOC1
        END
```

Algorithm:

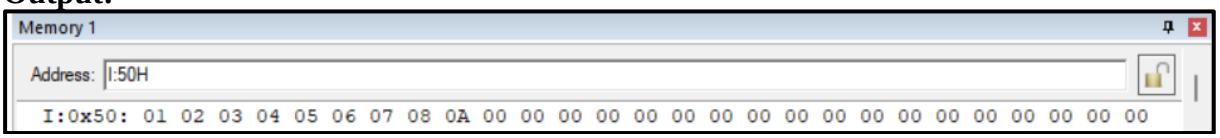
1. Set up memory location ORG 0000H.
2. Load immediate value 40H into register R0.
3. Load immediate value 50H into register R1.
4. Load immediate value 0AH into register R2.
5. Start a loop labeled LOC1.
6. Load the byte from the memory address pointed to by R0 into register A.
7. Store the content of register A into the memory address pointed to by R1.
8. Increment the value in register R0.
9. Increment the value in register R1.
10. Decrement register R2 and jump back to LOC1 if it's not zero.
11. End the program.

Result:

Input:



Output:



Conclusion:

The 8051-microcontroller code moves a block of data from memory addresses 40h-49h to 50h-59h in a forward order. It iterates through each byte using registers R0, R1, and R2, copying data from source to destination. This concise algorithm provides an efficient solution for data transfer within the microcontroller's memory.

Question-2: Write in 8051 Microcontroller to move a block from memory from the internal memory address 40h-49h to 50h-59h in the reverse order.

Code:

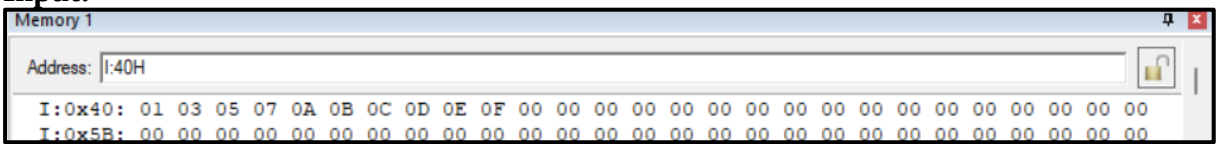
```
ORG 0000H
   MOV R0, #49H
   MOV R1, #59H
   MOV R2, #0AH

   LOC1: MOV A, @R0
        MOV @R1, A
        DEC R0
        DEC R1
        DJNZ R2, LOC1
   END
```

Algorithm:

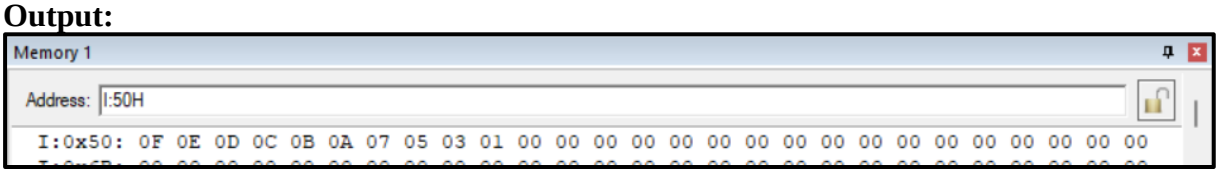
1. Set up memory location ORG 0000H.
2. Load immediate value 49H into register R0 to point to the last byte of the source block.
3. Load immediate value 50H into register R1 to point to the first byte of the destination block.
4. Load immediate value 0AH into register R2 to set the loop counter to 10 bytes.
5. Start a loop labeled LOC1.
6. Load the byte from the memory address pointed to by R0 into register A.
7. Store the byte at the memory address pointed to by R1.
8. Decrement the source address pointer (R0).
9. Increment the destination address pointer (R1).
10. Decrement the loop counter (R2).
11. Continue the loop until R2 becomes zero.
12. End the program.

Result:

Input:



Output:



Conclusion:

This 8051-microcontroller code moves a block of data from memory addresses 40h-49h to addresses 50h-59h in reverse order. It iterates through each byte using registers R0, R1, and R2, copying data from source to destination.

Question-3: Write in 8051 Microcontroller to move a block from memory from the internal memory address 40h-49h to 45h-4Eh in the overlapping order.

Code:

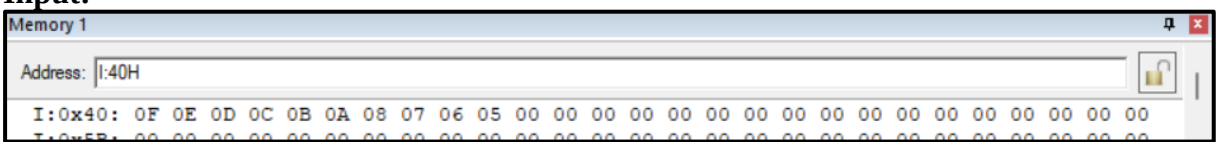
```
ORG 0000H
   MOV R0, #49H
   MOV R1, #4EH
   MOV R2, #0AH

   LOC1: MOV A, @R0
       MOV @R1, A
       DEC R0
       DEC R1
       DJNZ R2, LOC1
   END
```

Algorithm:

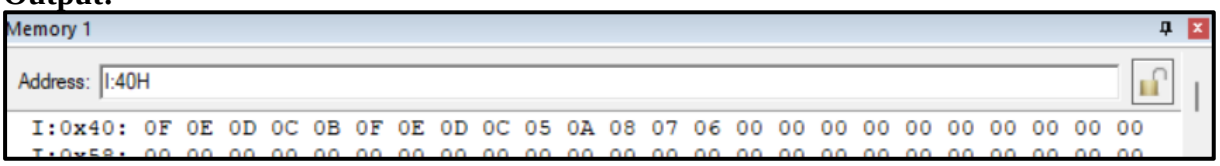
1. Set up memory location ORG 0000H.
2. Load immediate value 49H into register R0 to point to the last byte of the source block.
3. Load immediate value 4EH into register R1 to point to the last byte of the destination block.
4. Load immediate value 0AH into register R2 to set the loop counter to 10 bytes.
5. Start a loop labeled LOC1.
6. Load the byte from the memory address pointed to by R0 into register A.
7. Store the byte at the memory address pointed to by R1.
8. Decrement the source address pointer (R0).
9. Decrement the destination address pointer (R1).
10. Decrement the loop counter (R2).
11. Continue the loop until R2 becomes zero.
12. End the program.

Result:

Input:



Output:



Conclusion:

The provided 8051 microcontroller code efficiently moves a block of data from memory addresses 49h-40h to addresses 4Eh-45h in an overlapping order. It utilizes R0, R1, and R2 registers to manage the memory addresses and loop control. This algorithm ensures that each byte is correctly copied, providing an effective solution for data transfer within the microcontroller's memory.

Question-4: Write an assembly language program to find the sum of an array of 10 bytes stored in internal memory address 40h to 49h.

Code:

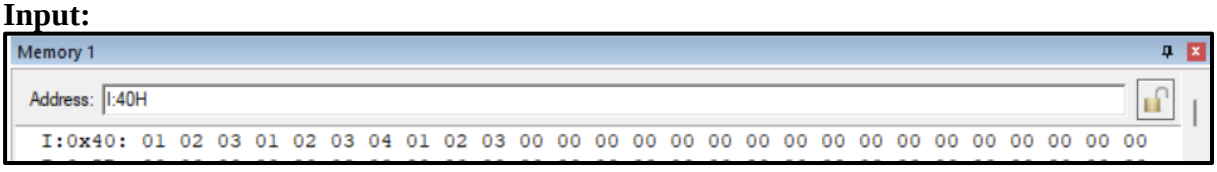
```
ORG 0000H
        MOV R0, #40H
        MOV 51, #00H
        MOV 50, #00H
        MOV R1, #0AH
        DO1: MOV A,51H
        ADD A, @R0
   MOV 51H, A
        INC R0
        JNC     DO
        MOV A,50H
        ADD A, #01H
        MOV 50H, A
        DO:
        DJNZ R1, DO1
        END
```

Algorithm:

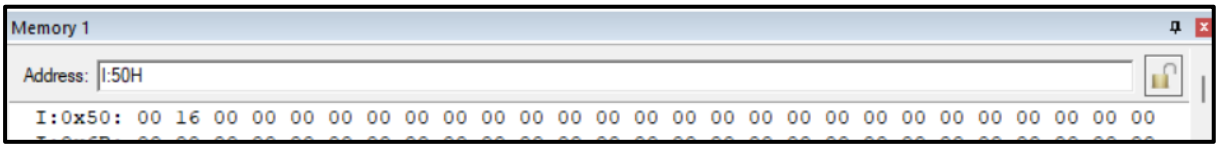
1. Initialize R0 with 40H.
2. Initialize memory addresses 51 and 50 with 00H.
3. Initialize R1 with 0AH.
4. Start a loop DO1:
    a. Read the value from memory address 51 into accumulator A.
    b. Add the value at the address pointed by R0 to A.
    c. Store the result back into memory address 51.
    d. Increment R0.
    e. If no carry, jump to DO.
5. If carry, increment the value at memory address 50.
6. Continue the loop DO until R1 becomes zero.
7. End the program.

Result:

Input:



Output:



Conclusion:
The assembly program efficiently calculates the sum of a 10-byte array stored in memory. It demonstrates fundamental concepts such as looping, arithmetic operations, and conditional branching in assembly language programming.

Question-5: Write an assembly language program in 8051 to arrange 10 bytes of hexadecimal numbers in ascending order starting from the internal memory location 20h.

Code:

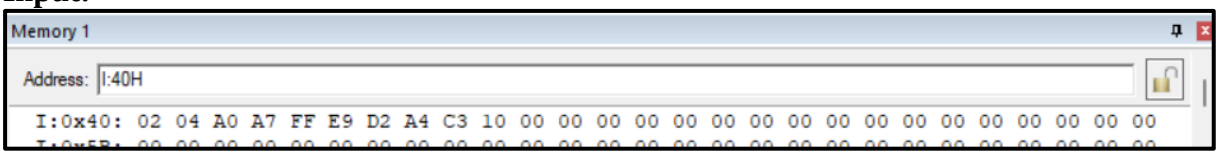
```
ORG 0000H
        MOV R4, #09H
        LOOP1: MOV R3, #09H
        MOV R0, #40H
        LOOP2: MOV A, @R0
        MOV R1, A
        INC R0
        MOV A, @R0
        SUBB A, R1
        JNC NEXT
        MOV A, @R0
        DEC R0
        MOV @R0, A
        MOV A, R1
        INC R0
        MOV @R0, A
        NEXT: DJNZ R3, LOOP2
        DJNZ R4, LOOP1
        END
```

Algorithm:

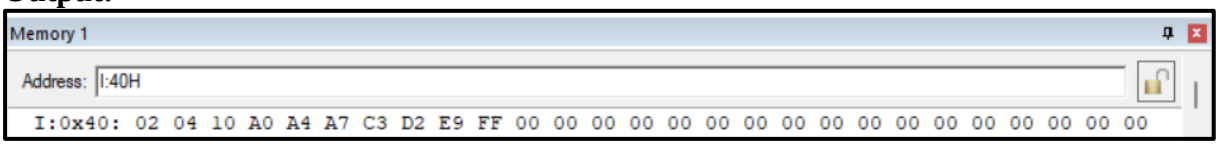
1. Initialize R4 with 09H.
2. Start a loop LOOP1:
    a. Initialize R3 with 09H.
3. Start another loop LOOP2:
    a. Load byte at R0 into A.
    b. Move A to R1.
    c. Increment R0.
    d. Load the next byte into A.
    e. Subtract R1 from A.
    f. If no carry, jump to NEXT.
    g. Load the current byte into A.
    h. Decrement R0.
    i. Move A to memory at R0.
    j. Move R1 to A.
    k. Increment R0.

    l. Move A to the next memory.
4. Decrement R3 and repeat LOOP2 until R3 is zero.
5. Decrement R4 and repeat LOOP1 until R4 is zero.
6. End the program.

Result:
Input:



Output:



Conclusion: The provided assembly program sorts 10 bytes of hexadecimal numbers in ascending order using the bubble sort algorithm. It demonstrates iterative comparison and swapping of elements to arrange them sequentially in memory.

Question-6: Write an assembly language program in 8051 to arrange 10 bytes of hexadecimal numbers in descending order starting from the internal memory location 20h.

Code:

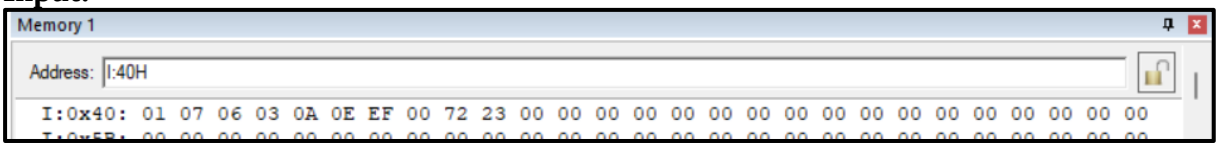
```
ORG 0000H
        MOV R4, #09H
        LOOP1: MOV R3, #09H
        MOV R0, #40H
        LOOP2: MOV A, @R0
        MOV R1, A
        INC R0
        MOV A, @R0
        SUBB A, R1
        JC NEXT
        MOV A, @R0
        DEC R0
        MOV @R0, A
        MOV A, R1
        INC R0
        MOV @R0, A
        NEXT: DJNZ R3, LOOP2
        DJNZ R4, LOOP1
        END
```

Algorithm:

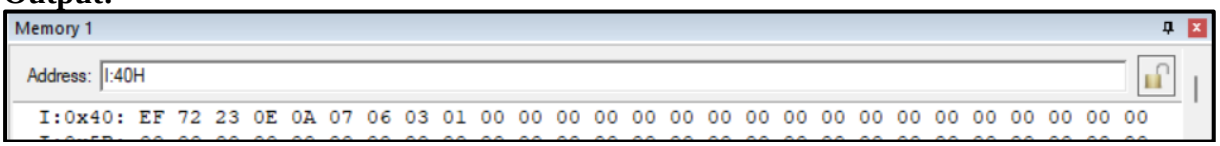
1. Initialize R4 with 09H.
2. Start a loop LOOP1:
    a. Initialize R3 with 09H.
3. Start another loop LOOP2:

   a. Load byte at R0 into A.
   b. Move A to R1.
   c. Increment R0.
   d. Load the next byte into A.
   e. Subtract R1 from A.
   f. If carry, jump to NEXT.
   g. Load the current byte into A.
   h. Decrement R0.
   i. Move A to memory at R0.
   j. Move R1 to A.
   k. Increment R0.
   l. Move A to the next memory.
4. Decrement R3 and repeat LOOP2 until R3 is zero.
5. Decrement R4 and repeat LOOP1 until R4 is zero.
6. End the program.

Result:
Input:



Output:



Conclusion:
The provided assembly program sorts 10 bytes of hexadecimal numbers in descending order using the bubble sort algorithm. It iteratively compares and swaps elements to arrange them in reverse sequential order in memory.

Question-7: Write an assembly language program in 8051 to find the factorial of a number less than 5h stored in the internal memory location 40h and store the result in 45h.

Code:

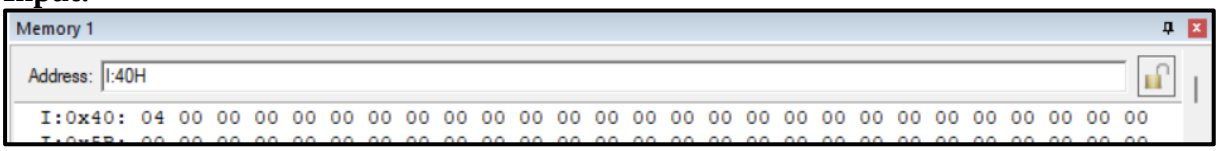
```
ORG 0000H
        MOV R0,40H
        MOV A, R0
        DO: DEC R0
        CJNE R0, #01H, LOC1
        SJMP LOC2
        LOC1: MOV B, R0
        MUL AB
        SJMP DO
        LOC2: MOV 45H, A
        END
```

Algorithm:

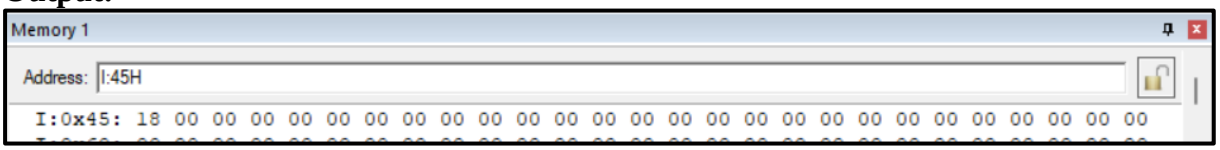
1. Initialize R0 with 40H.
2. Move R0 to A.
3. Start a loop DO:
    a. Decrement R0.
    b. If R0 is not 01H, jump to LOC1.
    c. If R0 is 01H, jump to LOC2.
4. At LOC1:
    a. Move R0 to B.
    b. Multiply A and B.
    c. Go back to the DO loop.
5. At LOC2:
    a. Move A to memory address 45H.
6. End the program.

Result:

Input:



Output:



Conclusion:

The provided assembly program calculates the factorial of a number less than 5h stored in memory. It effectively demonstrates looping, decrementing, conditional branching, and multiplication operations to compute the factorial and stores the result in memory.