Ex No:1	Assembly Language Programs
Date:	

Aim:

The purpose of this experiment is to write and implement 8051 assembly language experiments using simulator.

Software Required:

Edsim51 simulator

Program Logic:

An Assembly language program consists of, among other things, a series of line of Assemble language instructions. An Assembly language instruction consists of mnemonic, optionally followed by one or two operands. The operands are the data items being manipulated and the mnemonics are the commands to the CPU, telling it what to do with those items.

1. **Looping:**

Repeating a sequence of instructions a certain number of time is called a loop. The loop is one of the most widely used actions that any microprocessor performs. In this instruction, the register is decremented; if it is not zero, it jumps to the target address referred by the label. Prior to the start of the loop the register is loaded with the counter for the number of repetitions. In this instruction both the register decrement and the decision to jump are combined into a single instruction.

A loop inside another loop is called nested loop.

2. **Unconditional jump instruction:**

The unconditional jump is a jump in which control is transferred unconditionally to the target location. In 8051 there is two unconditional jumps:

- **LJMP(long jump):**

LJMP is an unconditional long jump. It is a 3-byte instruction in which the first byte is a opcode, second and third bytes represent the 16-bit address of the target location. The 2-byte target address allows a jump to any memory location from 0000 to FFFFH.

- **SYMP(Short jump):**

SJMP is an unconditional short jump. It is a 2-byte instruction in which the first byte is a opcode and second bye is the relative address of the target location. The target address ranges from 00 to FFH.

Procedure:

Step-1: Open the Edsim51 simulator.

Step-2: Type the program and save it with ".asm" or ".hex" extension.

Step-3: Run the program and rectify the result.

Step-4: Observe the output.

Step-5: Repeate the above steps for all the program.

Program:

1.Sample of Assembly language program:

```
        ORG 0H              ;start (origin) at location 0
        MOV R5, #25H        ;load 25H into R5
        MOV R7,#34H         ;load 34H into R7
        MOV A, #0           ;load0 into A
        ADD A, R5           ;add contents of R5 to A
                            ;now A = A + R5
        ADD A, R7           ;add contents of R7 to A
                            ;now A = A + R7
        ADD A, #12H         ;add to A value 12H
                            ;now A = A + 12H
HERE:   SJMP HERE           ;Stay in this loop
        END                 ;end of asm source file
```

2.Multipy 25 by 10 using the technique of repeated addition:

```
        MOV A, #0           ;A=0, clear ACC
        MOV R2, #10         ;the multiplier is placed in R2
AGAIN:  ADD A, #25          ;add the multiplicand to the ACC
        DJNZ R2, AGAIN      ;repeat until R2=0 (10 times)
        MOV R5,A            ;save A in R5 ;R5-FAH
```

3.Program to add first 10 natural number:

```
        MOV A, 0            ;A0,clear Acc
        MOV R2, #10         ;load counter value in R2
        MOV R0, #0          ;initialize R0 to zero
AGAIN:  INC R0              ;increment R0 to hold the natural numbers
        ADD A,R0            ;add first number to ACC
        DJNZ R2, AGAIN      ;repeat until R2=0(10 times)
        MOV 46H, A          ;save the result (37H) in RAM location 46H
```

4.Program to load the accumulator with the value of 55H and complement the ACC 700 times:

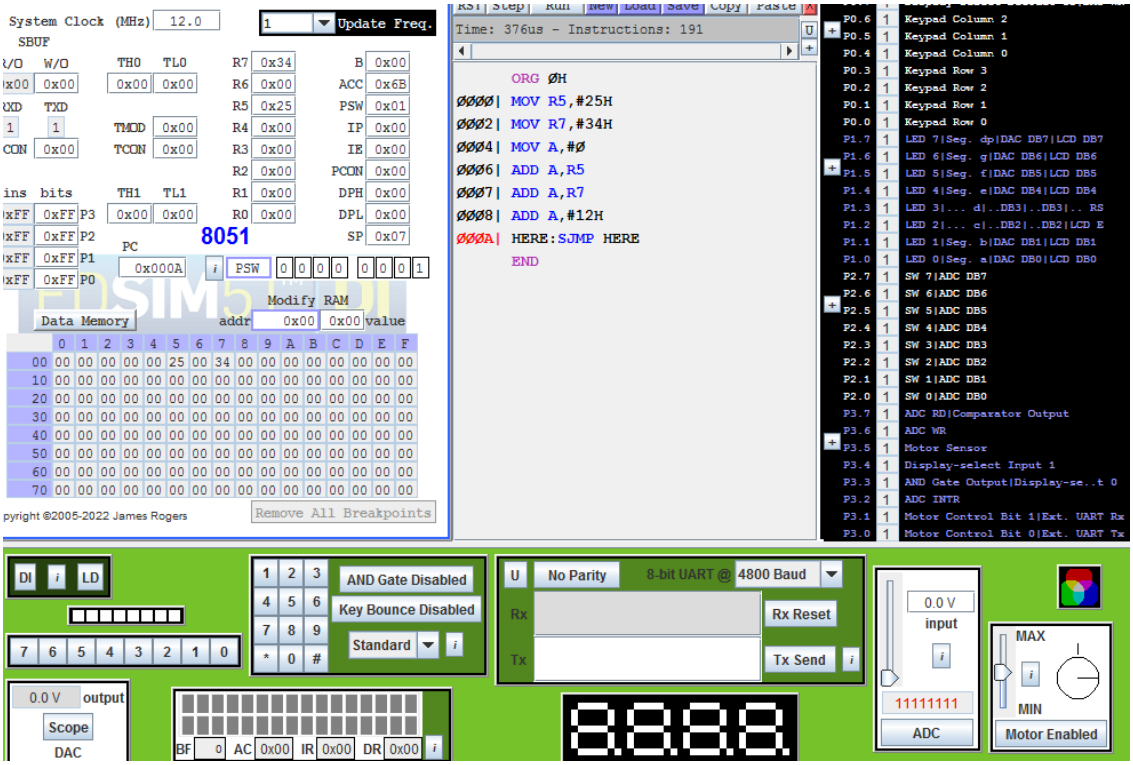
```
        MOV A,#55H          ;A=55H
        MOV R3,#10          ;R3=10, the outer loop count
NEXT:   MOV R2,#70          ;R2=70,the inner loop count
AGAIN:  CPL A               ;complement A register
        DJNZ R2,AGAIN       ;repeat it 70 times(inner loop)
        DJNZ R3,NEXT
```

5.Multiply ECH by 25H using the technique of repeated addition:

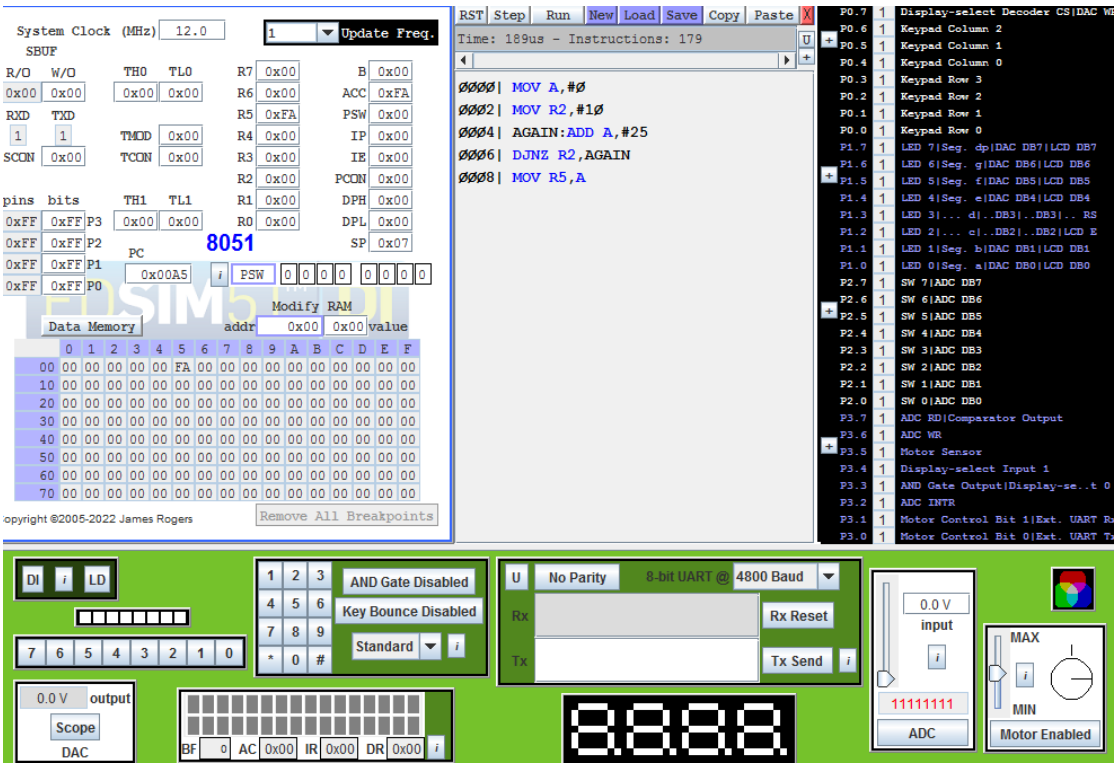
```
        MOV R1,#0          ;R1=0,this 1s the register to store the MSB
        MOV A,#0           ;clear ACC
        MOV R0,#25H        ;the multiplier is placed in R0
AGAIN:  ADD A,#0ECH        ;add the multiplicand to the Acc
        JNC HERE           ;if no carry, then repeat the addition
        INC R1             ;increment R1 for each carry generated
HERE:   DJNZ R0,AGAIN      ;repeat until R0=0
        MOV R0,A           ;the LSB of the product is moved to R0
                           ;the MSB of the product is in R1
                           ; now R1=22H and R0=1CH
```

Output:

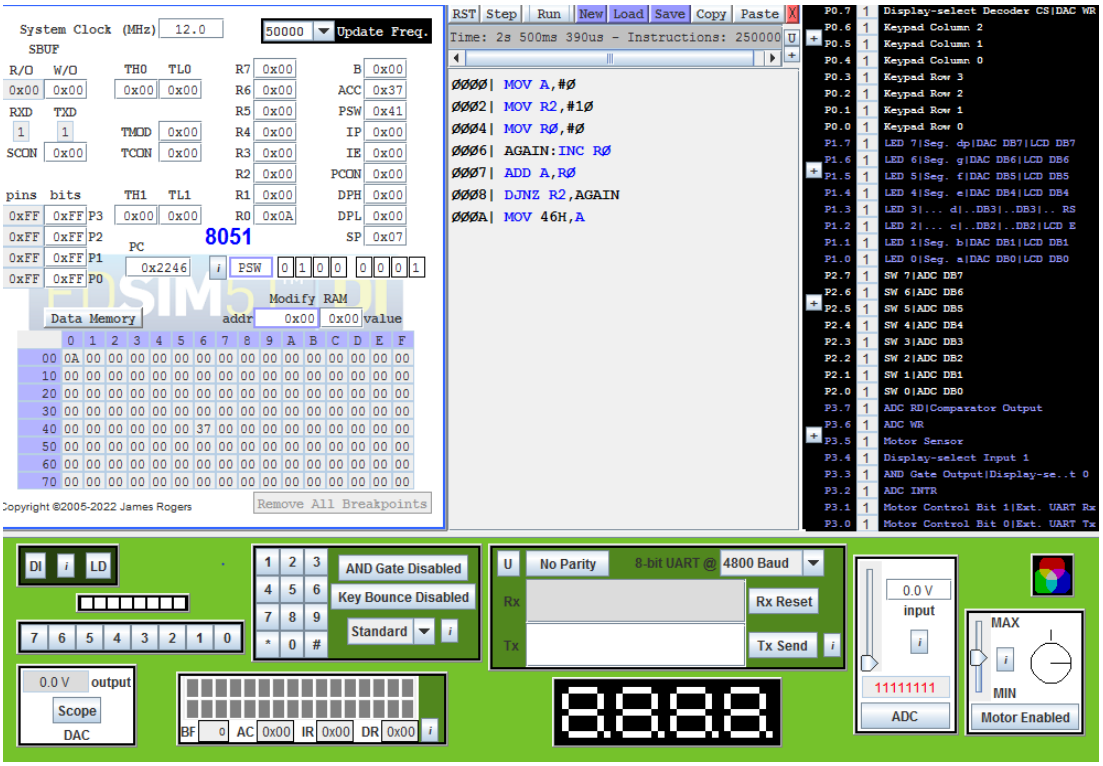
1.Sample of Assembly language program:



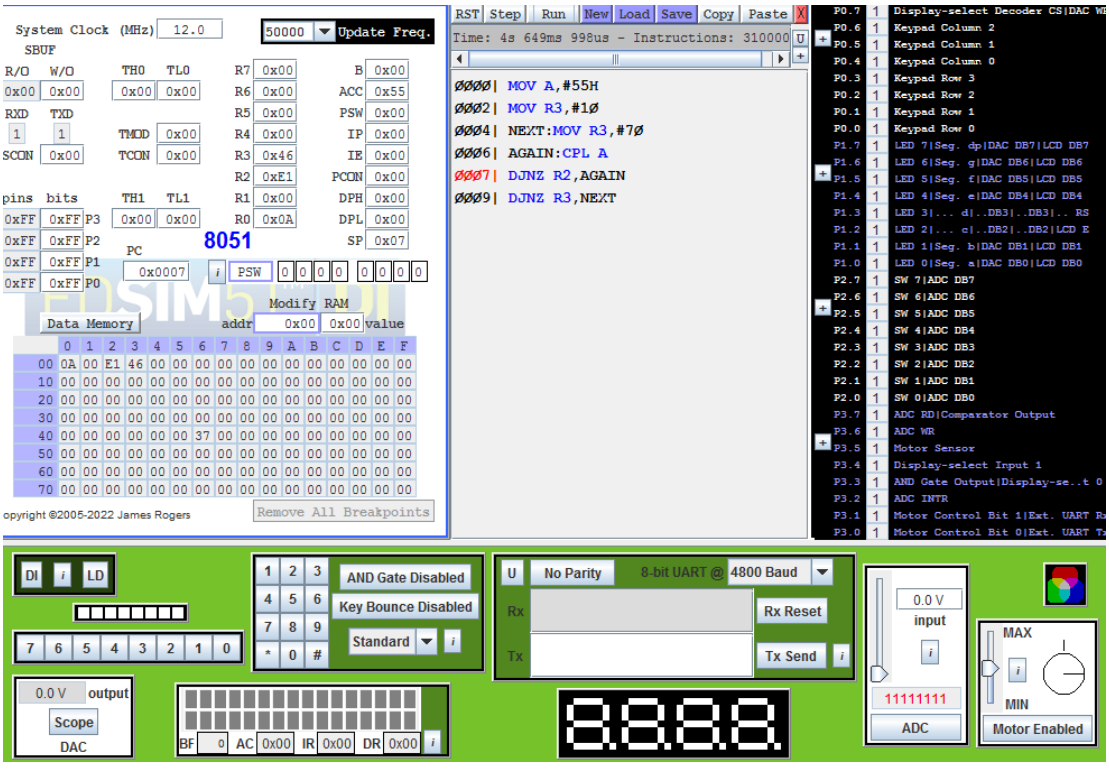
2.Multipy 25 by 10 using the technique of repeated addition:



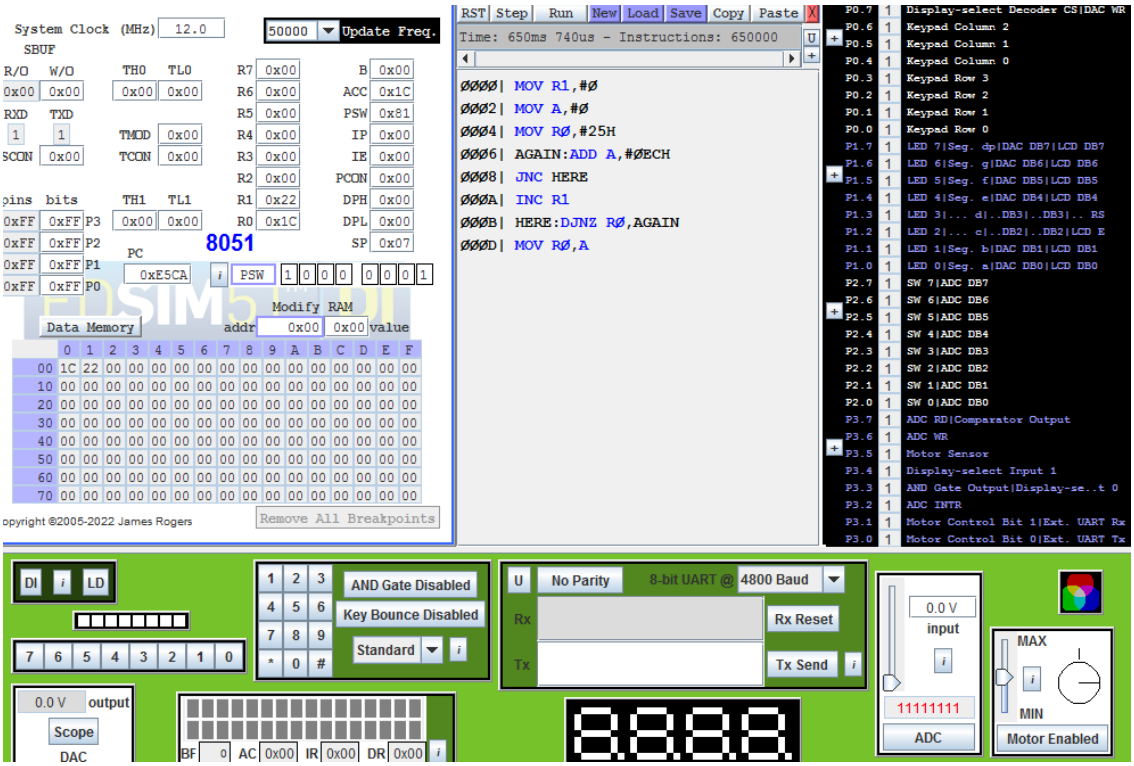
3.Program to add first 10 natural number:



4.Program to load the accumulator with the value of 55H and complement the ACC 700 times:



5.Multiply ECH by 25H using the technique of repeated addition:



Result:

Thus the 8051 assembly level program was written and implemented successfully

Ex No:2	To Text Data Transfer Between Register and Memory
Date:	

Aim:

The purpose of this experiment is to text data transfer between register and memory.

Software Required:

Edsim51 simulator.

Program Logic:

In 8051, the action of comparing and jumping are combined into single instruction called CJNE. The CJNE instruction compares two operands and jump if they are not equal. In addition, it changes the CY flag to indicate if the destination operand is larger or smaller. It is important to notice that the operand themselves remains unchanged.

Procedure:

Step-1: Enter the opcode in Edsim51 simulator.

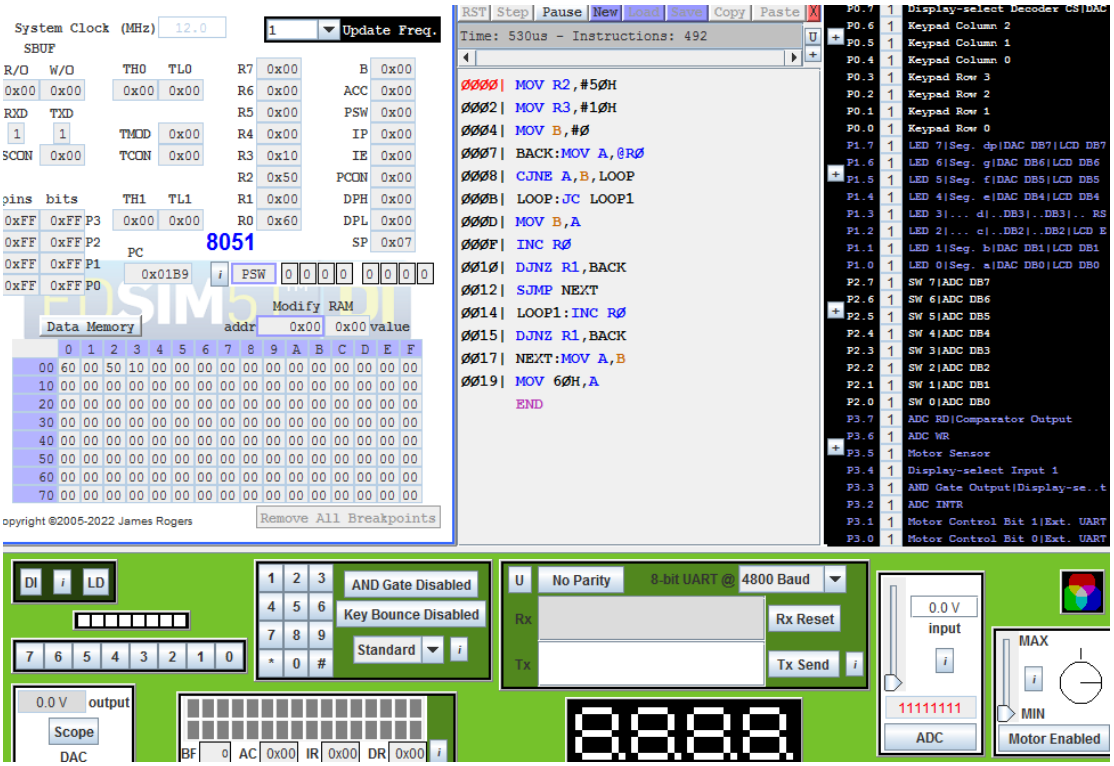
Step-2: Execute the program.

Step-3: Check the result in register A.

Program:

```
        MOV R0,#50H     ;R0 is the pointer to the data
        MOV R1,#10H     ;R1 is the counter
        MOV B,#0        ;B=0
BACK:MOV A,@R0          ;move a number to A
        CJNE A,B,LOOP   ;compare with B
LOOP:JC LOOP1           ;if A<B, jump to loop1
        MOV B,A         ;if A>B, move it to B ie., the biggest number should be in B
        INC R0          ;increment the pointer
        DJNZ R1,BACK    ;repeat until the counter=0
        SJMP NEXT       ;jump to EXIT, the biggest number is in B
LOOP1:INC R0            ;this is another loop, taken when the biggest number was already in buffer
                        ;as comparison
        DJNZ R1,BACK    ;repeat until the counter=0
NEXT:MOV A,B            ;transfer the biggest number to A register
        MOV 60H,A       ;transfer the result to ROM location 6011
        END             ;end of asm source file
```

Output:


System Clock (MHz) 12.0
Update Freq.
8051
Data Memory
Modify RAM
Time: 530us - Instructions: 492
ØØØØ| MOV R2,#5ØH
ØØØ2| MOV R3,#1ØH
ØØØ4| MOV B,#Ø
ØØØ7| BACK:MOV A,@RØ
ØØØ8| CJNE A,B,LOOP
ØØØB| LOOP:JC LOOP1
ØØØD| MOV B,A
ØØØF| INC RØ
ØØ1Ø| DJNZ R1,BACK
ØØ12| SJMP NEXT
ØØ14| LOOP1:INC RØ
ØØ15| DJNZ R1,BACK
ØØ17| NEXT:MOV A,B
ØØ19| MOV 6ØH,A
END
AND Gate Disabled
Key Bounce Disabled
Standard
No Parity
8-bit UART @
4800 Baud
Rx Reset
Tx Send
0.0 V
input
11111111
ADC
MAX
MIN
Motor Enabled
0.0 V
output
Scope
DAC

Result:

Thus the program to text data transfer between Register and Memory was written and executed successfully.

Ex No:3	Perform ALU operation using 8051 microcontroller
Date:	

Aim:

The purpose of this experiment is to add, subtract, multiply and divide the given two 8 bit numbers and store them in a memory location.

Software Requirement:

Edsim 51 Simulator.

Program Logic:

To perform addition in 8051 one of the data should be in accumulator, another data can be in any of the general purpose register or in memory or immediate data. After addition the sum will be in accumulator. The sum of two 8-bit data can be either 8-bits(sum only) or 9-bits(sum and carry). The accumulator can accumulate only the sum and there is a carry the 8051 will indicate by setting carry flag. Hence one of the register is used to account for carry.

The 8051 has MUL instruction unlike many other 8-bit processors. MUL instruction multiplies the unsigned 8-bit integers in A and B. The lower order byte of the product is left in A and the higher order byte in B.

The 8051 has DIV instruction unlike many other 8-bit processors. DIV instruction divides the unsigned 8-bit integers in A and B. The accumulator receives the integer part of the quotient and the register B receives the remainder.

Procedure:

Step-1: Enter the opcodes from memory location 4200.

Step-2: Execute the program.

Step-3: Check for the result at 4100 and 4101.Using the accumulator, subtraction is performed and the result is stored. Immediate addressing is employed. The SUBB instruction drives the result in the accumulator.

Program:

1.ADDITION:

```
CLR C                ;make CY=0
MOV A,#45H           ;load the low byte into A
ADD A,#0ECH          ;add the low byte ,now A=31,CY=1
MOV R0,A             ;Save the low byte of sum in R0
MOV A,#02H           ;load the high byte into A
ADD A,#0FCH          ;add the high bytes with carry
MOV R1,A             ;save the high byte of result n R1
```

2.SUBTRACTION:

i) Subtraction with CY=0:

```
CLR C                ;make CY=0
MOV A,#3FH           ;load 3FH into A(A=3FH)
MOV R3,#23H          ;load 23H into R3(R3=23H)
SUBB A,R3            ;subtract A-R3,place the result in A
```

ii) Subtraction with CY=1:

```
MOV A,62H            ;A=62H
SUBB A,#96H          ;62H-96H=CCH with CY=1
MOV R7,A             ;save the result
MOV A,#27H           ;A=27H
SUBB A,#12H          ;27H-12H-1=14H
MOV R6,A             ;save the result
```

3)MULTIPLICATION:

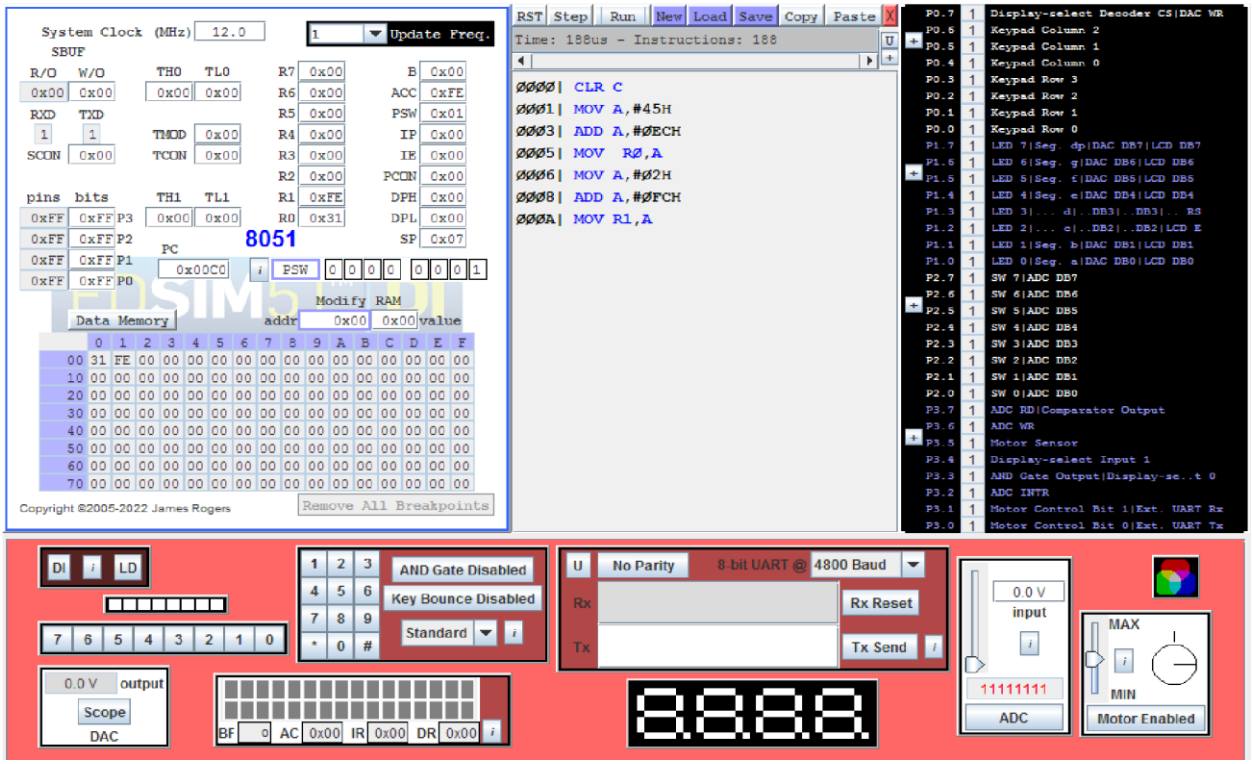
```
MOV A,#25H           ;load 25H to reg A
MOV B,65H            ;load 65H to reg B
MUL AB               ;25H*65H=E99 where
                     ;B=0EH and A=99H
```

4)DIVISION:

```
MOV A,#95H           ;load 95 into A
MOV B,#0AH           ;move the divisor into B
DIV AB               ;divide the hex number by 10
MOV R0,B             ;the remainder is in B, move it to R0
MOV B,#0AH           ;reload the divisor into B
DIV AB               ;divide the quotient by 10
MOV R1,B             ;move the remainder to R1
MOV R2,A             ;move the last quotient to R2
```

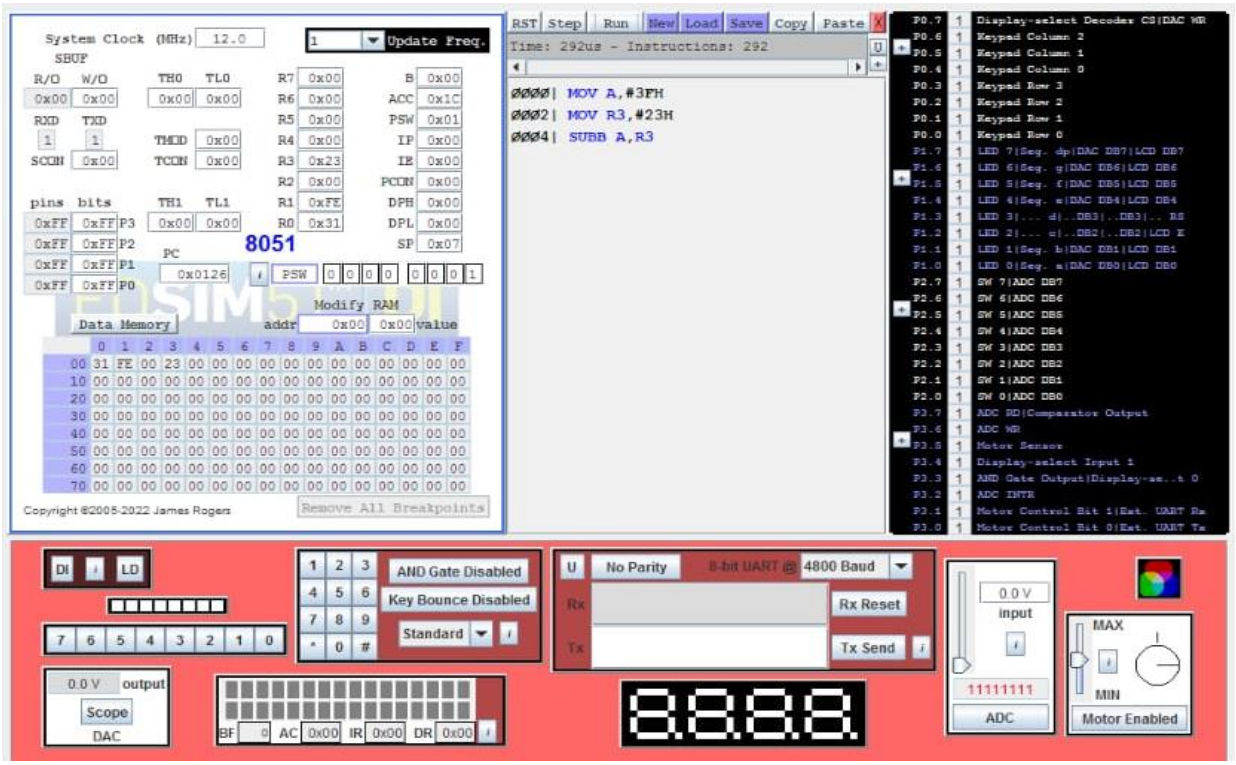
Output:

1.ADDITION:

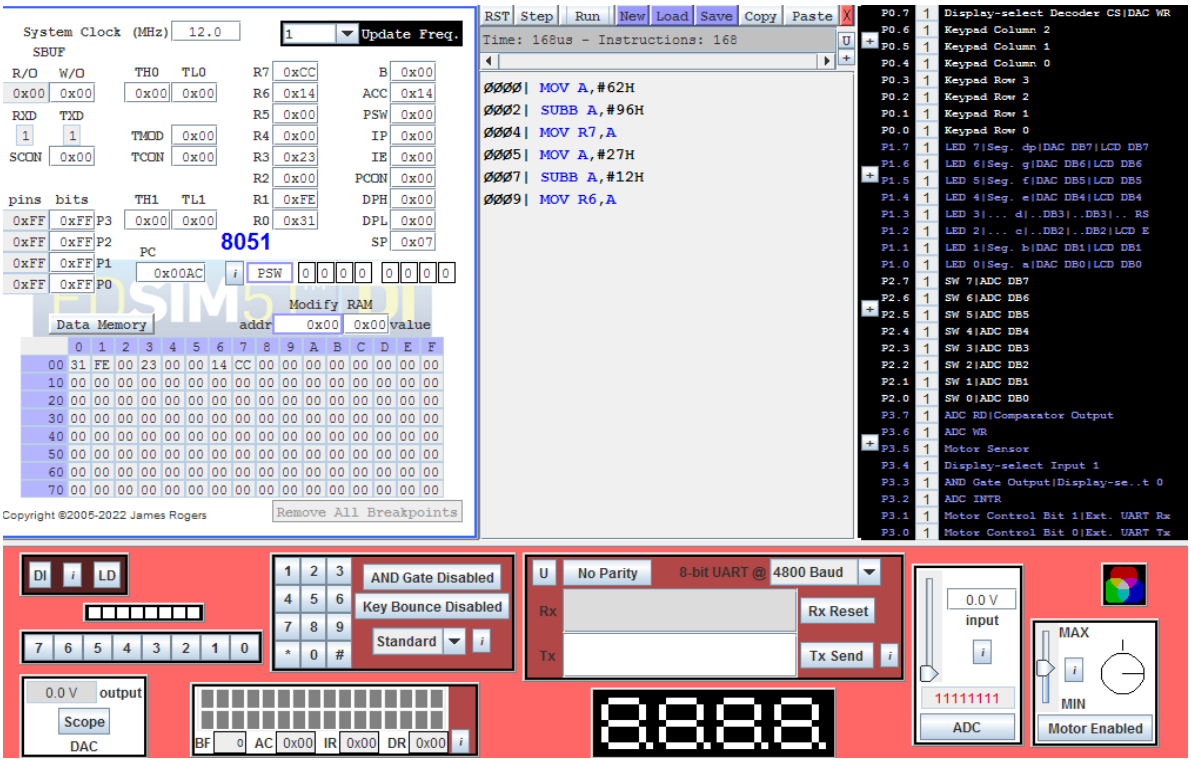


2.SUBTRACTION:

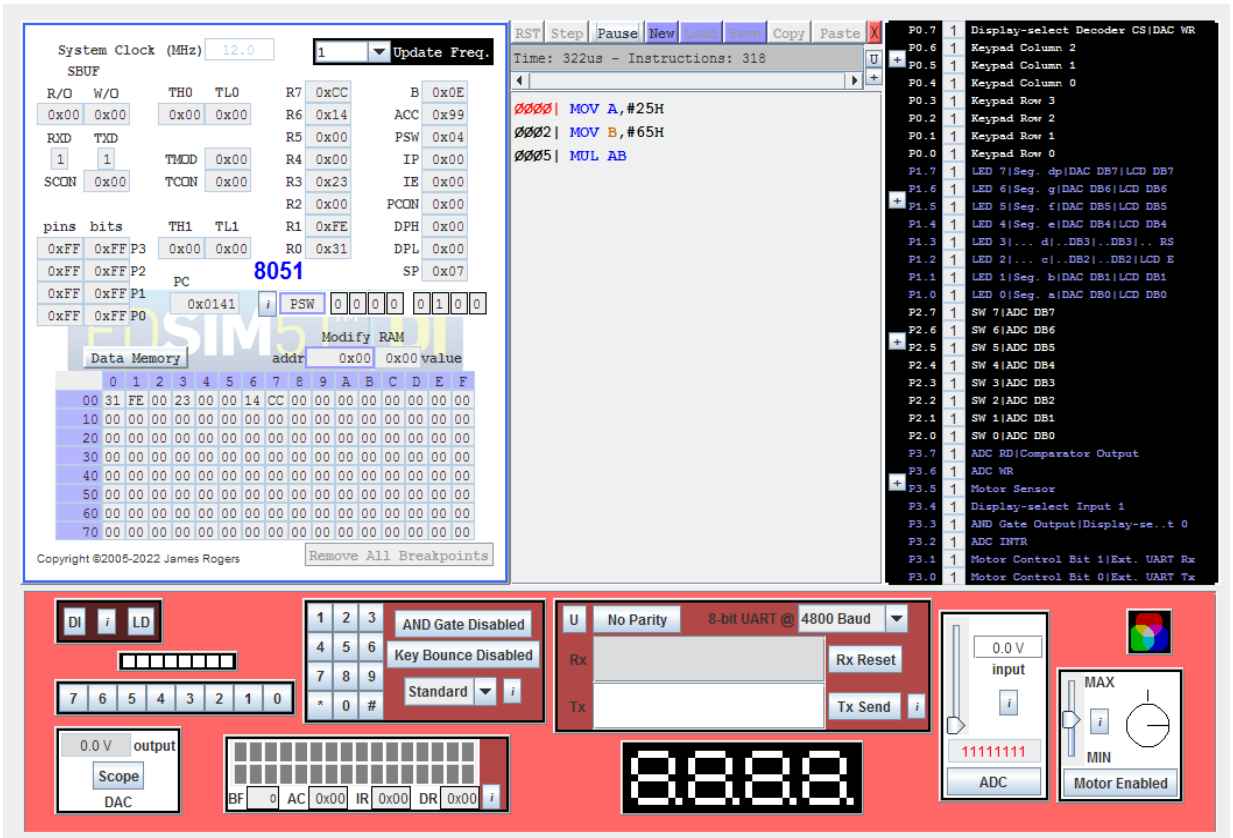
i) Subtraction with CY=0:



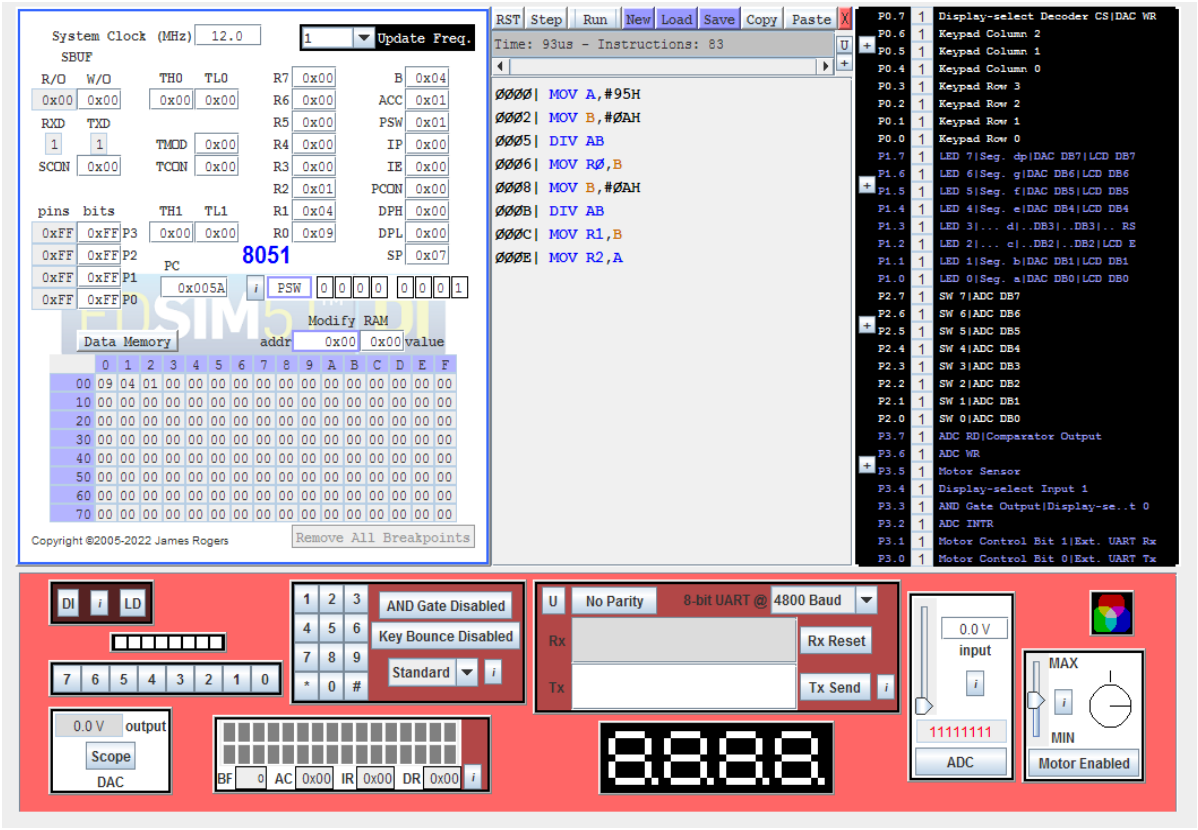
ii) Subtraction with CY=1:



3.MULTIPLICATION:



4.DIVISION


System Clock (MHz) 12.0
Update Freq.
8051
RST Step Run New Load Save Copy Paste
Time: 93us - Instructions: 83
ØØØØ| MOV A,#95H
ØØØ2| MOV B,#ØAH
ØØØ5| DIV AB
ØØØ6| MOV RØ,B
ØØØ8| MOV B,#ØAH
ØØØB| DIV AB
ØØØC| MOV R1,B
ØØØE| MOV R2,A
Data Memory
Modify RAM
Copyright ©2005-2022 James Rogers
Remove All Breakpoints
AND Gate Disabled
Key Bounce Disabled
Standard
No Parity
8-bit UART @ 4800 Baud
Rx Reset
Tx Send
Scope
input
MAX
MIN
Motor Enabled
ADC

Result:

Thus the ALU operation using 8051 microcontroller was perform successfully.