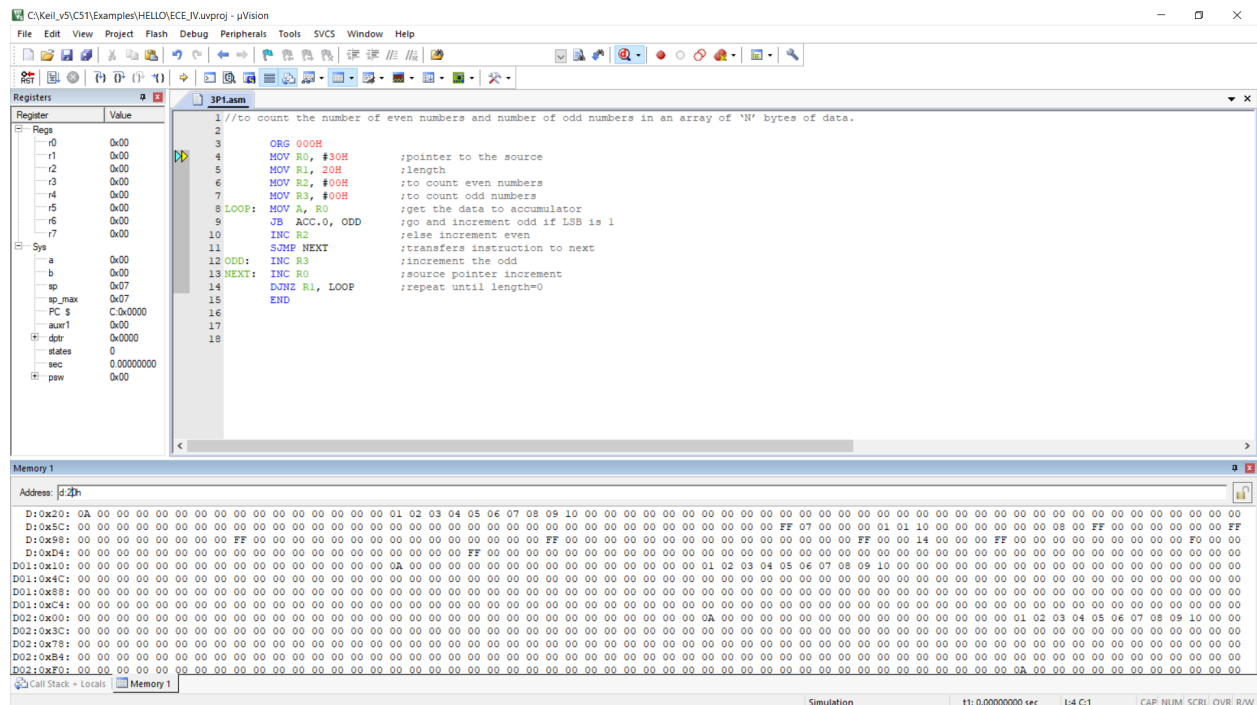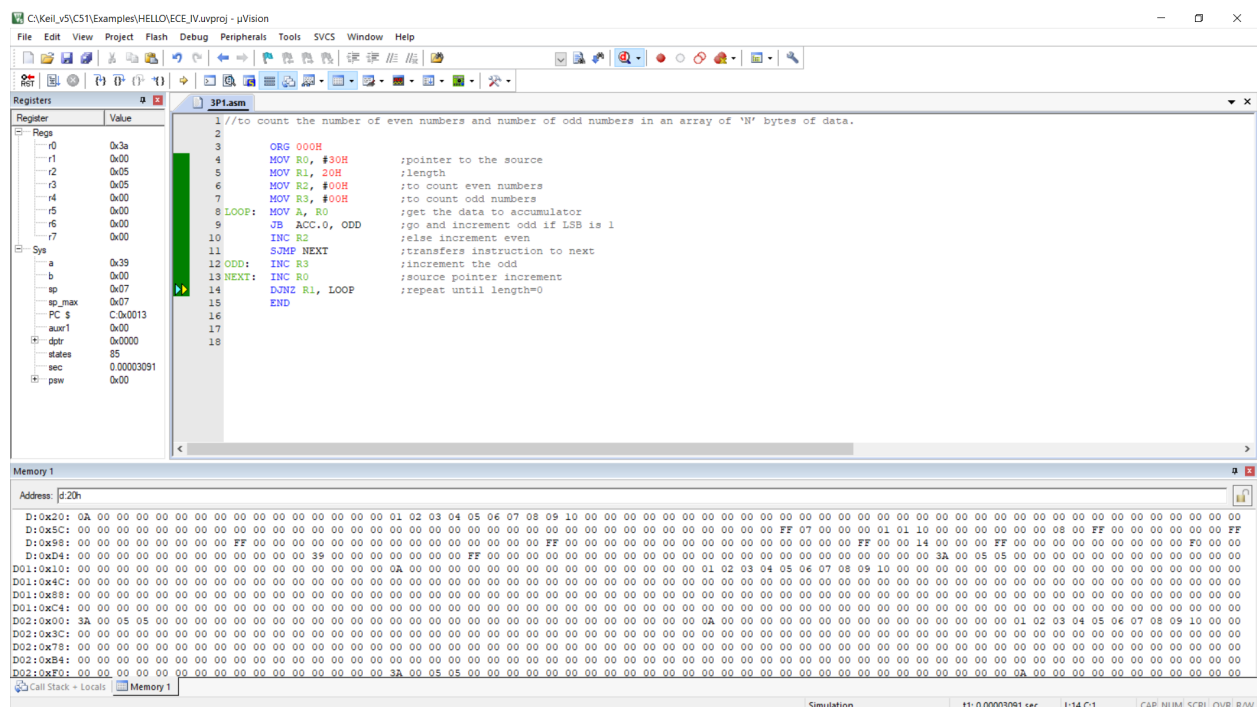# SET 3

Write an 8051 assembly level program to count the number of even numbers and number of odd numbers in an array of 'N' bytes of data.
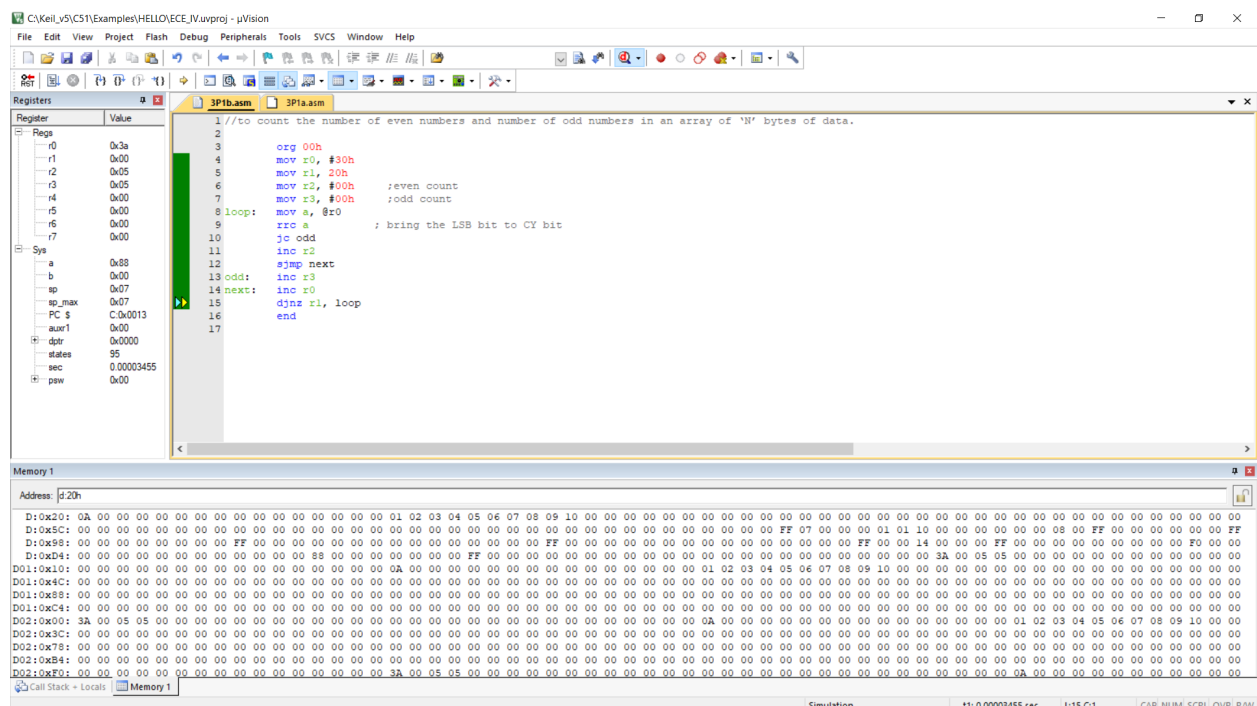
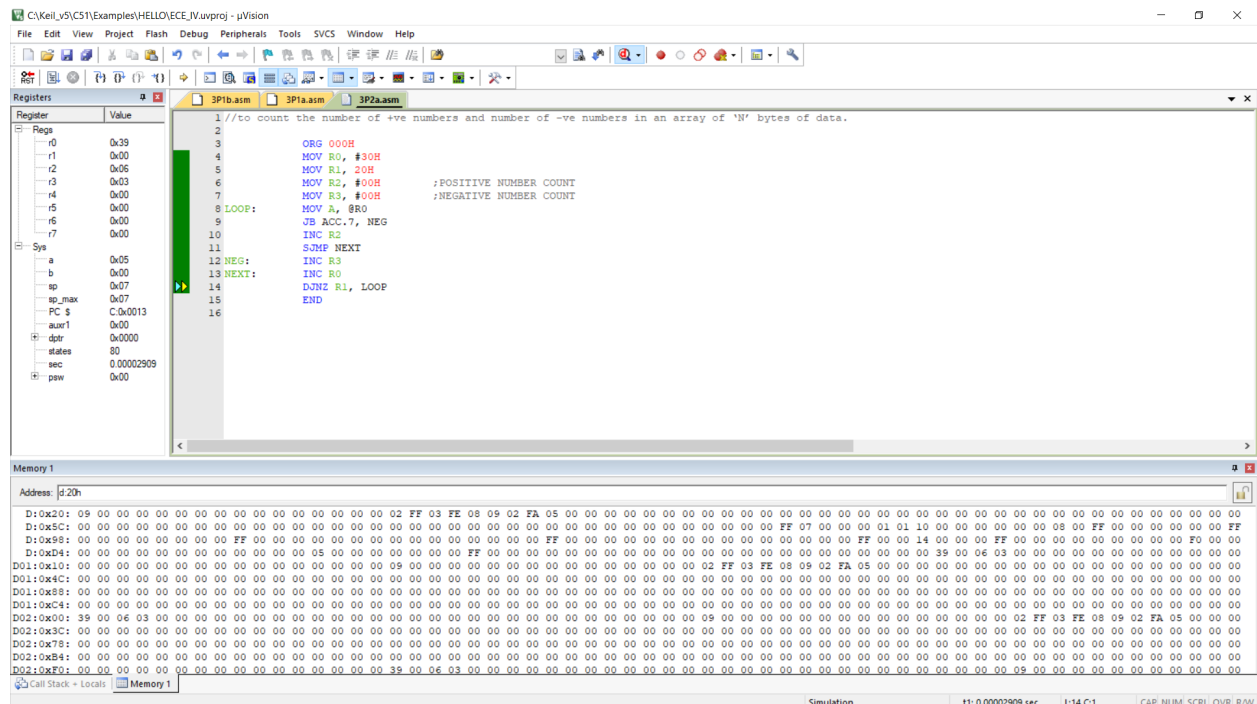Before execution:

After execution:

Write an 8051 assembly level program to count the number of even numbers and number of odd numbers in an array of 'N' bytes of data.

Before execution:

After execution:


```asm
//to count the number of even numbers and number of odd numbers in an array of 'N' bytes of data.

        org 00h
        mov r0, #30h
        mov r1, 20h
        mov r2, #00h    ;even count
        mov r3, #00h    ;odd count
loop:   mov a, @r0
        rrc a           ; bring the LSB bit to CY bit
        jc odd
        inc r2
        sjmp next
odd:    inc r3
next:   inc r0
        djnz r1, loop
        end
```

Write an 8051 assembly level program to count the number of +ve numbers and number of -ve numbers in an array of 'N' bytes of data.

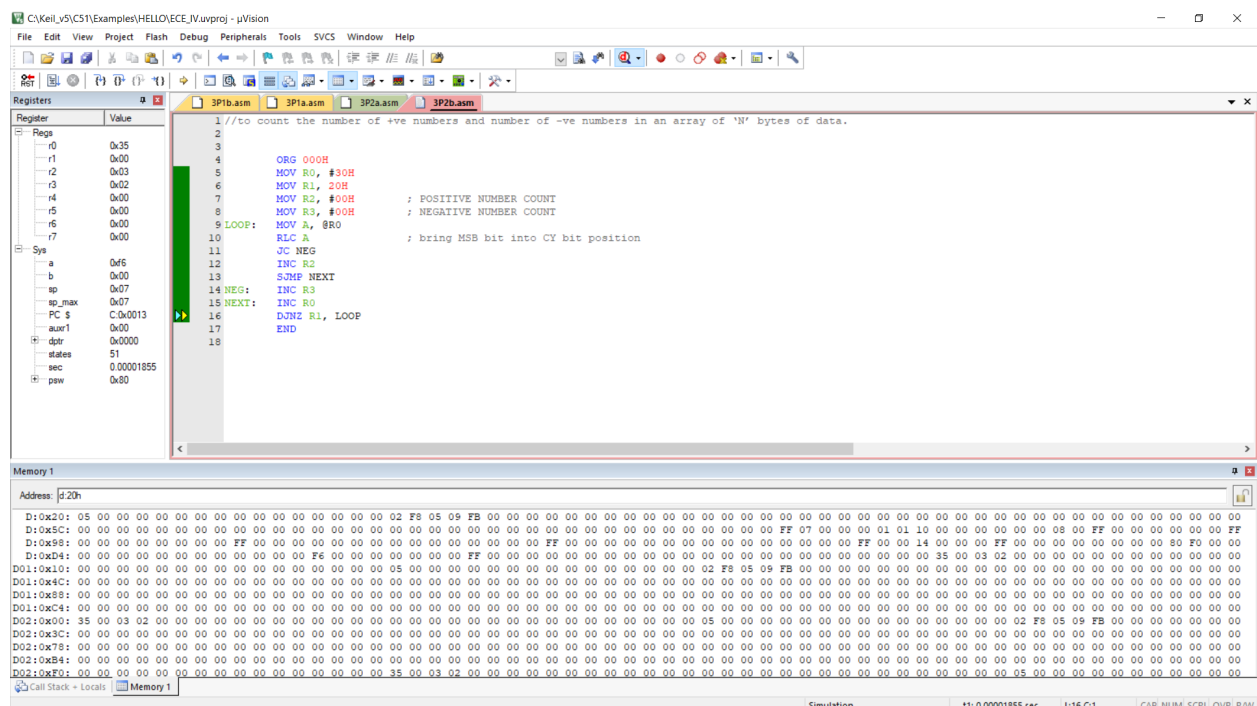## Before execution:

After execution:

The assembly program shown in the editor:

```
1 //to count the number of +ve numbers and number of -ve numbers in an array of 'N' bytes of data.
2
3           ORG 000H
4           MOV R0, #30H
5           MOV R1, 20H
6           MOV R2, #00H        ;POSITIVE NUMBER COUNT
7           MOV R3, #00H        ;NEGATIVE NUMBER COUNT
8 LOOP:     MOV A, @R0
9           JB ACC.7, NEG
10          INC R2
11          SJMP NEXT
12 NEG:     INC R3
13 NEXT:    INC R0
14          DJNZ R1, LOOP
15          END
16
```

Write an 8051 assembly level program to count the number of +ve numbers and number of -ve numbers in an array of 'N' bytes of data.

Before execution:

After execution:

The assembly code shown in the screenshots:

```
1  //to count the number of +ve numbers and number of -ve numbers in an array of 'N' bytes of data.
2
3
4          ORG 000H
5          MOV R0, #30H
6          MOV R1, 20H
7          MOV R2, #00H      ; POSITIVE NUMBER COUNT
8          MOV R3, #00H      ; NEGATIVE NUMBER COUNT
9  LOOP:   MOV A, @R0
10         RLC A             ; bring MSB bit into CY bit position
11         JC NEG
12         INC R2
13         SJMP NEXT
14 NEG:    INC R3
15 NEXT:   INC R0
16         DJNZ R1, LOOP
17         END
18
```

Check whether the given byte of data is present in an array of 'N' bytes of
data. If present send 00 in Port 0 else send FF in Port 0.

Before execution:

After execution:

Read the data from Port 1. If P1.1 is at logic 0, find the largest number in an array of 'N' bytes of data and store in location 40h. If P1.0 is at logic 1, find the smallest number in the array and store in the location 40h.

Before execution:

After execution: