AIM 1: Matrix Addition of 8 bit numbers

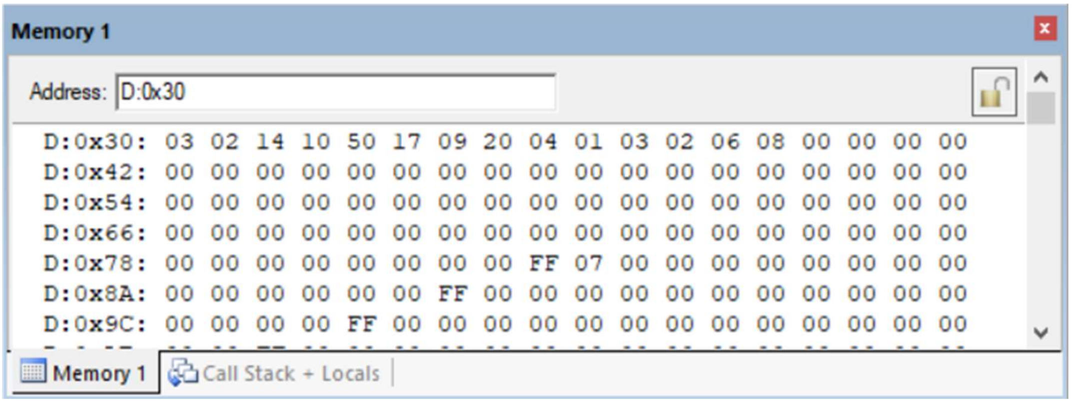
ASSEMBLY LANGUAGE CODE:

- Representing the number of rows in the location 30
- Representing the number of columns in the location 31
- Each matrix is having R7 elements
- implementing a counter m*n 32(1,1),33(2,1),34...

```
;[14 10 50] + [4 1 3] = [18 11 53]
;[17 16 20] + [2 6 8] = [19 22 28]

            MOV R0,#30H              ;pointing to first location
            MOV A,@R0;                    ;Getting the data to accumulator A=m
            INC R0;
            MOV B,@R0;                    ;B=n
            MUL AB;                 ;A=mxn
            MOV R7,A;               ;with reference to the second matrix
            MOV R6,A;               ;With reference to count
            MOV B,#60H;                   ;result from 60 memory
LOOP:INC R0;
            MOV A,R0;               ;reference to 1st matrix
            ADD A,R7;
            MOV R1,A ;              ;reference to 2nd matrix
            MOV A,@R0;
            ADD A,@R1;
            MOV 20H,R0;                   ;saving to 20h location
            MOV R0,B;
            MOV @R0,A;
            INC B;
            MOV R0,20H;
            DJNZ R6,LOOP
STOP: SJMP STOP;
            END;
```

→

INPUT GIVEN:



Memory 1

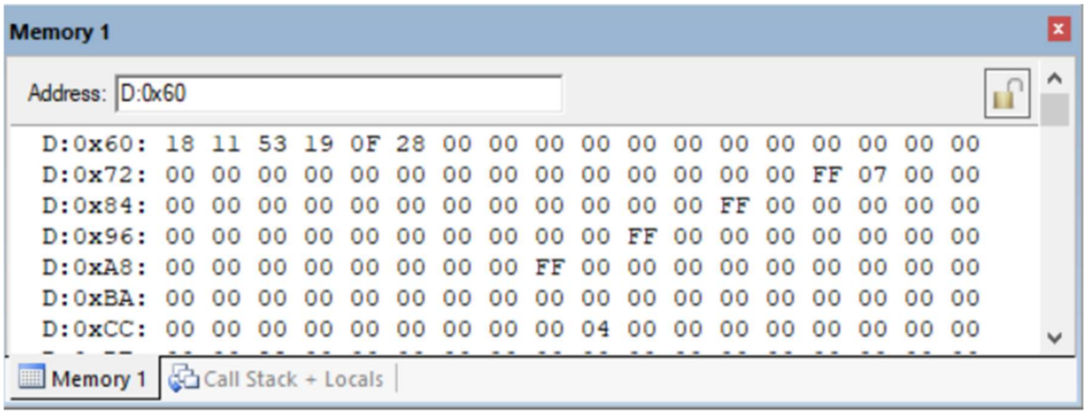
Address: D:0x30

```
D:0x30: 03 02 14 10 50 17 09 20 04 01 03 02 06 08 00 00 00 00
D:0x42: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x54: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x66: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x78: 00 00 00 00 00 00 00 00 FF 07 00 00 00 00 00 00 00 00
D:0x8A: 00 00 00 00 00 00 FF 00 00 00 00 00 00 00 00 00 00 00
D:0x9C: 00 00 00 00 FF 00 00 00 00 00 00 00 00 00 00 00 00 00
```

EXPECTED OUTPUT:

$$\begin{bmatrix} 14 & 10 & 50 \\ 17 & 9 & 20 \end{bmatrix} + \begin{bmatrix} 4 & 1 & 3 \\ 2 & 6 & 8 \end{bmatrix} = \begin{bmatrix} 18 & 11 & 53 \\ 19 & 15 & 28 \end{bmatrix}$$

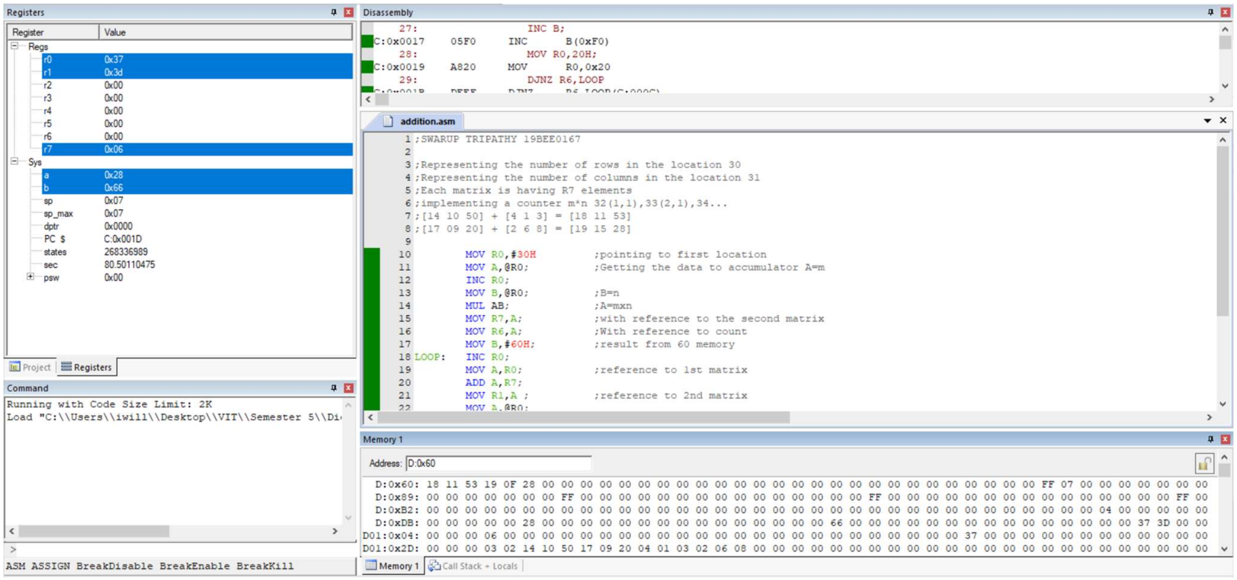
OBSERVED OUTPUT:



Memory 1

Address: D:0x60

```
D:0x60: 18 11 53 19 0F 28 00 00 00 00 00 00 00 00 00 00 00 00
D:0x72: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF 07 00 00
D:0x84: 00 00 00 00 00 00 00 00 00 00 00 00 FF 00 00 00 00 00
D:0x96: 00 00 00 00 00 00 00 00 00 00 FF 00 00 00 00 00 00 00
D:0xA8: 00 00 00 00 00 00 00 00 FF 00 00 00 00 00 00 00 00 00
D:0xBA: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0xCC: 00 00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00 00
```



Registers

Register	Value
Regs	
r0	0x37
r1	0x3d
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x06
Sys	
a	0x28
b	0x66
sp	0x07
sp_max	0x07
dptr	0x0000
PC $	C:0x001D
states	268336989
sec	80.50110475
psw	0x00

Command

```
Running with Code Size Limit: 2K
Load "C:\\Users\\iwill\\Desktop\\VIT\\Semester 5\\Di
```

```
>
ASM ASSIGN BreakDisable BreakEnable BreakKill
```

Disassembly

```
    27:                  INC B;
C:0x0017    05F0     INC      B(0xF0)
    28:                  MOV R0,20H;
C:0x0019    A820     MOV      R0,0x20
    29:                  DJNZ R6,LOOP
```

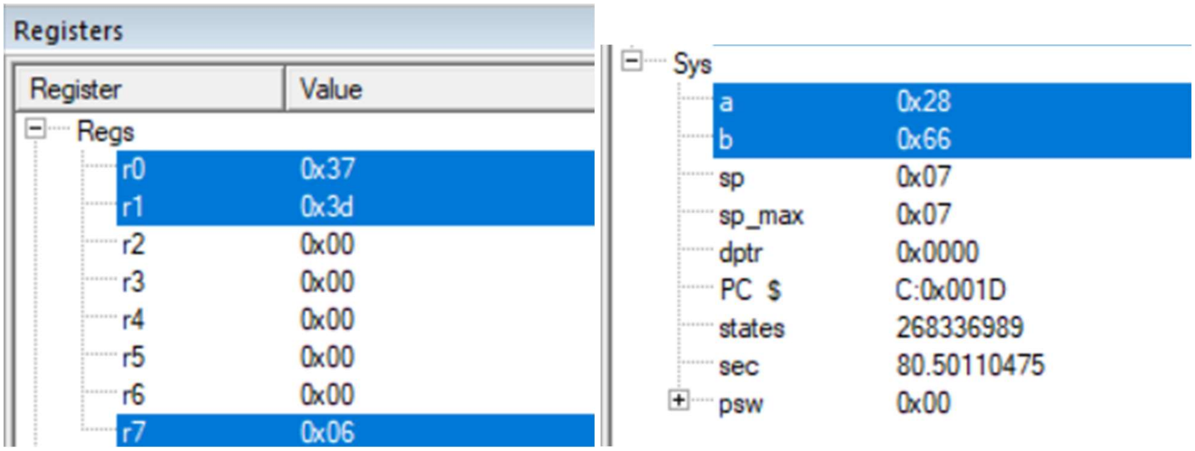
addition.asm

```
;SWARUP TRIPATHY 19BEE0167

;Representing the number of rows in the location 30
;Representing the number of columns in the location 31
;Each matrix is having R7 elements
;implementing a counter m*n 32(1,1),33(2,1),34...
;[14 10 50] + [4 1 3] = [18 11 53]
;[17 09 20] + [2 6 8] = [19 15 28]

        MOV R0,#30H           ;pointing to first location
        MOV A,@R0;            ;Getting the data to accumulator A=m
        INC R0;
        MOV B,@R0;            ;B=n
        MUL AB;               ;A=mxn
        MOV R7,A;             ;with reference to the second matrix
        MOV R6,A;             ;With reference to count
        MOV B,#60H;           ;result from 60 memory
LOOP:   INC R0;
        MOV A,R0;             ;reference to 1st matrix
        ADD A,R7;
        MOV R1,A ;            ;reference to 2nd matrix
        MOV A,@R0;
```

Memory 1

Address: D:0x60

```
  D:0x60: 18 11 53 19 0F 28 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 FF 07 00 00 00 00 00 00 00
  D:0x89: 00 00 00 00 00 00 00 FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF 00
  D:0xB2: 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 00 00 00 00 00 00 00 00 04 00 00 00 00 00
  D:0xDB: 00 00 00 00 00 28 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 66 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 37 3D 00 00
D01:0x04: 00 00 00 06 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 37 00 00 00 00 00 00 00 00 00 00 00 00
D01:0x2D: 00 00 00 03 02 14 10 50 17 09 20 04 01 03 02 06 08 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```



Registers

Register	Value
Regs	
r0	0x37
r1	0x3d
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x06

Sys	
a	0x28
b	0x66
sp	0x07
sp_max	0x07
dptr	0x0000
PC $	C:0x001D
states	268336989
sec	80.50110475
psw	0x00

AIM 2: Matrix Subtraction of 8 bit numbers

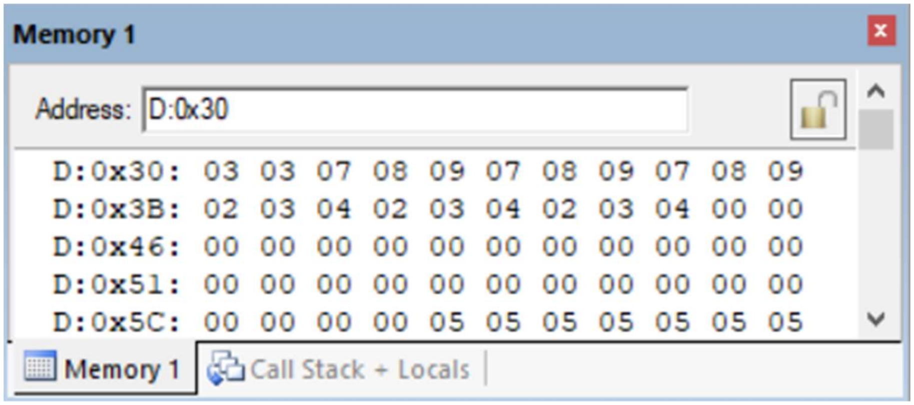
ASSEMBLY LANGUAGE CODE:

- Representing the number of rows in the location 30
- Representing the number of columns in the location 31
- Each matrix is having R7 elements
- implementing a counter m*n 32(1,1),33(2,1),34...

```
;[14 10 50] - [4 1 3] = [18 11 53]
;[17 16 20] - [2 6 8] = [19 22 28]

   mov r0,#30h
      mov a,@r0
      inc r0
      mov b,@r0
      mul ab
      mov r7,a
      mov r6,a
      mov b,#60h
loop: inc r0
      mov a,r0
      add a,r7
      mov r1,a
      mov a,@r0
      subb a,@r1
      mov 20h,r0
      mov r0,b
      mov @r0,a
      inc b
      mov r0,20h
      djnz r6,loop
stop: sjmp stop
      End
```

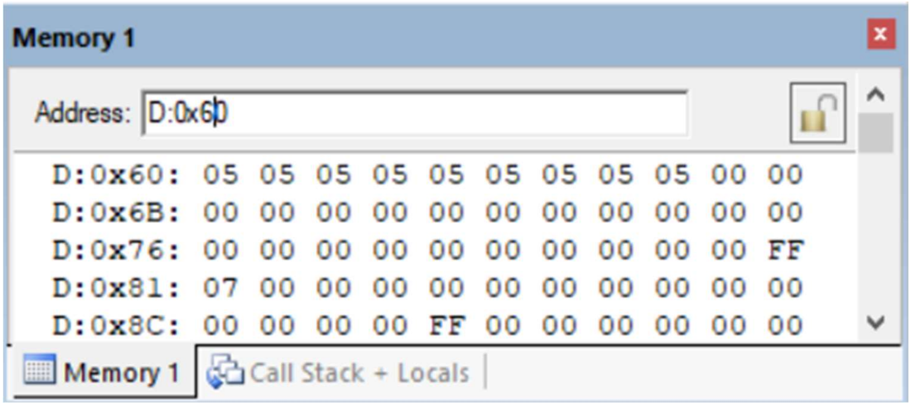
→

INPUT GIVEN:

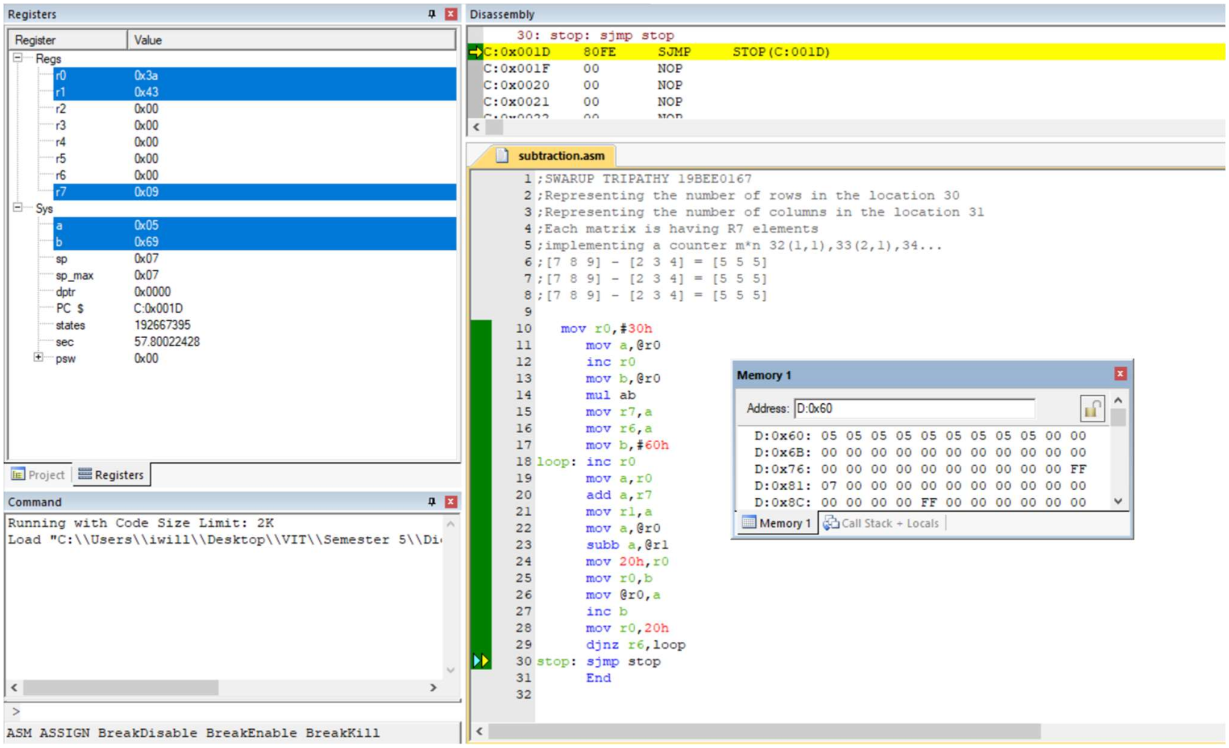


```
Address: D:0x30

D:0x30: 03 03 07 08 09 07 08 09 07 08 09
D:0x3B: 02 03 04 02 03 04 02 03 04 00 00
D:0x46: 00 00 00 00 00 00 00 00 00 00 00
D:0x51: 00 00 00 00 00 00 00 00 00 00 00
D:0x5C: 00 00 00 00 05 05 05 05 05 05 05
```

OBSERVED OUTPUT:



```
Address: D:0x60

D:0x60: 05 05 05 05 05 05 05 05 05 00 00
D:0x6B: 00 00 00 00 00 00 00 00 00 00 00
D:0x76: 00 00 00 00 00 00 00 00 00 00 FF
D:0x81: 07 00 00 00 00 00 00 00 00 00 00
D:0x8C: 00 00 00 00 FF 00 00 00 00 00 00
```



Registers

Register	Value
Regs	
r0	0x3a
r1	0x43
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x09
Sys	
a	0x05
b	0x69
sp	0x07
sp_max	0x07
dptr	0x0000
PC $	C:0x001D
states	192667395
sec	57.80022428
psw	0x00

Disassembly

```
    30: stop: sjmp stop
C:0x001D    80FE    SJMP    STOP(C:001D)
C:0x001F    00      NOP
C:0x0020    00      NOP
C:0x0021    00      NOP
```

subtraction.asm

```
;SWARUP TRIPATHY 19BEE0167
;Representing the number of rows in the location 30
;Representing the number of columns in the location 31
;Each matrix is having R7 elements
;implementing a counter m*n 32(1,1),33(2,1),34...
;[7 8 9] - [2 3 4] = [5 5 5]
;[7 8 9] - [2 3 4] = [5 5 5]
;[7 8 9] - [2 3 4] = [5 5 5]

  mov r0,#30h
     mov a,@r0
     inc r0
     mov b,@r0
     mul ab
     mov r7,a
     mov r6,a
     mov b,#60h
loop: inc r0
     mov a,r0
     add a,r7
     mov r1,a
     mov a,@r0
     subb a,@r1
     mov 20h,r0
     mov r0,b
     mov @r0,a
     inc b
     mov r0,20h
     djnz r6,loop
stop: sjmp stop
     End
```

Command

```
Running with Code Size Limit: 2K
Load "C:\\Users\\iwill\\Desktop\\VIT\\Semester 5\\Di
>
ASM ASSIGN BreakDisable BreakEnable BreakKill
```

Memory 1

```
Address: D:0x60

D:0x60: 05 05 05 05 05 05 05 05 05 00 00
D:0x6B: 00 00 00 00 00 00 00 00 00 00 00
D:0x76: 00 00 00 00 00 00 00 00 00 00 FF
D:0x81: 07 00 00 00 00 00 00 00 00 00 00
D:0x8C: 00 00 00 00 FF 00 00 00 00 00 00
```

INFERENCE:

Using Keil software I was able to run the matrix addition of two matrices and subtraction as what can be checked through observed output. For addition we could see our expected output matched with our observed output.