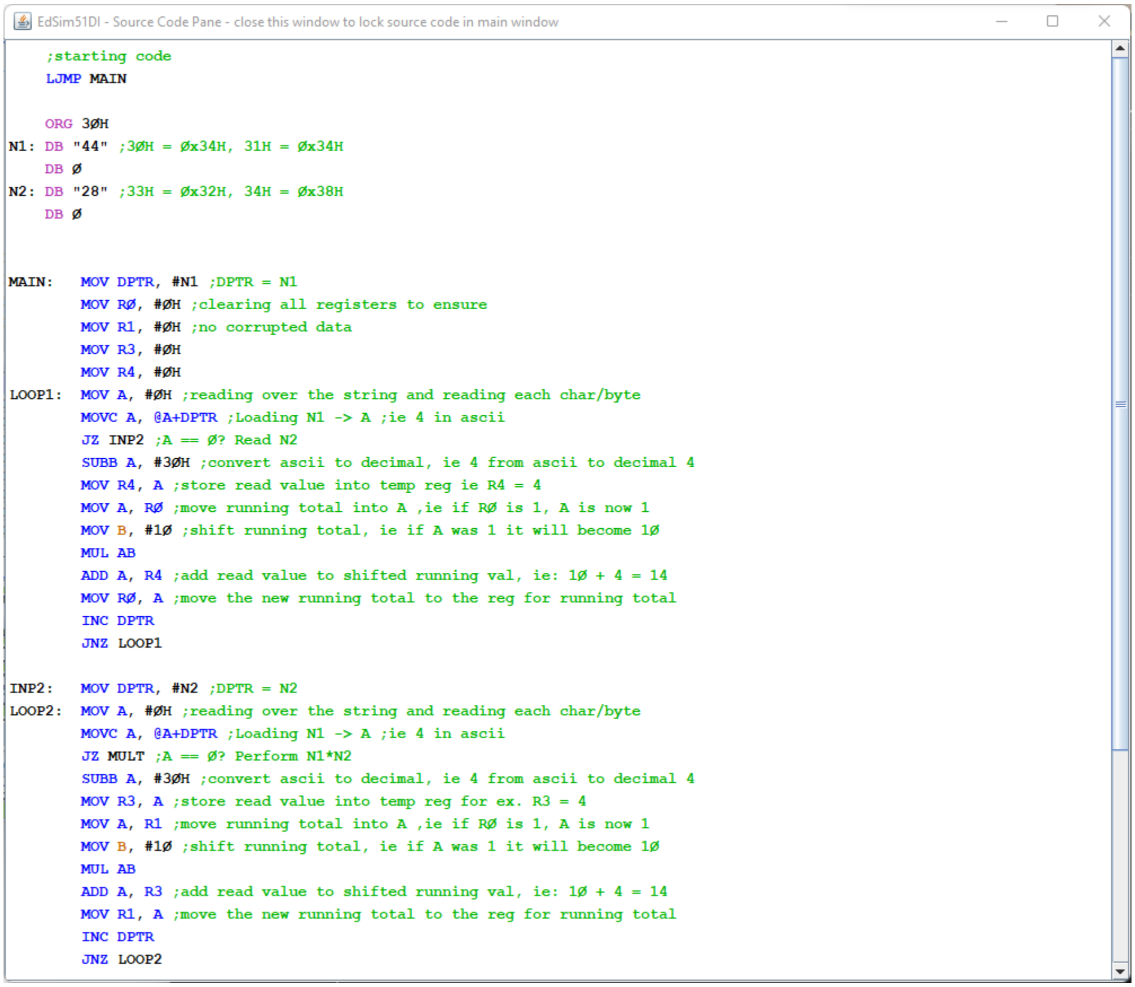
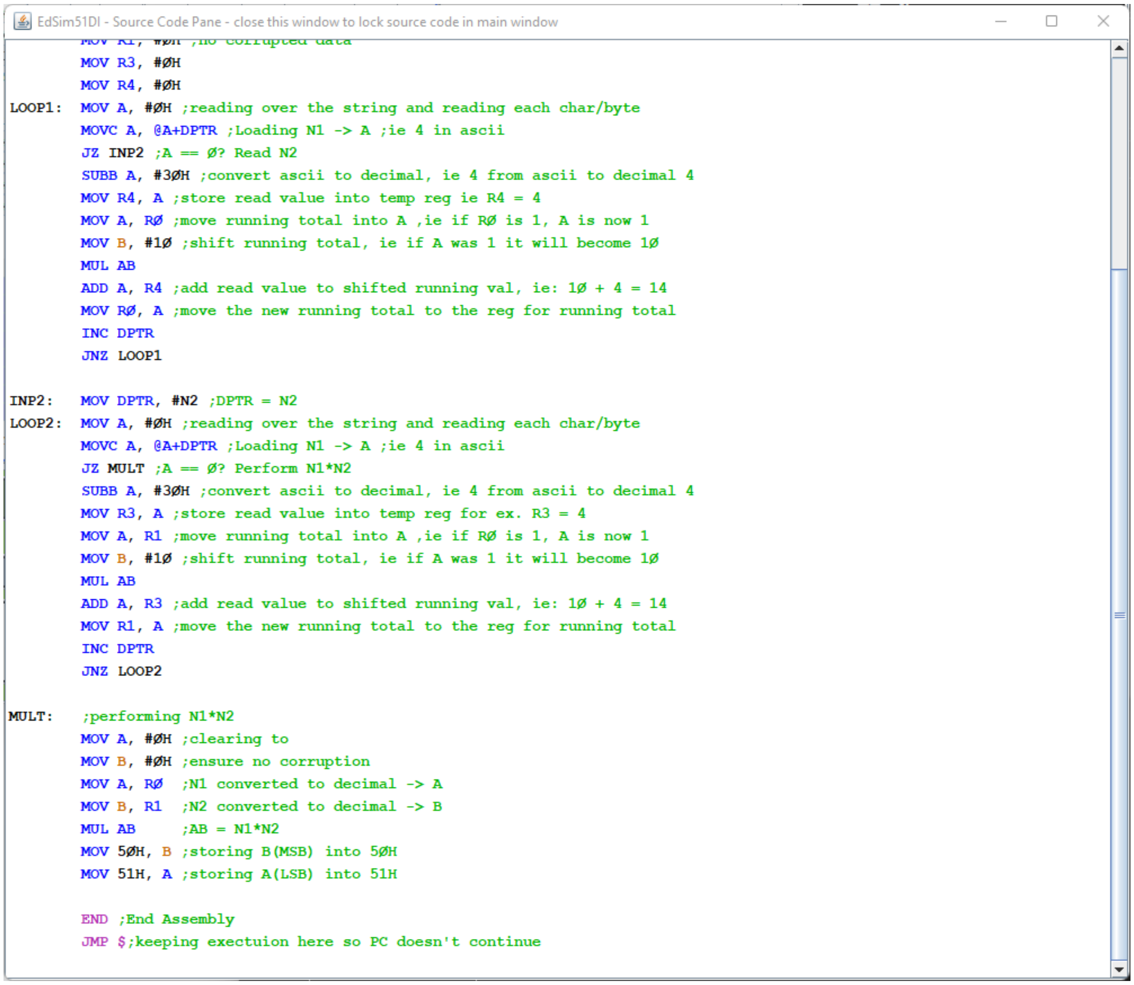
PROCESSOR HW/SW INTER

EECS 113
Assignment 2
Kyle Zyler Cayanan
80576955
04/26/2022

For this assignment we are to convert two input strings from ASCII digits to their proper decimal representation. We then take the converted strings and multiply the values and store them in data memory locations 50H (MSB) and 51H (LSB).


EdSim51DI - Source Code Pane - close this window to lock source code in main window

```
    ;starting code
    LJMP MAIN

    ORG 30H
N1: DB "44" ;30H = 0x34H, 31H = 0x34H
    DB 0
N2: DB "28" ;33H = 0x32H, 34H = 0x38H
    DB 0


MAIN:   MOV DPTR, #N1 ;DPTR = N1
        MOV R0, #0H ;clearing all registers to ensure
        MOV R1, #0H ;no corrupted data
        MOV R3, #0H
        MOV R4, #0H
LOOP1:  MOV A, #0H ;reading over the string and reading each char/byte
        MOVC A, @A+DPTR ;Loading N1 -> A ;ie 4 in ascii
        JZ INP2 ;A == 0? Read N2
        SUBB A, #30H ;convert ascii to decimal, ie 4 from ascii to decimal 4
        MOV R4, A ;store read value into temp reg ie R4 = 4
        MOV A, R0 ;move running total into A ,ie if R0 is 1, A is now 1
        MOV B, #10 ;shift running total, ie if A was 1 it will become 10
        MUL AB
        ADD A, R4 ;add read value to shifted running val, ie: 10 + 4 = 14
        MOV R0, A ;move the new running total to the reg for running total
        INC DPTR
        JNZ LOOP1

INP2:   MOV DPTR, #N2 ;DPTR = N2
LOOP2:  MOV A, #0H ;reading over the string and reading each char/byte
        MOVC A, @A+DPTR ;Loading N1 -> A ;ie 4 in ascii
        JZ MULT ;A == 0? Perform N1*N2
        SUBB A, #30H ;convert ascii to decimal, ie 4 from ascii to decimal 4
        MOV R3, A ;store read value into temp reg for ex. R3 = 4
        MOV A, R1 ;move running total into A ,ie if R0 is 1, A is now 1
        MOV B, #10 ;shift running total, ie if A was 1 it will become 10
        MUL AB
        ADD A, R3 ;add read value to shifted running val, ie: 10 + 4 = 14
        MOV R1, A ;move the new running total to the reg for running total
        INC DPTR
        JNZ LOOP2
```

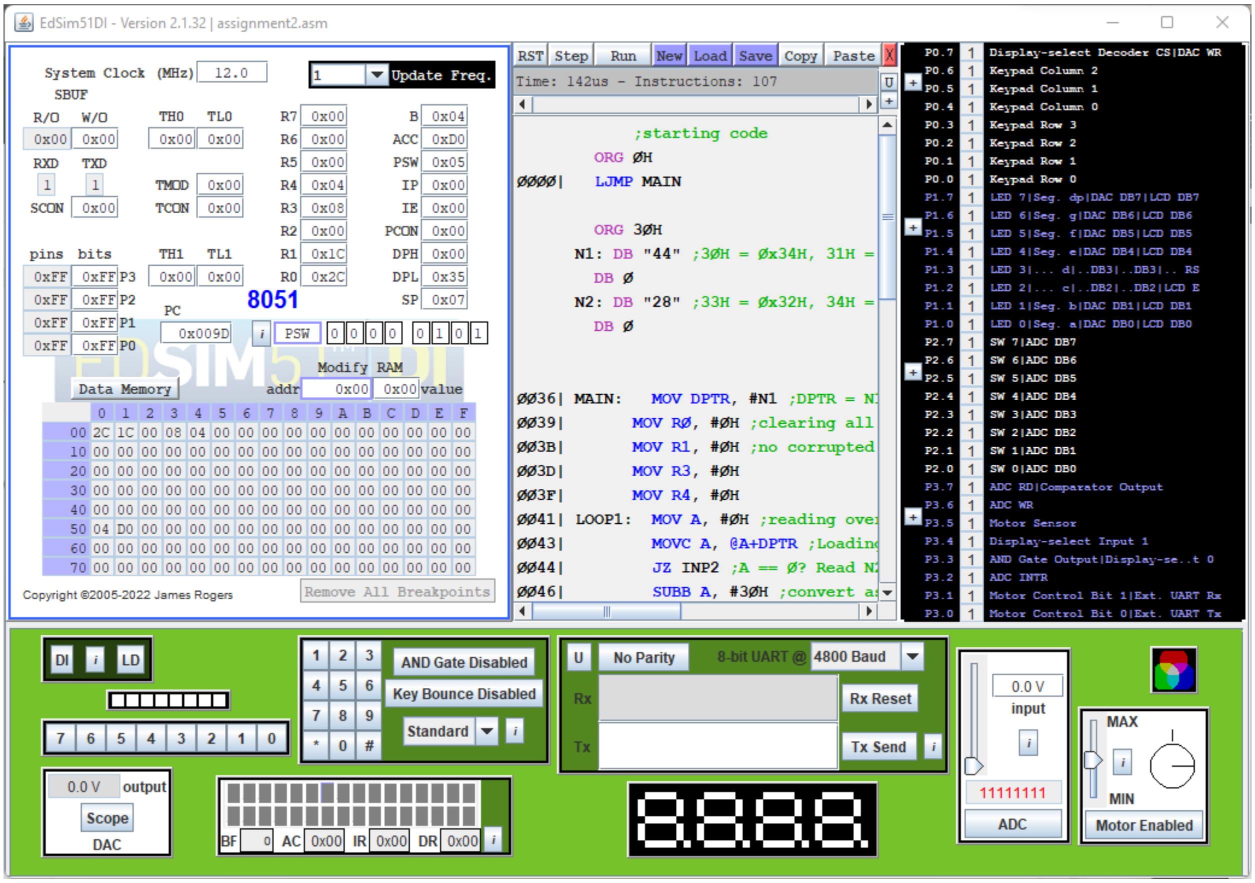

EdSim51DI - Source Code Pane - close this window to lock source code in main window

```
        MOV R1, #ØH ;no corrupted data
        MOV R3, #ØH
        MOV R4, #ØH
LOOP1:  MOV A, #ØH ;reading over the string and reading each char/byte
        MOVC A, @A+DPTR ;Loading N1 -> A ;ie 4 in ascii
        JZ INP2 ;A == Ø? Read N2
        SUBB A, #3ØH ;convert ascii to decimal, ie 4 from ascii to decimal 4
        MOV R4, A ;store read value into temp reg ie R4 = 4
        MOV A, RØ ;move running total into A ,ie if RØ is 1, A is now 1
        MOV B, #1Ø ;shift running total, ie if A was 1 it will become 1Ø
        MUL AB
        ADD A, R4 ;add read value to shifted running val, ie: 1Ø + 4 = 14
        MOV RØ, A ;move the new running total to the reg for running total
        INC DPTR
        JNZ LOOP1

INP2:   MOV DPTR, #N2 ;DPTR = N2
LOOP2:  MOV A, #ØH ;reading over the string and reading each char/byte
        MOVC A, @A+DPTR ;Loading N1 -> A ;ie 4 in ascii
        JZ MULT ;A == Ø? Perform N1*N2
        SUBB A, #3ØH ;convert ascii to decimal, ie 4 from ascii to decimal 4
        MOV R3, A ;store read value into temp reg for ex. R3 = 4
        MOV A, R1 ;move running total into A ,ie if RØ is 1, A is now 1
        MOV B, #1Ø ;shift running total, ie if A was 1 it will become 1Ø
        MUL AB
        ADD A, R3 ;add read value to shifted running val, ie: 1Ø + 4 = 14
        MOV R1, A ;move the new running total to the reg for running total
        INC DPTR
        JNZ LOOP2

MULT:   ;performing N1*N2
        MOV A, #ØH ;clearing to
        MOV B, #ØH ;ensure no corruption
        MOV A, RØ  ;N1 converted to decimal -> A
        MOV B, R1  ;N2 converted to decimal -> B
        MUL AB     ;AB = N1*N2
        MOV 5ØH, B ;storing B(MSB) into 5ØH
        MOV 51H, A ;storing A(LSB) into 51H

        END ;End Assembly
        JMP $;keeping exectuion here so PC doesn't continue
```

The code itself is commented and self documenting but will also be explained here. We have four main sections in which the code is broken up into; The input section, a section for retrieving N1, another section for retrieving N2, and the last section for performing the multiplication of the our retrieved values then storing it in the data memory. The first section starting from the ORG 30H instruction sets up our test cases and loads our strings into N1 and N2 at their respective labels.The next section is the setup of N1 and reading each char/byte and converting the value from ASCII to decimal in a loop, the result gets stored in R0. The section for N2 is a similar procedure, the only difference is that we store the result in different registers. The last section just loads the results onto the A and B accumulators and stores the results in our desired memory locations.

Final Memory and Register Values

Test Case 1: Our inputs N1 and N2 are 44 and 28 respectively. We can see N1 and N2 being stored in R0 and R1 as their hexadecimal representations. (2CH and 1CH). The expected output was 04D0H which is shown in memory locations 50H and 51H.



EdSim51DI - Version 2.1.32 | assignment2.asm

```
System Clock (MHz) 12.0        1   Update Freq.
 SBUF
R/O   W/O      TH0   TL0     R7 0x00      B 0x04
0x00  0x00     0x00  0x00    R6 0x00    ACC 0xD0
RXD   TXD                    R5 0x00    PSW 0x05
 1     1       TMOD  0x00    R4 0x04     IP 0x00
SCON  0x00     TCON  0x00    R3 0x08     IE 0x00
                             R2 0x00   PCON 0x00
pins  bits     TH1   TL1     R1 0x1C    DPH 0x00
0xFF  0xFF P3  0x00  0x00    R0 0x2C    DPL 0x35
0xFF  0xFF P2                8051        SP 0x07
0xFF  0xFF P1       PC
0xFF  0xFF P0     0x009D   i  PSW  0 0 0 0  0 1 0 1
                               Modify RAM
Data Memory                addr  0x00  0x00 value
```

```
    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00 2C 1C 00 08 04 00 00 00 00 00 00 00 00 00 00 00
10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
50 04 D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00
60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
70 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

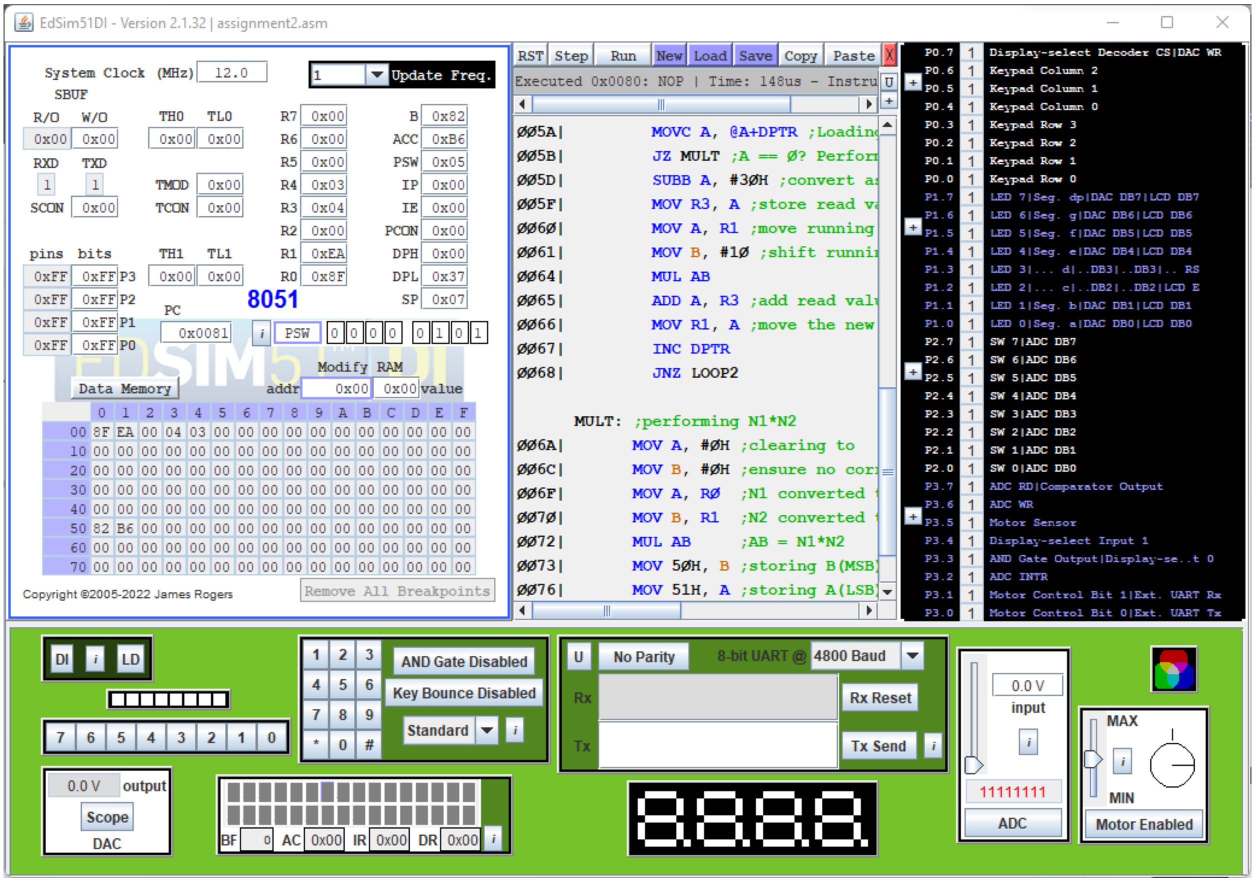
Copyright ©2005-2022 James Rogers

Time: 142us - Instructions: 107

```
              ;starting code
          ORG ØH
ØØØØ|     LJMP MAIN

          ORG 3ØH
       N1: DB "44" ;3ØH = Øx34H, 31H =
          DB Ø
       N2: DB "28" ;33H = Øx32H, 34H =
          DB Ø


ØØ36| MAIN:    MOV DPTR, #N1 ;DPTR = N
ØØ39|          MOV RØ, #ØH ;clearing all
ØØ3B|          MOV R1, #ØH ;no corrupted
ØØ3D|          MOV R3, #ØH
ØØ3F|          MOV R4, #ØH
ØØ41| LOOP1:   MOV A, #ØH ;reading ove
ØØ43|          MOVC A, @A+DPTR ;Loadin
ØØ44|          JZ INP2 ;A == Ø? Read N
ØØ46|          SUBB A, #3ØH ;convert a
```

```
P0.7 1 Display-select Decoder CS|DAC WR
P0.6 1 Keypad Column 2
P0.5 1 Keypad Column 1
P0.4 1 Keypad Column 0
P0.3 1 Keypad Row 3
P0.2 1 Keypad Row 2
P0.1 1 Keypad Row 1
P0.0 1 Keypad Row 0
P1.7 1 LED 7|Seg. dp|DAC DB7|LCD DB7
P1.6 1 LED 6|Seg. g|DAC DB6|LCD DB6
P1.5 1 LED 5|Seg. f|DAC DB5|LCD DB5
P1.4 1 LED 4|Seg. e|DAC DB4|LCD DB4
P1.3 1 LED 3|... d|..DB3|..DB3|.. RS
P1.2 1 LED 2|... c|..DB2|..DB2|LCD E
P1.1 1 LED 1|Seg. b|DAC DB1|LCD DB1
P1.0 1 LED 0|Seg. a|DAC DB0|LCD DB0
P2.7 1 SW 7|ADC DB7
P2.6 1 SW 6|ADC DB6
P2.5 1 SW 5|ADC DB5
P2.4 1 SW 4|ADC DB4
P2.3 1 SW 3|ADC DB3
P2.2 1 SW 2|ADC DB2
P2.1 1 SW 1|ADC DB1
P2.0 1 SW 0|ADC DB0
P3.7 1 ADC RD|Comparator Output
P3.6 1 ADC WR
P3.5 1 Motor Sensor
P3.4 1 Display-select Input 1
P3.3 1 AND Gate Output|Display-se..t 0
P3.2 1 ADC INTR
P3.1 1 Motor Control Bit 1|Ext. UART Rx
P3.0 1 Motor Control Bit 0|Ext. UART Tx
```

Test Case 2: Our inputs N1 and N2 are 143 and 234 respectively. We can see N1 and N2 being stored in R0 and R1 as their hexadecimal representations. (8FH and EAH). The expected output was 82B6H which is shown in memory locations 50H and 51H.



EdSim51DI - Version 2.1.32 | assignment2.asm

```
System Clock (MHz) 12.0        1  Update Freq.
SBUF
R/O   W/O     TH0   TL0     R7 0x00      B 0x82
0x00  0x00    0x00  0x00    R6 0x00    ACC 0xB6
RXD   TXD                   R5 0x00    PSW 0x05
 1     1      TMOD  0x00    R4 0x03     IP 0x00
SCON  0x00    TCON  0x00    R3 0x04     IE 0x00
                            R2 0x00   PCON 0x00
pins  bits    TH1   TL1     R1 0xEA    DPH 0x00
0xFF  0xFF P3 0x00  0x00    R0 0x8F    DPL 0x37
0xFF  0xFF P2           8051            SP 0x07
0xFF  0xFF P1   PC
0xFF  0xFF P0   0x0081     PSW 0 0 0 0  0 1 0 1
                            Modify RAM
Data Memory            addr 0x00  0x00 value
```

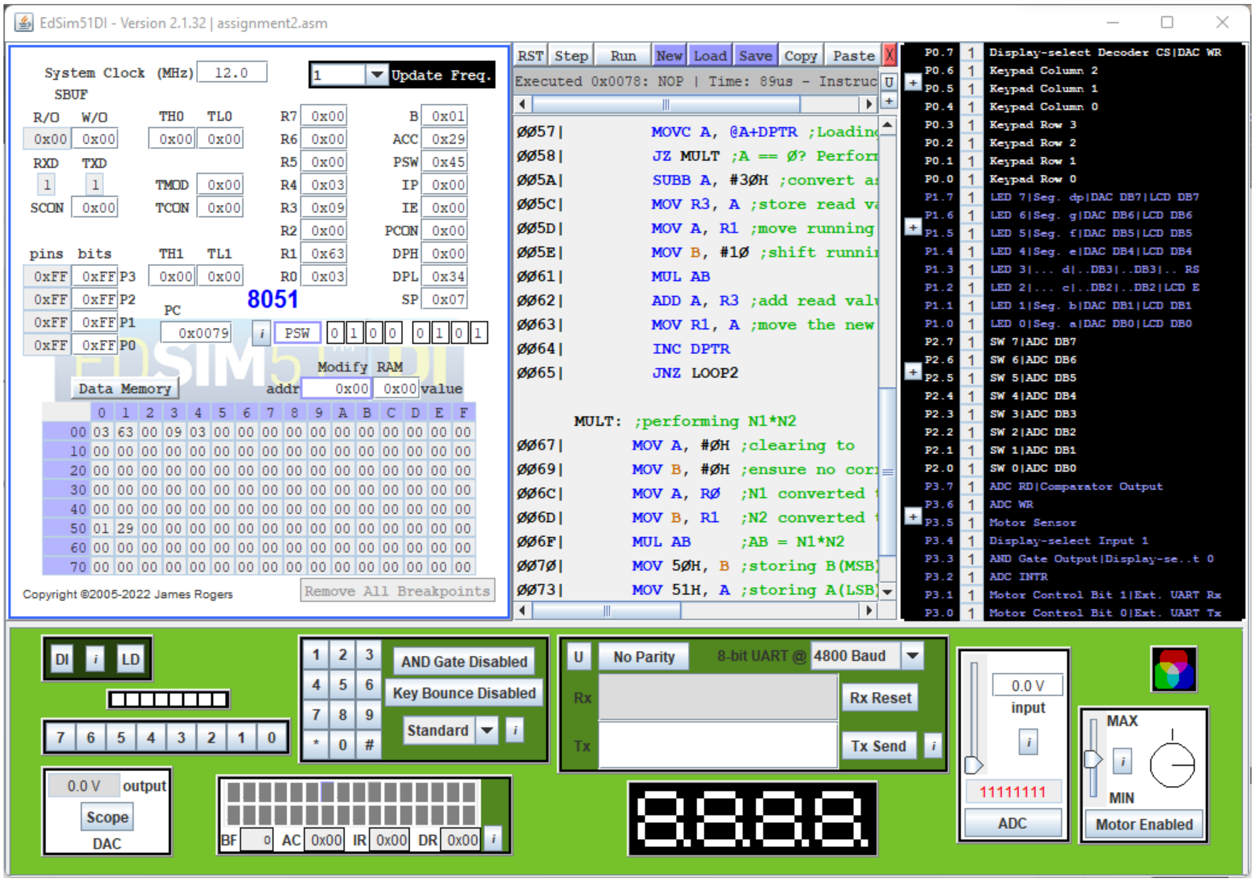
```
    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00 8F EA 00 04 03 00 00 00 00 00 00 00 00 00 00 00
10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
50 82 B6 00 00 00 00 00 00 00 00 00 00 00 00 00 00
60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
70 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Copyright ©2005-2022 James Rogers

```
Executed 0x0080: NOP | Time: 148us - Instru
ØØ5A|         MOVC A, @A+DPTR ;Loadin
ØØ5B|         JZ MULT ;A == Ø? Perfor
ØØ5D|         SUBB A, #3ØH ;convert a
ØØ5F|         MOV R3, A ;store read v
ØØ6Ø|         MOV A, R1 ;move running
ØØ61|         MOV B, #1Ø ;shift runni
ØØ64|         MUL AB
ØØ65|         ADD A, R3 ;add read val
ØØ66|         MOV R1, A ;move the new
ØØ67|         INC DPTR
ØØ68|         JNZ LOOP2

      MULT: ;performing N1*N2
ØØ6A|       MOV A, #ØH ;clearing to
ØØ6C|       MOV B, #ØH ;ensure no cor
ØØ6F|       MOV A, RØ  ;N1 converted
ØØ7Ø|       MOV B, R1  ;N2 converted
ØØ72|       MUL AB     ;AB = N1*N2
ØØ73|       MOV 5ØH, B ;storing B(MSB
ØØ76|       MOV 51H, A ;storing A(LSB
```

```
P0.7 1 Display-select Decoder CS|DAC WR
P0.6 1 Keypad Column 2
P0.5 1 Keypad Column 1
P0.4 1 Keypad Column 0
P0.3 1 Keypad Row 3
P0.2 1 Keypad Row 2
P0.1 1 Keypad Row 1
P0.0 1 Keypad Row 0
P1.7 1 LED 7|Seg. dp|DAC DB7|LCD DB7
P1.6 1 LED 6|Seg. g|DAC DB6|LCD DB6
P1.5 1 LED 5|Seg. f|DAC DB5|LCD DB5
P1.4 1 LED 4|Seg. e|DAC DB4|LCD DB4
P1.3 1 LED 3|... d|..DB3|..DB3|.. RS
P1.2 1 LED 2|... c|..DB2|..DB2|LCD E
P1.1 1 LED 1|Seg. b|DAC DB1|LCD DB1
P1.0 1 LED 0|Seg. a|DAC DB0|LCD DB0
P2.7 1 SW 7|ADC DB7
P2.6 1 SW 6|ADC DB6
P2.5 1 SW 5|ADC DB5
P2.4 1 SW 4|ADC DB4
P2.3 1 SW 3|ADC DB3
P2.2 1 SW 2|ADC DB2
P2.1 1 SW 1|ADC DB1
P2.0 1 SW 0|ADC DB0
P3.7 1 ADC RD|Comparator Output
P3.6 1 ADC WR
P3.5 1 Motor Sensor
P3.4 1 Display-select Input 1
P3.3 1 AND Gate Output|Display-se..t 0
P3.2 1 ADC INTR
P3.1 1 Motor Control Bit 1|Ext. UART Rx
P3.0 1 Motor Control Bit 0|Ext. UART Tx
```

Test Case 3: Our inputs N1 and N2 are 3 and 99 respectively. We can see N1 and N2 being stored in R0 and R1 as their hexadecimal representations. (03H and 63H). The expected output was 0129H which is shown in memory locations 50H and 51H.



System Clock (MHz) 12.0
Update Freq. 1

SBUF R/O 0x00 W/O 0x00
RXD 1 TXD 1
SCON 0x00
TH0 0x00 TL0 0x00
TMOD 0x00
TCON 0x00
TH1 0x00 TL1 0x00

pins	bits	
0xFF	0xFF	P3
0xFF	0xFF	P2
0xFF	0xFF	P1
0xFF	0xFF	P0

R7	0x00	B	0x01
R6	0x00	ACC	0x29
R5	0x00	PSW	0x45
R4	0x03	IP	0x00
R3	0x09	IE	0x00
R2	0x00	PCON	0x00
R1	0x63	DPH	0x00
R0	0x03	DPL	0x34
		SP	0x07

PC 0x0079
PSW 0 1 0 0 0 1 0 1

Modify RAM addr 0x00 0x00 value

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	03	63	00	09	03	00	00	00	00	00	00	00	00	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	01	29	00	00	00	00	00	00	00	00	00	00	00	00	00	00
60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

```
Executed 0x0078: NOP | Time: 89us - Instruc

ØØ57|         MOVC A, @A+DPTR ;Loadin
ØØ58|         JZ MULT ;A == Ø? Perfor
ØØ5A|         SUBB A, #3ØH ;convert a
ØØ5C|         MOV R3, A ;store read v
ØØ5D|         MOV A, R1 ;move running
ØØ5E|         MOV B, #1Ø ;shift runni
ØØ61|         MUL AB
ØØ62|         ADD A, R3 ;add read val
ØØ63|         MOV R1, A ;move the new
ØØ64|         INC DPTR
ØØ65|         JNZ LOOP2

      MULT: ;performing N1*N2
ØØ67|       MOV A, #ØH ;clearing to
ØØ69|       MOV B, #ØH ;ensure no cor
ØØ6C|       MOV A, RØ   ;N1 converted
ØØ6D|       MOV B, R1   ;N2 converted
ØØ6F|       MUL AB      ;AB = N1*N2
ØØ7Ø|       MOV 5ØH, B ;storing B(MSB)
ØØ73|       MOV 51H, A ;storing A(LSB)
```

P0.7	1	Display-select Decoder CS|DAC WR
P0.6	1	Keypad Column 2
P0.5	1	Keypad Column 1
P0.4	1	Keypad Column 0
P0.3	1	Keypad Row 3
P0.2	1	Keypad Row 2
P0.1	1	Keypad Row 1
P0.0	1	Keypad Row 0
P1.7	1	LED 7|Seg. dp|DAC DB7|LCD DB7
P1.6	1	LED 6|Seg. g|DAC DB6|LCD DB6
P1.5	1	LED 5|Seg. f|DAC DB5|LCD DB5
P1.4	1	LED 4|Seg. e|DAC DB4|LCD DB4
P1.3	1	LED 3|... d|..DB3|..DB3|.. RS
P1.2	1	LED 2|... c|..DB2|..DB2|LCD E
P1.1	1	LED 1|Seg. b|DAC DB1|LCD DB1
P1.0	1	LED 0|Seg. a|DAC DB0|LCD DB0
P2.7	1	SW 7|ADC DB7
P2.6	1	SW 6|ADC DB6
P2.5	1	SW 5|ADC DB5
P2.4	1	SW 4|ADC DB4
P2.3	1	SW 3|ADC DB3
P2.2	1	SW 2|ADC DB2
P2.1	1	SW 1|ADC DB1
P2.0	1	SW 0|ADC DB0
P3.7	1	ADC RD|Comparator Output
P3.6	1	ADC WR
P3.5	1	Motor Sensor
P3.4	1	Display-select Input 1
P3.3	1	AND Gate Output|Display-se..t 0
P3.2	1	ADC INTR
P3.1	1	Motor Control Bit 1|Ext. UART Rx
P3.0	1	Motor Control Bit 0|Ext. UART Tx

4800 Baud
input 0.0 V
11111111
output 0.0 V
BF 0 AC 0x00 IR 0x00 DR 0x00