**Exp No:** 12 **Date:** 01/11/2020

**Name:** Swetha Saseendran

**Reg No:** 185001183

# 8-BIT ARITHMETIC OPERATIONS USING 8051

## Aim:

To program and execute 8-bit arithmetic operations using 8051 microcontrollers using EDSim.

## Programs:

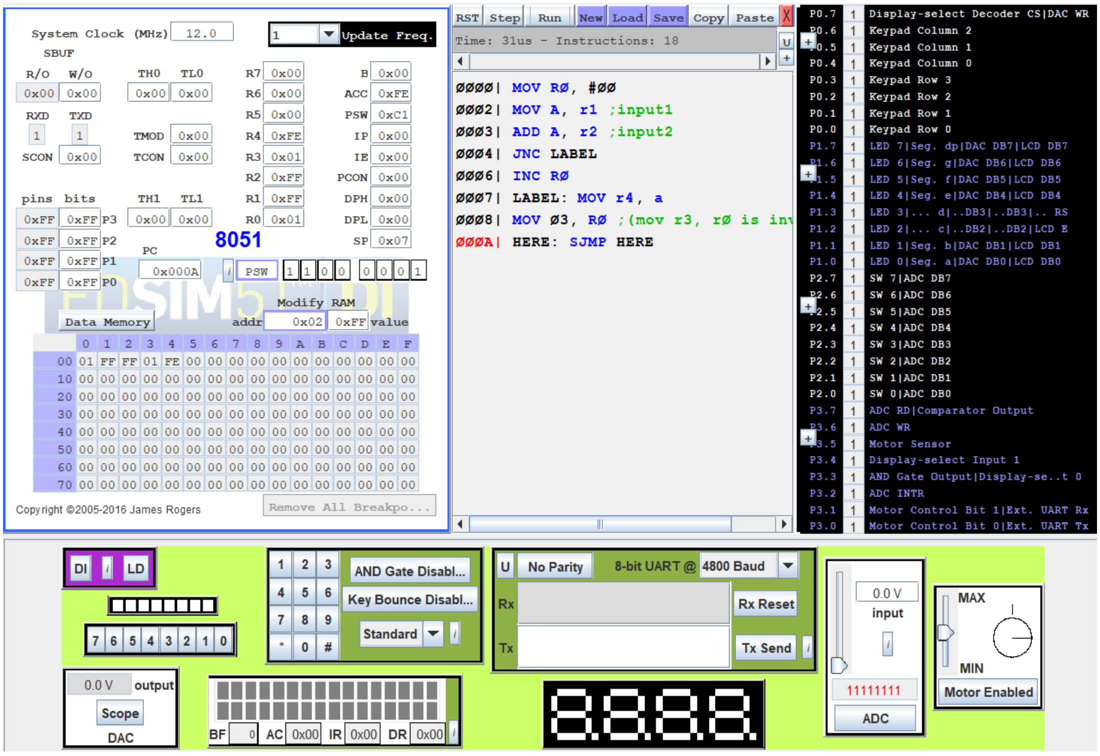
### (i) 8-BIT ADDITION

## Algorithm:

- Move input1 to A.
- Add the second input to A and store the result in A.
- Jump if no carry to label.
- Increment register 0.
- Label: mov the result to register 4 and carry to register 3.
- Here: short jump here.

| *Program* | *Comment* |
|---|---|
| *;Program to subtract 2 8 bit numbers using 8051 microcontroller* | |
| *MOV R0, #00* | Move value 00 to R0. |
| *MOV A, r1* | Move input1 to A |
| *ADD A, r2* | Add A and 2nd input and store in A |
| *JNC LABEL* | Jump if no carry to label |
| *INC R0* | Increment R0 |
| ***LABEL:** MOV r4, a* | Move result to R4 |
| *MOV 03, R0 ;(mov r3, r0 is invalid)* | Move carry to R3 |
| ***HERE:** SJMP HERE* | End |

## Snapshot of sample input and output:



System Clock (MHz) 12.0 | 1 Update Freq.

Time: 31us - Instructions: 18

```
ØØØØ| MOV RØ, #ØØ
ØØØ2| MOV A, r1 ;input1
ØØØ3| ADD A, r2 ;input2
ØØØ4| JNC LABEL
ØØØ6| INC RØ
ØØØ7| LABEL: MOV r4, a
ØØØ8| MOV Ø3, RØ ;(mov r3, rØ is inv
ØØØA| HERE: SJMP HERE
```

```
SBUF
R/O   W/O      TH0   TL0       R7 0x00      B 0x00
0x00  0x00     0x00  0x00      R6 0x00    ACC 0xFE
RXD   TXD                      R5 0x00    PSW 0xC1
1     1        TMOD  0x00      R4 0xFE     IP 0x00
SCON  0x00     TCON  0x00      R3 0x01     IE 0x00
                               R2 0xFF   PCON 0x00
pins  bits     TH1   TL1       R1 0xFF    DPH 0x00
0xFF  0xFF P3  0x00  0x00      R0 0x01    DPL 0x00
0xFF  0xFF P2                  8051        SP 0x07
0xFF  0xFF P1  PC
0xFF  0xFF P0  0x000A          i PSW 1 1 0 0  0 0 0 1

Data Memory                    Modify RAM  addr 0x02  0xFF value
```

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 01 | FF | FF | 01 | FE | 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 | 00 | 00 | 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 ©2005-2016 James Rogers

```
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
```

## (ii) 8-BIT SUBTRACTION

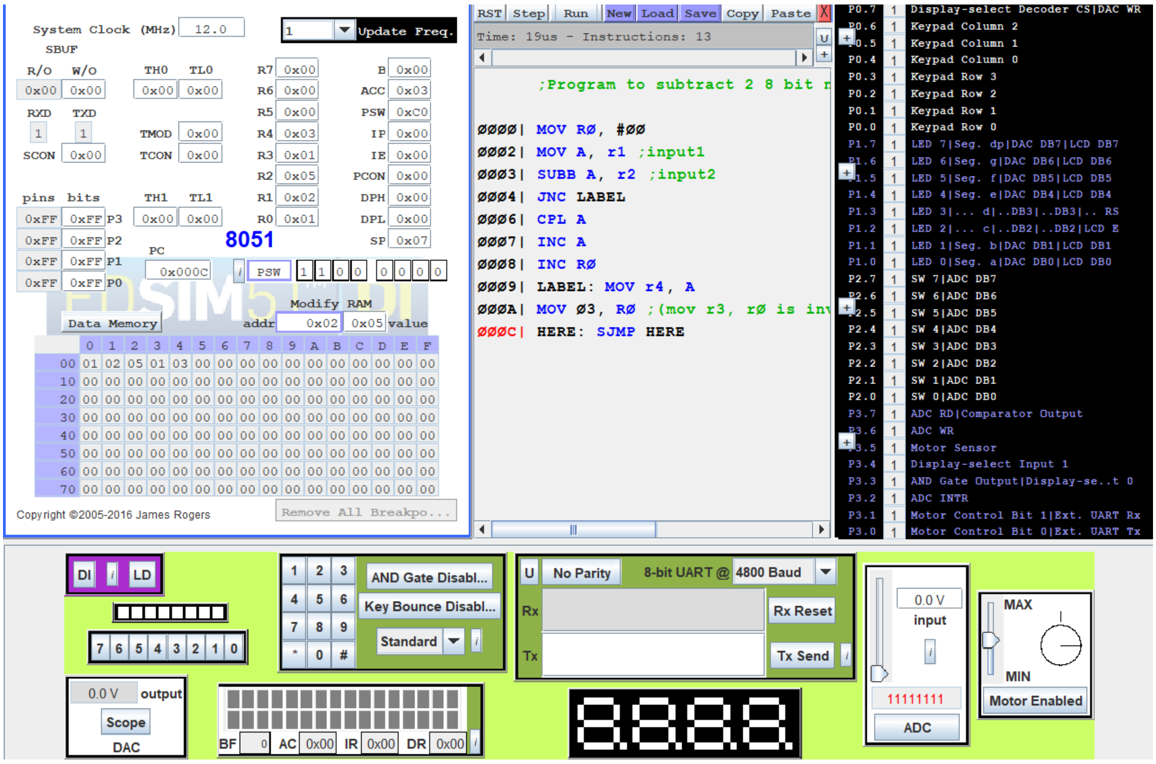
## Algorithm:

- Move input1 to A.
- Add the second input to A and store the result in A.
- Jump if no carry to label.
- Increment register 0.
- Label: mov the result to register 4 and carry to register 3.
- Here: short jump here.

| *Program* | *Comment* |
|---|---|
| *;Program to subtract 2 8 bit numbers using 8051 microcontroller* | |
| *MOV R0, #00*<br>*MOV A, r1*<br>*SUBB A, r2*<br>*JNC LABEL* | Move value 00 to R0.<br>Move input1 to A<br>Subtract A and 2$^{nd}$ input and store in A |

| | |
|---|---|
| *CPL A* | Jump if no carry to label |
| *INC A* | complement A |
| *INC R0* | Increment A |
| *LABEL: MOV r4, A* | Increment R0 |
| *MOV 03, R0* | Move result to R4 |
| ***HERE:** SJMP HERE* | Move carry to R3 |
| | End |

## Snapshot of sample input and output:



Time: 19us - Instructions: 13

```
      ;Program to subtract 2 8 bit r

ØØØØ| MOV RØ, #ØØ
ØØØ2| MOV A, r1 ;input1
ØØØ3| SUBB A, r2 ;input2
ØØØ4| JNC LABEL
ØØØ6| CPL A
ØØØ7| INC A
ØØØ8| INC RØ
ØØØ9| LABEL: MOV r4, A
ØØØA| MOV Ø3, RØ ;(mov r3, rØ is inv
ØØØC| HERE: SJMP HERE
```

R7 0x00, R6 0x00, R5 0x00, R4 0x03, R3 0x01, R2 0x05, R1 0x02, R0 0x01

B 0x00, ACC 0x03, PSW 0xC0, IP 0x00, IE 0x00, PCON 0x00, DPH 0x00, DPL 0x00, SP 0x07

PC 0x000C

Data Memory 00: 01 02 05 01 03 00 00 00 00 00 00 00 00 00 00 00

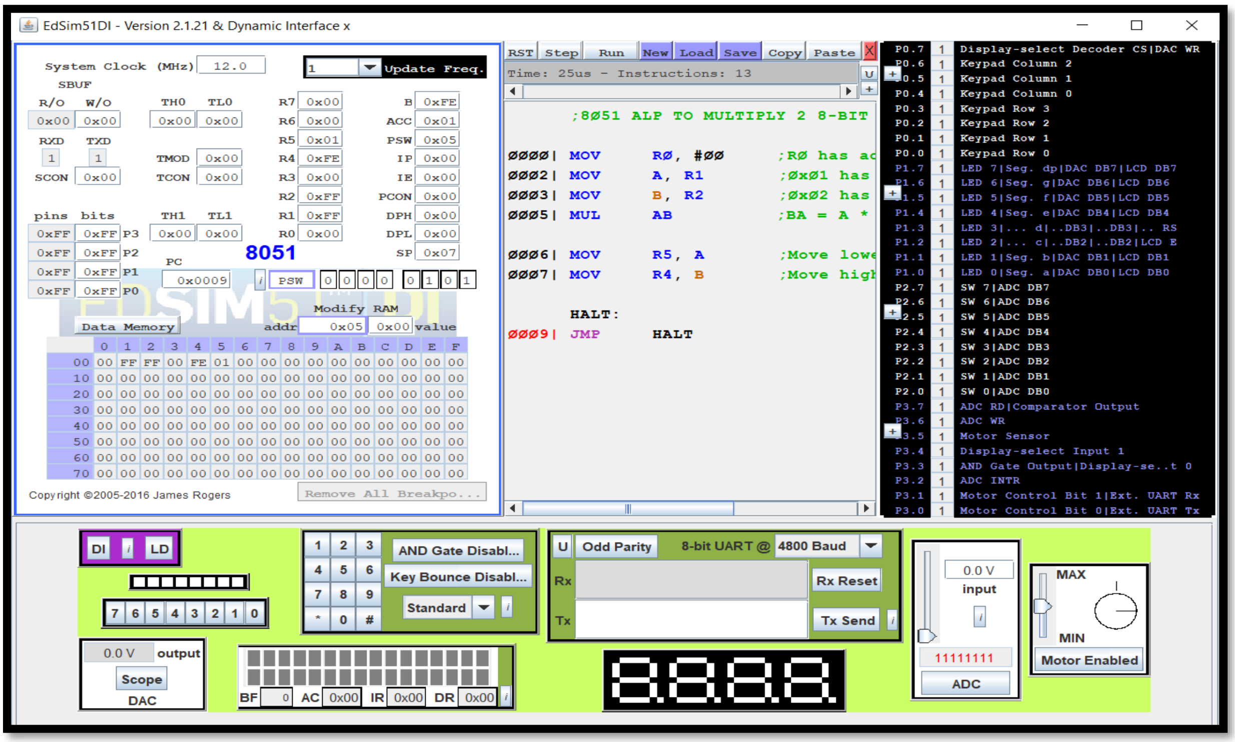
# (iii) 8-BIT MULTIPLICATION

## Algorithm:

- Initialize R0 with 00h
- Move the value in R1 to A.
- Move the value in R2 to B.
- Multiply A and B.
- Move B to R4 (MSB of product) and A to R5 (LSB of product)

| *PROGRAM* | *COMMENTS* |
|---|---|
| *MOV  R0, #00* | R0 has address of 0x00 |
| *MOV  A, R1* | 0x01 has 1st 8-bit number |
| *MOV  B, R2* | 0x02 has 2nd 8-bit number |
| *MUL  AB* | BA = A * B |
| *MOV  R5, A* | Move lower byte to R5 from A |
| *MOV  R4, B* | Move higher byte to R4 from B |
| ***HALT:*** | |
| *SJMP  HALT* | Halt the program with a loop. |

## Snapshot of sample input and output:



EdSim51DI - Version 2.1.21 & Dynamic Interface x

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| System Clock (MHz) | 12.0 | | | | | Update Freq. | 1 |
| SBUF R/O | 0x00 | SBUF W/O | 0x00 | TH0 | 0x00 | TL0 | 0x00 |
| RXD | 1 | TXD | 1 | TMOD | 0x00 | TCON | 0x00 |
| SCON | 0x00 | | | TH1 | 0x00 | TL1 | 0x00 |
| P3 pins | 0xFF | P3 bits | 0xFF | PC | 0x0009 | PSW | 0 0 0 0 0 1 0 1 |
| P2 pins | 0xFF | P2 bits | 0xFF | R7 | 0x00 | B | 0xFE |
| P1 pins | 0xFF | P1 bits | 0xFF | R6 | 0x00 | ACC | 0x01 |
| P0 pins | 0xFF | P0 bits | 0xFF | R5 | 0x01 | PSW | 0x05 |
| | | | | R4 | 0xFE | IP | 0x00 |
| | | | | R3 | 0x00 | IE | 0x00 |
| | | | | R2 | 0xFF | PCON | 0x00 |
| | | | | R1 | 0xFF | DPH | 0x00 |
| | | | | R0 | 0x00 | DPL | 0x00 |
| | | | | | | SP | 0x07 |

Modify RAM addr 0x05 value 0x00

Data Memory

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 00 | FF | FF | 00 | FE | 01 | 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 | 00 | 00 | 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 ©2005-2016 James Rogers

Time: 25us - Instructions: 13

```
        ;8Ø51 ALP TO MULTIPLY 2 8-BIT

ØØØØ| MOV     RØ, #ØØ         ;RØ has ac
ØØØ2| MOV     A, R1           ;ØxØ1 has
ØØØ3| MOV     B, R2           ;ØxØ2 has
ØØØ5| MUL     AB              ;BA = A *

ØØØ6| MOV     R5, A           ;Move lowe
ØØØ7| MOV     R4, B           ;Move high

      HALT:
ØØØ9| JMP     HALT
```

```
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
```

BF 0 AC 0x00 IR 0x00 DR 0x00

Odd Parity, 8-bit UART @ 4800 Baud

DAC output 0.0 V; ADC input 0.0 V, 11111111

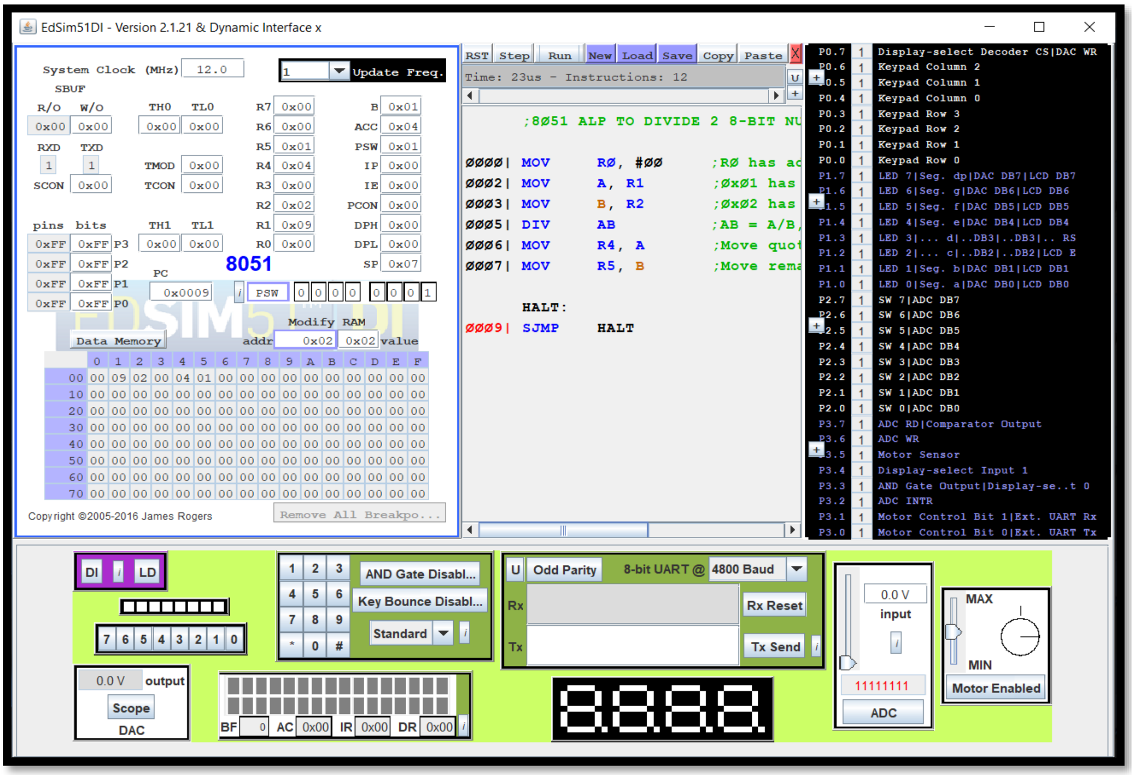
# (iv) 8-BIT DIVISION

## Algorithm:

- Initialize R0 with 00h.
- Move the value in R1 to A.
- Move the value in R2 to B.
- Divide A by B.
- Move A to R4 (Quotient) and B to R5 (Remainder)

| *PROGRAM* | *COMMENTS* |
|---|---|
| *MOV  R0, #00* | R0 has address of 0x00 |
| *MOV  A, R1* | 0x01 has 1st 8-bit number |
| *MOV  B, R2* | 0x02 has 2nd 8-bit number |
| *DIV  AB* | BA = A / B, A: Quotient, B: Remainder |
| *MOV  R5, A* | Move quotient to R4 from A |
| *MOV  R4, B* | Move remainder to R5 from B |
| ***HALT:*** | |
| *SJMP  HALT* | Halt the program with a loop. |

## Snapshot of sample input and output



EdSim51DI - Version 2.1.21 & Dynamic Interface x

```
System Clock (MHz) 12.0        Update Freq. 1

R/O 0x00   W/O 0x00    TH0 0x00   TL0 0x00
RXD 1      TXD 1       TMOD 0x00
SCON 0x00              TCON 0x00
P3 pins 0xFF bits 0xFF TH1 0x00   TL1 0x00
P2 pins 0xFF bits 0xFF
P1 pins 0xFF bits 0xFF
P0 pins 0xFF bits 0xFF PC 0x0009

R7 0x00   B    0x01
R6 0x00   ACC  0x04
R5 0x01   PSW  0x01
R4 0x04   IP   0x00
R3 0x00   IE   0x00
R2 0x02   PCON 0x00
R1 0x09   DPH  0x00
R0 0x00   DPL  0x00
          SP   0x07
PSW 0 0 0 0 0 0 0 1
Modify RAM addr 0x02 value 0x02
```

Data Memory

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 00 | 09 | 02 | 00 | 04 | 01 | 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 | 00 | 00 | 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 ©2005-2016 James Rogers

Time: 23us - Instructions: 12

```
        ;8Ø51 ALP TO DIVIDE 2 8-BIT NU

ØØØØ| MOV     RØ, #ØØ      ;RØ has ac
ØØØ2| MOV     A, R1        ;ØxØ1 has
ØØØ3| MOV     B, R2        ;ØxØ2 has
ØØØ5| DIV     AB           ;AB = A/B,
ØØØ6| MOV     R4, A        ;Move quot
ØØØ7| MOV     R5, B        ;Move rema

      HALT:
ØØØ9| SJMP    HALT
```

```
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
```

Odd Parity  8-bit UART @ 4800 Baud

DAC output 0.0 V  ADC input 0.0 V  11111111

BF 0  AC 0x00  IR 0x00  DR 0x00

## Result:

The assembly level programs were written to perform the above specified 8-bit arithmetic operations using an 8051 microcontroller and the outputs were verified.

**Exp No:** 13 **Date:** 01/11/2020
**Name:** Swetha Saseendran
**Reg No:** 185001183

# CUBE OF A NUMBER

## Aim:

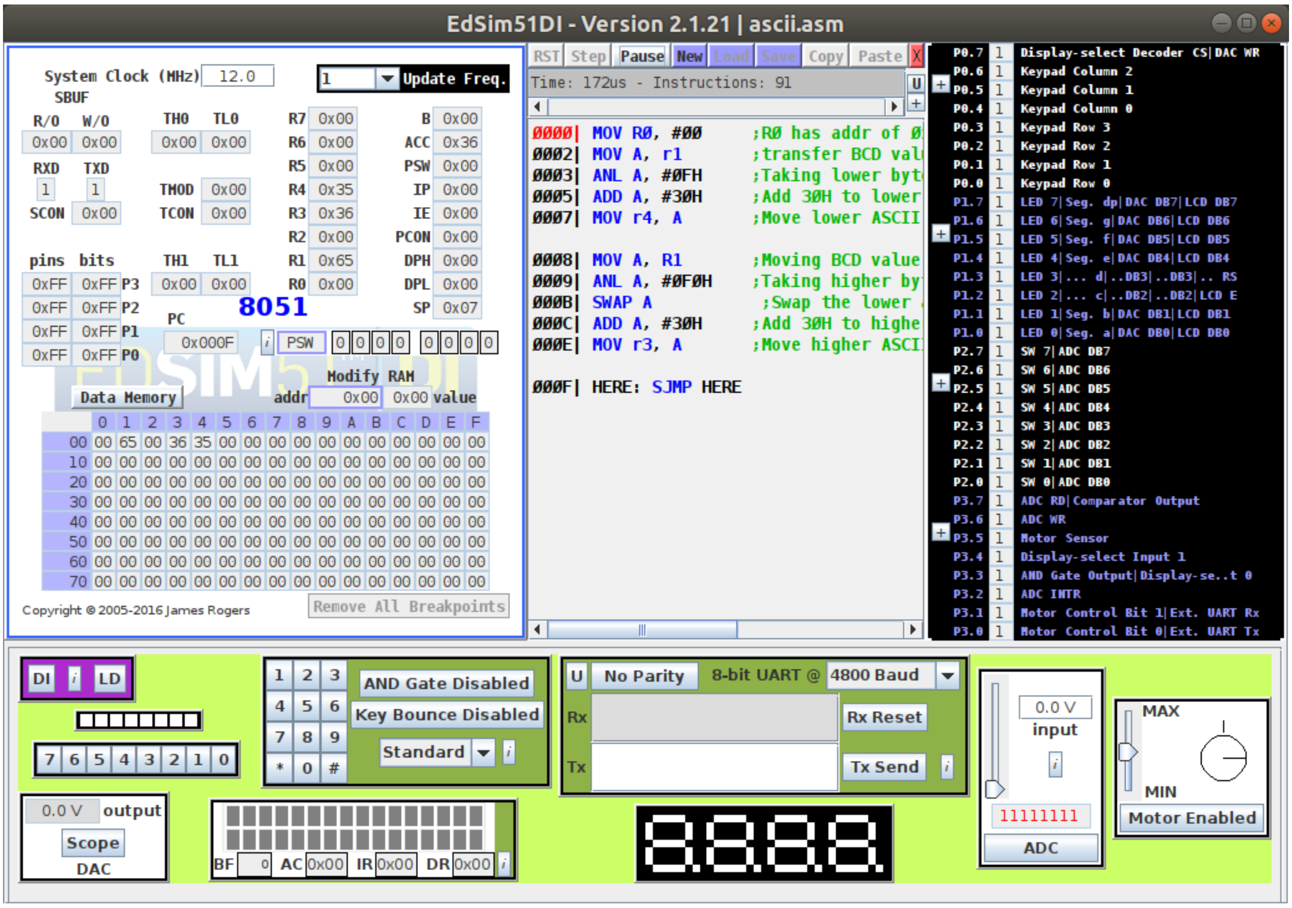
To write an Assembly language program that converts BCDvalue to its corresponding ascii value using an 8051 micro controller.

## Algorithm:

- Move the value in R1 to A.
- Get the lower byte at A by performing logical AND over A & 0F.
- Add 30h to A.
- Move A to R4.
- Move the value in R1 to A.
- Get the higher byte at Aby performing logical AND over A & F0.
- Swap the lower and higher nibble in A.
- Add 30H to A.
- Move A to R3.

| *Program* | *Comment* |
|---|---|
| *MOV R0, #00* | R0 has addr of 0x00 |
| *MOV A, r1* | transfer BCD value to A |
| *ANL A, #0FH* | Taking lower byte value of A |
| *ADD A, #30H* | Add 30H to lower byte to convert it to ASCII |
| *MOV r4, A* | Move lower ASCII byte to R4 from A |
| | |
| *MOV A, R1* | Moving BCD value again to A |
| *ANL A, #0F0H* | Taking higher byte value of A |
| *SWAP A* | Swap the lower and higher nibble in A |
| *ADD A, #30H* | Add 30H to higher byte to convert it to ASCII |
| *MOV r3, A* | Move higher ASCII byte to R3 from A |
| | |
| ***HERE:** SJMP HERE* | |

## Snapshot of sample input and output:



EdSim51DI - Version 2.1.21 | ascii.asm

Time: 172us - Instructions: 91

```
0000| MOV R0, #00        ;R0 has addr of 0
0002| MOV A, r1          ;transfer BCD valu
0003| ANL A, #0FH        ;Taking lower byte
0005| ADD A, #30H        ;Add 30H to lower
0007| MOV r4, A          ;Move lower ASCII

0008| MOV A, R1          ;Moving BCD value
0009| ANL A, #0F0H       ;Taking higher by
000B| SWAP A              ;Swap the lower
000C| ADD A, #30H        ;Add 30H to highe
000E| MOV r3, A          ;Move higher ASCI

000F| HERE: SJMP HERE
```

```
System Clock (MHz) 12.0                1  Update Freq.
SBUF
R/O   W/O       TH0   TL0      R7 0x00        B 0x00
0x00  0x00      0x00  0x00     R6 0x00      ACC 0x36
RXD   TXD                      R5 0x00      PSW 0x00
1     1         TMOD  0x00     R4 0x35       IP 0x00
SCON  0x00      TCON  0x00     R3 0x36       IE 0x00
                               R2 0x00     PCON 0x00
pins  bits      TH1   TL1      R1 0x65      DPH 0x00
0xFF  0xFF P3   0x00  0x00     R0 0x00      DPL 0x00
0xFF  0xFF P2              8051              SP 0x07
0xFF  0xFF P1   PC
0xFF  0xFF P0   0x000F     PSW 0 0 0 0  0 0 0 0
                                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 | 00 | 65 | 00 | 36 | 35 | 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 | 00 | 00 | 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 © 2005-2016 James Rogers

## Result:

An assembly level program was written to calculate the cube of a given 8-bit number using an 8051 micro controller and the output was verified.

**Exp No:** 14 **Date:** 01/11/2020
**Name:** Swetha Saseendran
**Reg No:** 185001183

# COVERSION OF BCD TO ASCII

## Aim:

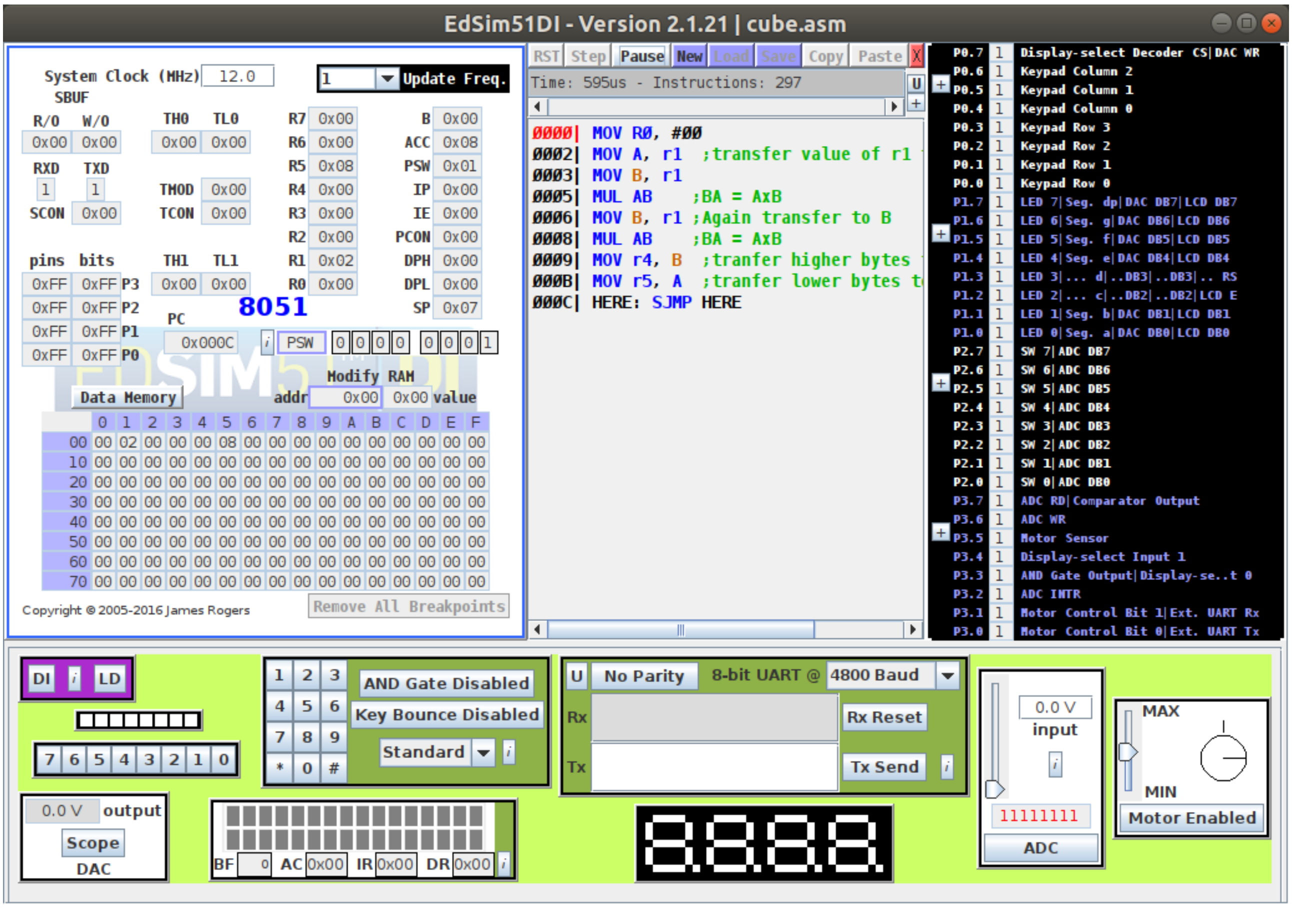
To write an assembly language program to calculate the cube of an 8-bit number using an 8051 micro controller.

## Algorithm:

- Initialize R0 with 00h.
- Move the value in R1 to A.
- Move the value in R1 to B.
- Multiply A and B.
- Move the value in R1 to B.
- Multiply A and B.
- Move B to R4 (MSB of cube) and A to R5 (LSB of cube)

| *Program* | *Comment* |
|---|---|
| *MOV R0, #00* | R0 has address of 0x00 |
| *MOV A, R1* | Transferring 8-bit number to reg A |
| *MOV B, R1* | Transferring 8-bit number to reg B |
| *MUL AB* | BA = A x B |
| | B is empty since bit multiplication |
| *MOV B, R1* | Transfer 8-bit value to B |
| *MUL AB* | BA = A x B |
| *MOV R5, A* | Moving lower byte to R5 |
| *MOV R4, B* | Moving higher byte to R4 |
| | |
| ***HERE:*** *SJMP HALT* | Halt the program with a loop. |

## Snapshot of sample input and output:



EdSim51DI - Version 2.1.21 | cube.asm

Time: 595us - Instructions: 297

```
0000| MOV R0, #00
0002| MOV A, r1  ;transfer value of r1
0003| MOV B, r1
0005| MUL AB     ;BA = AxB
0006| MOV B, r1 ;Again transfer to B
0008| MUL AB     ;BA = AxB
0009| MOV r4, B  ;tranfer higher bytes
000B| MOV r5, A  ;tranfer lower bytes t
000C| HERE: SJMP HERE
```

```
System Clock (MHz) 12.0
R7 0x00    B    0x00
R6 0x00    ACC  0x08
R5 0x08    PSW  0x01
R4 0x00    IP   0x00
R3 0x00    IE   0x00
R2 0x00    PCON 0x00
R1 0x02    DPH  0x00
R0 0x00    DPL  0x00
           SP   0x07
PC 0x000C  PSW 0000 0001
P3 0xFF  P2 0xFF  P1 0xFF  P0 0xFF
Data Memory 00: 00 02 00 00 00 08 00 00 00 00 00 00 00 00 00 00
```

## Result:

An assembly level program was written to convert a given BCD value to its corresponding ASCII value using an 8051 microcontroller and the output was verified.