8 BIT ARITHMETIC OPERATIONS USING 8051

Exp No: 12	Name: Srinithyee S K
Date: 23-10-20	Register Number: 185001166

AIM:

To write assembly language programs to perform the following arithmetic operations using an 8051 microcontroller:

1. 8-bit addition
2. 8-bit subtraction
3. 8-bit multiplication
4. 8-bit division

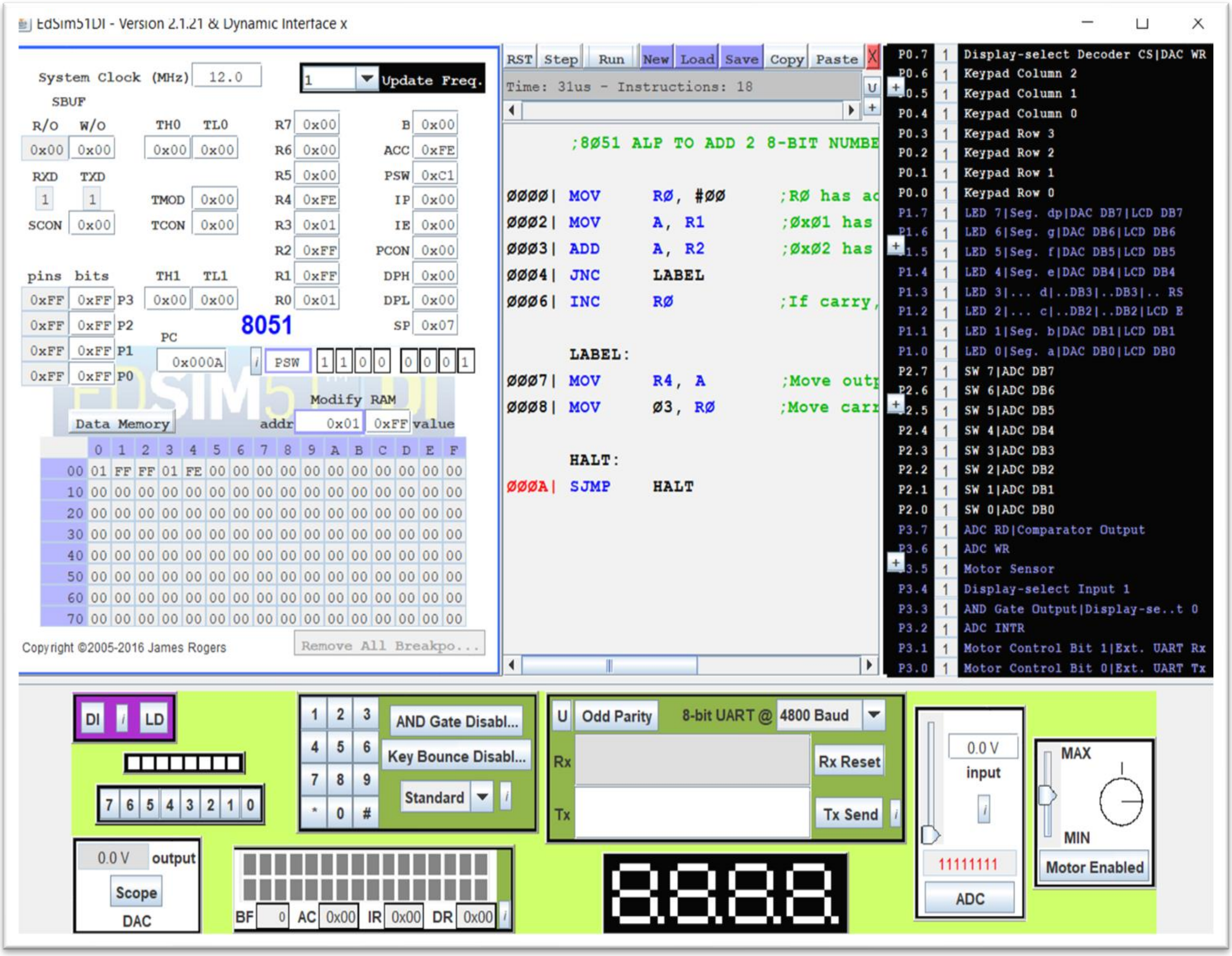
PROGRAM 1: 8 BIT ADDITION

ALGORITHM:

1. Begin
2. Initialize R0 with 00h.
3. Move the value in R1 to A.
4. Add the value in A to with value in R2.
5. Increment R0 if carry is produced.
6. Move R0 to R3 (carry) and A to R4 (sum).
7. End.

PROGRAM	COMMENTS
MOV R0, #00	R0 has address of 0x00
MOV A, R1	0x01 has 1st 8-bit number
ADD A, R2	0x02 has 2nd 8-bit number. Add it with A
JNC LABEL	If no carry, jump to “LABEL”.
INC R0	If carry, increment R0
LABEL:	
MOV R4, A	Move output to R4 from A
MOV 03, R0	Move carry to R3. (MOV R3, R0) is invalid
HALT:	
SJMP HALT	Halt the program with a loop.

SAMPLE I/O SNAPSHOT:



EdSim51DI - Version 2.1.21 & Dynamic Interface x

System Clock (MHz) 12.0 | Update Freq. 1

SBUF R/O	SBUF W/O	TH0	TL0	TMOD	TCON	SCON
0x00	0x00	0x00	0x00	0x00	0x00	0x00

RXD 1 TXD 1

	pins	bits	TH1	TL1
P3	0xFF	0xFF	0x00	0x00
P2	0xFF	0xFF		
P1	0xFF	0xFF		
P0	0xFF	0xFF		

Register	Value	Register	Value
R7	0x00	B	0x00
R6	0x00	ACC	0xFE
R5	0x00	PSW	0xC1
R4	0xFE	IP	0x00
R3	0x01	IE	0x00
R2	0xFF	PCON	0x00
R1	0xFF	DPH	0x00
R0	0x01	DPL	0x00
		SP	0x07

8051

PC 0x000A

PSW 1 1 0 0 0 0 0 1

Modify RAM addr 0x01 value 0xFF

Data Memory

	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

Time: 31us - Instructions: 18

```
      ;8Ø51 ALP TO ADD 2 8-BIT NUMBE

ØØØØ| MOV     RØ, #ØØ       ;RØ has ac
ØØØ2| MOV     A, R1         ;ØxØ1 has
ØØØ3| ADD     A, R2         ;ØxØ2 has
ØØØ4| JNC     LABEL
ØØØ6| INC     RØ            ;If carry,

      LABEL:
ØØØ7| MOV     R4, A         ;Move outp
ØØØ8| MOV     Ø3, RØ        ;Move carr

      HALT:
ØØØA| 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

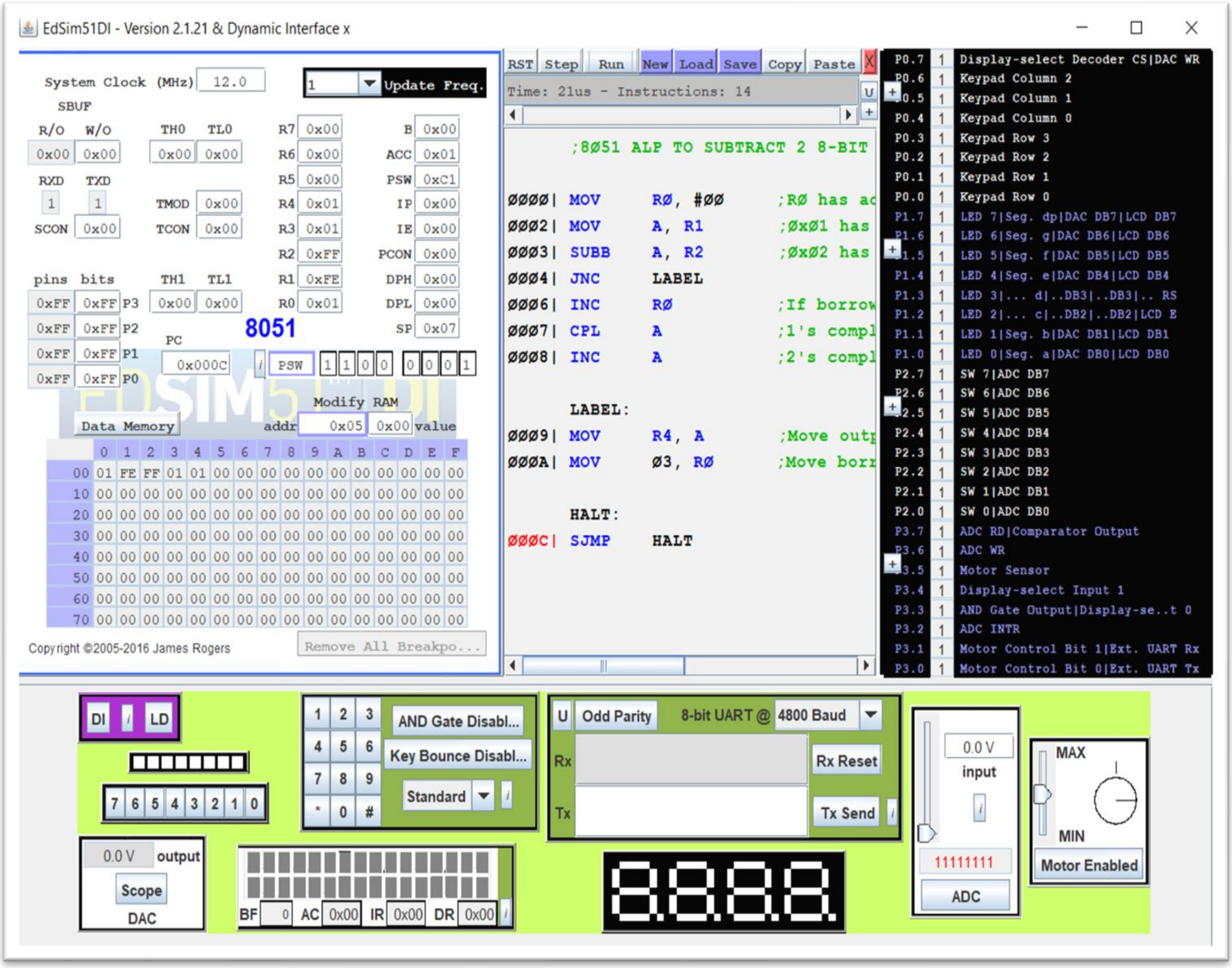
PROGRAM – 2: 8-BIT SUBTRACTION

ALGORITHM:

1. Begin.
2. Initialize R0 with 00h
3. Move the value in R1 to A.
4. Subtract the value in A to with value in R2.
5. Increment R0 if carry is produced and take 2's complement of A.
6. Move R0 to R3 (borrow) and A to R4 (difference)
7. End.

PROGRAM	COMMENTS
MOV R0, #00	R0 has address of 0x00
MOV A, R1	0x01 has 1st 8-bit number
SUBB A, R2	;0x02 has 2nd 8-bit number. Subtract it from A.
JNC LABEL	If no carry, jump to "LABEL".
INC R0	If carry, increment R0
CPL A	1's complement the difference
INC A	2's complement the difference
LABEL:	
MOV R4, A	Move output to R4 from A
MOV 03, R0	Move carry to R3. (MOV R3, R0) is invalid
HALT:	
SJMP HALT	Halt the program with a loop.

SAMPLE I/O SNAPSHOT:



EdSim51DI - Version 2.1.21 & Dynamic Interface x

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 0x01
                               R5 0x00      PSW 0xC1
RXD   TXD                      R4 0x01       IP 0x00
 1     1       TMOD 0x00       R3 0x01       IE 0x00
SCON 0x00      TCON 0x00       R2 0xFF     PCON 0x00
                               R1 0xFE      DPH 0x00
pins  bits     TH1   TL1       R0 0x01      DPL 0x00
0xFF  0xFF P3  0x00  0x00
0xFF  0xFF P2              8051              SP 0x07
0xFF  0xFF P1   PC
0xFF  0xFF P0   0x000C       PSW 1 1 0 0  0 0 0 1

Modify RAM  addr 0x05  0x00 value
```

Data Memory

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	01	FE	FF	01	01	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

Time: 21us - Instructions: 14

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

ØØØØ| MOV     RØ, #ØØ       ;RØ has ac
ØØØ2| MOV     A, R1         ;Øx01 has
ØØØ3| SUBB    A, R2         ;ØxØ2 has
ØØØ4| JNC     LABEL
ØØØ6| INC     RØ            ;If borrow
ØØØ7| CPL     A             ;1's compl
ØØØ8| INC     A             ;2's compl

      LABEL:
ØØØ9| MOV     R4, A         ;Move outp
ØØØA| MOV     Ø3, RØ        ;Move borr

      HALT:
ØØØC| 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
```

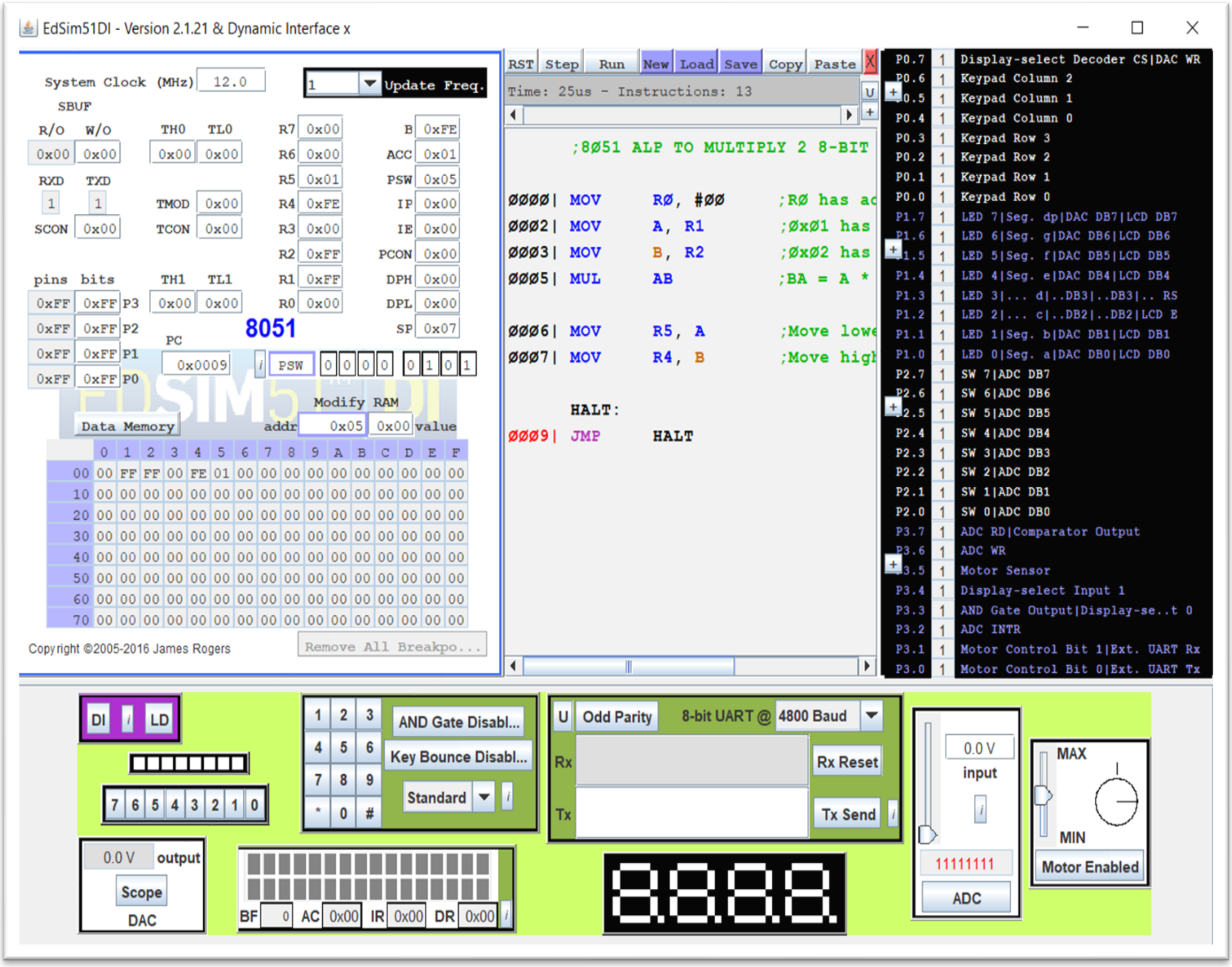
PROGRAM – 3: 8-BIT MULTIPLICATION

ALGORITHM:

1. Begin.
2. Initialize R0 with 00h
3. Move the value in R1 to A.
4. Move the value in R2 to B.
5. Multiply A and B.
6. Move B to R4 (MSB of product) and A to R5 (LSB of product)
7. End.

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.

SAMPLE I/O SNAPSHOT:



EdSim51DI - Version 2.1.21 & Dynamic Interface x

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

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

R7 0x00    B    0xFE
R6 0x00    ACC  0x01
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

8051
PC 0x0009    PSW 0 0 0 0 0 1 0 1
Modify RAM  addr 0x05  0x00 value
```

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

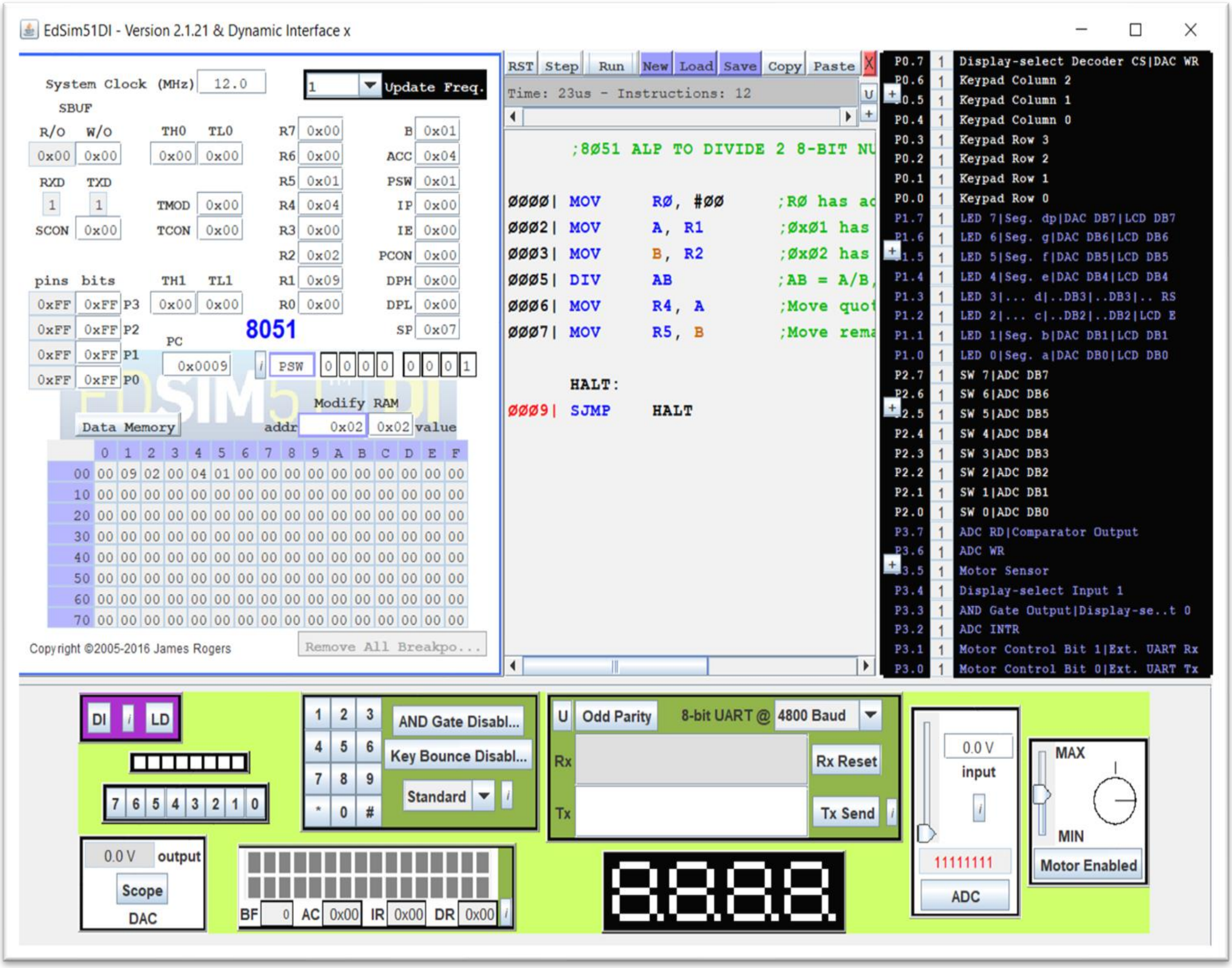
PROGRAM – 4: 8-BIT DIVISION

ALGORITHM:

1. Begin.
2. Initialize R0 with 00h.
3. Move the value in R1 to A.
4. Move the value in R2 to B.
5. Divide A by B.
6. Move A to R4 (quotient) and B to R5 (remainder)
7. End.

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.

SAMPLE I/O SNAPSHOT:



EdSim51DI - Version 2.1.21 & Dynamic Interface x

System Clock (MHz) 12.0 | 1 Update Freq.

SBUF R/O	SBUF W/O	TH0	TL0		
0x00	0x00	0x00	0x00		

RXD 1 TXD 1 — SCON 0x00 — TMOD 0x00 — TCON 0x00

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

TH1 0x00 TL1 0x00 — PC 0x0009

Register	Value	Register	Value
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

8051

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
```

DAC output 0.0 V — BF 0 AC 0x00 IR 0x00 DR 0x00 — Odd Parity — 8-bit UART @ 4800 Baud — Standard — ADC input 0.0 V, 11111111 — Motor Enabled

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.

CUBE OF A NUMBER USING 8051

Exp No: 13

Date: 23-10-20

Name: Srinithyee S K

Register Number: 185001166

AIM:

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

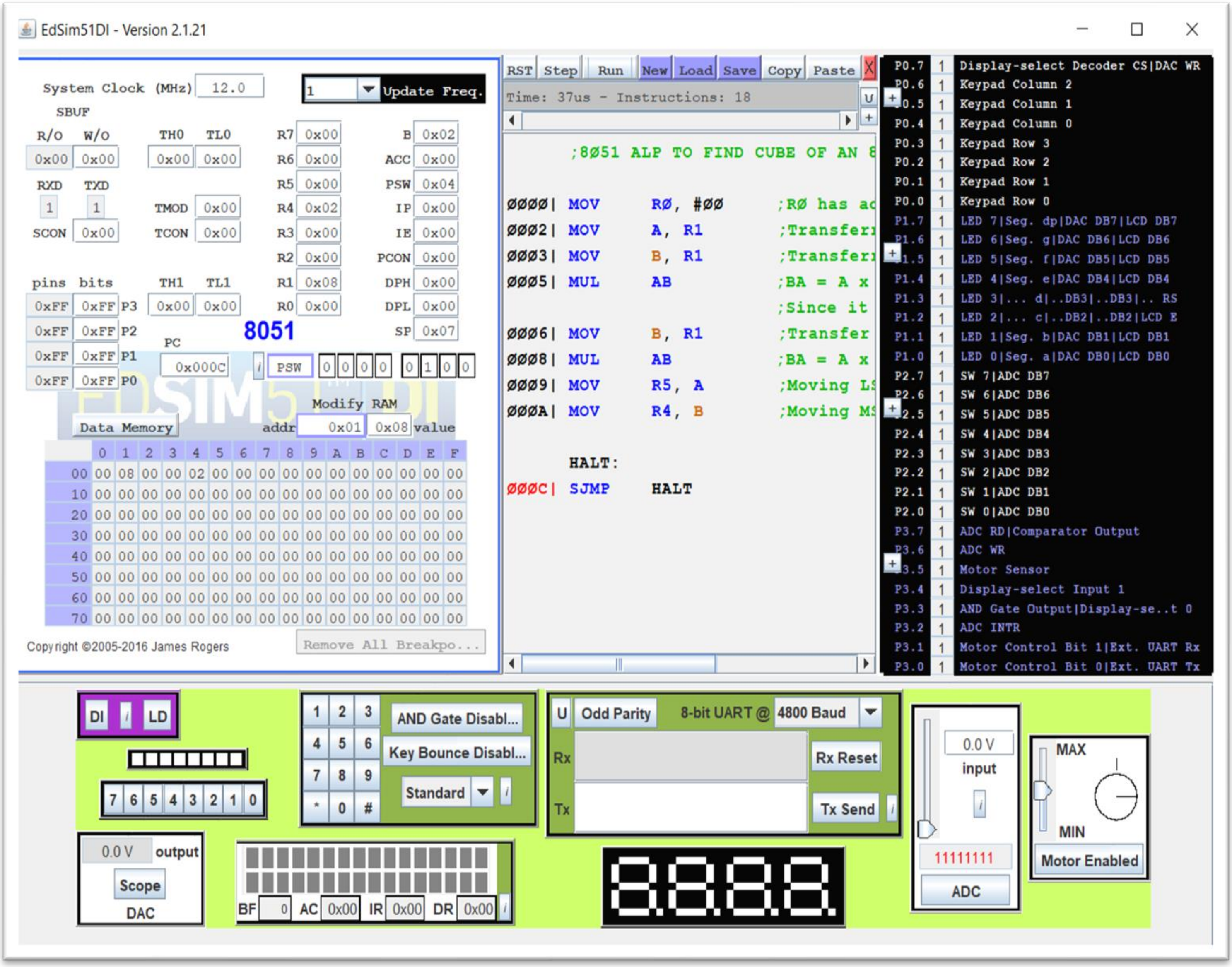
PROGRAM – 1: CUBE OF A NUMBER

ALGORITHM:

1. Begin.
2. Initialize R0 with 00h.
3. Move the value in R1 to A.
4. Move the value in R1 to B.
5. Multiply A and B.
6. Move the value in R1 to B.
7. Multiply A and B.
8. Move B to R4 (MSB of cube) and A to R5 (LSB of cube)
9. End.

PROGRAM	COMMENTS
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
	Since it is 8-bit B = 0x00
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
HALT:	
SJMP HALT	Halt the program with a loop.

SAMPLE I/O SNAPSHOT:



EdSim51DI - Version 2.1.21

System Clock (MHz) 12.0 | Update Freq. 1

Time: 37us - Instructions: 18

```
          ;8Ø51 ALP TO FIND CUBE OF AN 8

ØØØØ| MOV     RØ, #ØØ       ;RØ has ac
ØØØ2| MOV     A, R1         ;Transferr
ØØØ3| MOV     B, R1         ;Transferr
ØØØ5| MUL     AB            ;BA = A x
                            ;Since it
ØØØ6| MOV     B, R1         ;Transfer
ØØØ8| MUL     AB            ;BA = A x
ØØØ9| MOV     R5, A         ;Moving LS
ØØØA| MOV     R4, B         ;Moving MS

      HALT:
ØØØC| SJMP    HALT
```

R7	0x00	B	0x02	SBUF R/O	0x00
R6	0x00	ACC	0x00	SBUF W/O	0x00
R5	0x00	PSW	0x04	TH0	0x00
R4	0x02	IP	0x00	TL0	0x00
R3	0x00	IE	0x00	TMOD	0x00
R2	0x00	PCON	0x00	TCON	0x00
R1	0x08	DPH	0x00	TH1	0x00
R0	0x00	DPL	0x00	TL1	0x00
PC	0x000C	SP	0x07	SCON	0x00

RXD 1 TXD 1

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

PSW 0 0 0 0 0 1 0 0

Modify RAM addr 0x01 value 0x08

Data Memory

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	00	08	00	00	02	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
```

DAC output 0.0 V | ADC input 0.0 V, 11111111 | 8-bit UART @ 4800 Baud | Motor Enabled

LCD: BF 0 AC 0x00 IR 0x00 DR 0x00

RESULT:

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

BCD TO ASCII CONVERSION USING 8051

Exp No: 14 Name: Srinithyee S K

Date: 23-10-20 Register Number: 185001166

AIM:

To write an assembly language program to convert a given BCD value to its corresponding ASCII value using an 8051 microcontroller.

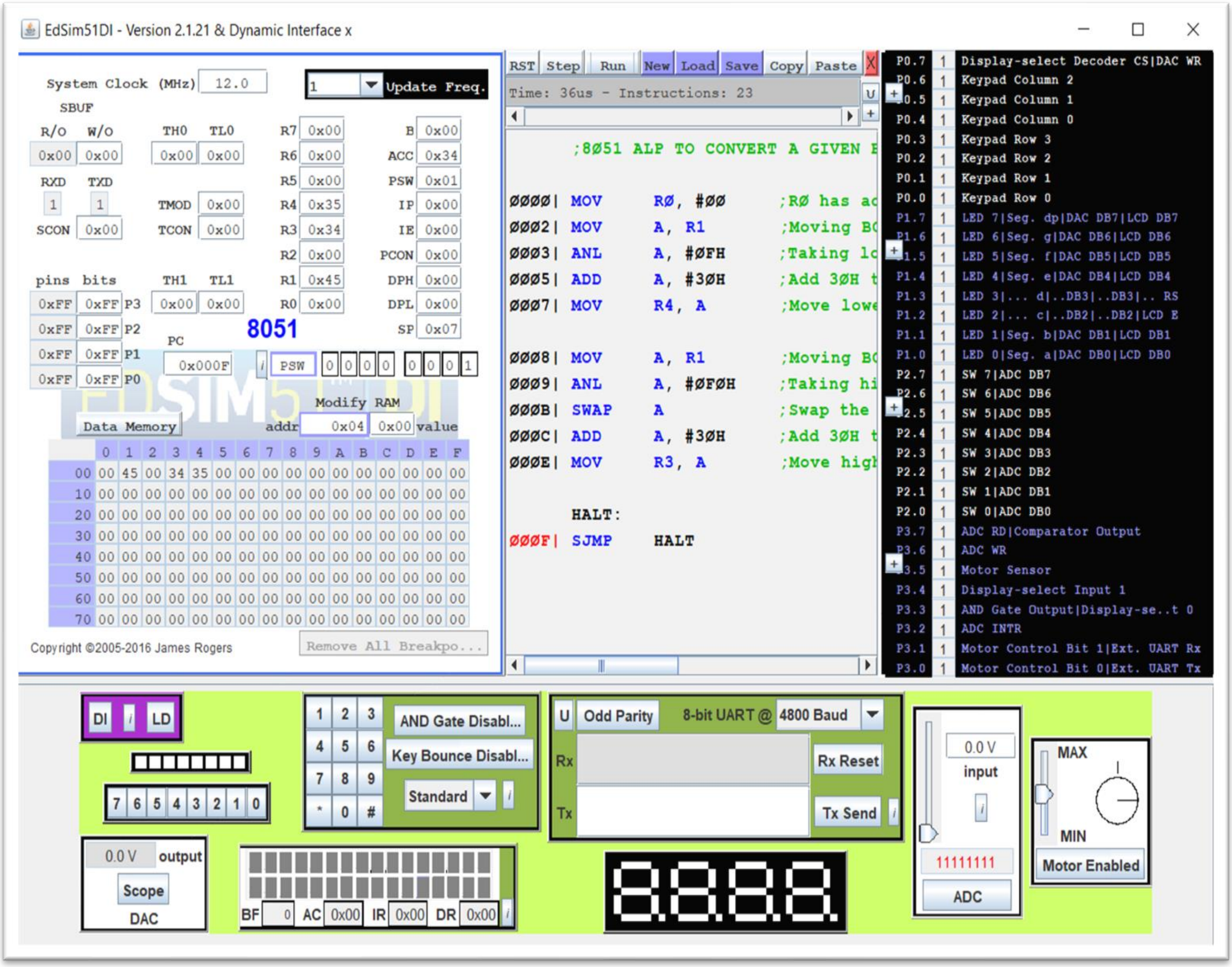
PROGRAM – 1: BCD TO ASCII CONVERSION

ALGORITHM:

1. Begin.
2. Move the value in R1 to A.
3. Get the lower byte at A by performing logical AND over A & 0F.
4. Add 30h to A.
5. Move A to R4.
6. Move the value in R1 to A.
7. Get the higher byte at A by performing logical AND over A & F0.
8. Swap the lower and higher nibble in A.
9. Add 30h to A.
10. Move A to R3.
11. End.

PROGRAM	COMMENTS
MOV R0, #00	R0 has address of 0x00
MOV A, R1	Moving BCD value to A
ANL A, #0FH	Taking lower byte value of A by doing (byte & 0F)
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 by doing (byte & F0)
SWAP A	Swap the lower and higher bytes 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
HALT:	
SJMP HALT	Halt the program with a loop.

SAMPLE I/O SNAPSHOT:



EdSim51DI - Version 2.1.21 & Dynamic Interface x

System Clock (MHz) 12.0 | Update Freq. 1

SBUF R/O	SBUF W/O	TH0	TL0	TMOD	TCON	TH1	TL1
0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00

RXD 1, TXD 1, SCON 0x00

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

Register	Value	Register	Value
R7	0x00	B	0x00
R6	0x00	ACC	0x34
R5	0x00	PSW	0x01
R4	0x35	IP	0x00
R3	0x34	IE	0x00
R2	0x00	PCON	0x00
R1	0x45	DPH	0x00
R0	0x00	DPL	0x00
		SP	0x07

8051 PC 0x000F PSW 0 0 0 0 0 0 0 1

Modify RAM addr 0x04 value 0x00

Data Memory

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	00	45	00	34	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

Time: 36us - Instructions: 23

```
        ;8Ø51 ALP TO CONVERT A GIVEN E

ØØØØ| MOV     RØ, #ØØ        ;RØ has ac
ØØØ2| MOV     A, R1          ;Moving BC
ØØØ3| ANL     A, #ØFH        ;Taking lc
ØØØ5| ADD     A, #3ØH        ;Add 3ØH t
ØØØ7| MOV     R4, A          ;Move lowe

ØØØ8| MOV     A, R1          ;Moving BC
ØØØ9| ANL     A, #ØFØH       ;Taking hi
ØØØB| SWAP    A              ;Swap the
ØØØC| ADD     A, #3ØH        ;Add 3ØH t
ØØØE| MOV     R3, A          ;Move high

      HALT:
ØØØF| 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:

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.