

Ex No:1	Assembly Language Programs
Date:	

Aim:

The purpose of this experiment is to write and implement 8051 assembly language experiments using simulator.

Software Required:

Edsim51 simulator

Program Logic:

An Assembly language program consists of, among other things, a series of line of Assemble language instructions. An Assembly language instruction consists of mnemonic, optionally followed by one or two operands. The operands are the data items being manipulated and the mnemonics are the commands to the CPU, telling it what to do with those items.

1. Looping:

Repeating a sequence of instructions a certain number of time is called a loop. The loop is one of the most widely used actions that any microprocessor performs. In this instruction, the register is decremented; if it is not zero, it jumps to the target address referred by the label. Prior to the start of the loop the register is loaded with the counter for the number of repetitions. In this instruction both the register decrement and the decision to jump are combined into a single instruction.

A loop inside another loop is called nested loop.

2. Unconditional jump instruction:

The unconditional jump is a jump in which control is transferred unconditionally to the target location. In 8051 there is two unconditional jumps:

- **LJMP(long jump):**

LJMP is an unconditional long jump. It is a 3-byte instruction in which the first byte is a opcode, second and third bytes represent the 16-bit address of the target location. The 2-byte target address allows a jump to any memory location from 0000 to FFFFH.

- **SYMP(Short jump):**

SJMP is an unconditional short jump. It is a 2-byte instruction in which the first byte is a opcode and second bye is the relative address of the target location. The target address ranges from 00 to FFH.

Procedure:

Step-1:Open the Edsim51 simulator.

Step-2:Type the program and save it with “.asm” or “.hex” extension.

Step-3:Run the program and rectify the result.

Step-4:Observe the output.

Step-5:Repeate the above steps for all the program.

Program:

1.Sample of Assembly language program:

```
ORG 0H           ;start (origin) at location 0
MOV R5, #25H      ;load 25H into R5
MOV R7,#34H       ;load 34H into R7
MOV A, #0         ;load 0 into A
ADD A, R5         ;add contents of R5 to A
                 ;now A = A + R5
ADD A, R7         ;add contents of R7 to A
                 ;now A = A + R7
ADD A, #12H       ;add to A value 12H
                 ;now A = A + 12H
HERE:SJMP HERE    ;Stay in this loop
END              ;end of asm source file
```

2.Multiply 25 by 10 using the technique of repeated addition:

```
MOV A, #0         ;A=0, clear ACC
MOV R2, #10       ;the multiplier is placed in R2
AGAIN:ADD A, #25   ;add the multiplicand to the ACC
DJNZ R2, AGAIN    ;repeat until R2=0 (10 times)
MOV R5,A          ;save A in R5 ;R5-FAH
```

3.Program to add first 10 natural number:

```
MOV A, 0          ;A0,clear Acc
MOV R2, #10       ;load counter value in R2
MOV R0, #0        ;initialize R0 to zero
AGAIN:INC R0      ;increment R0 to hold the natural numbers
ADD A,R0         ;add first number to ACC
DJNZ R2, AGAIN    ;repeat until R2=0(10 times)
MOV 46H, A        ;save the result (37H) in RAM location 46H
```

4.Program to load the accumulator with the value of 55H and complement the ACC 700 times:

```
MOV A,#55H        ;A=55H
MOV R3,#10        ;R3=10, the outer loop count
NEXT:MOV R2,#70   ;R2=70,the inner loop count
AGAIN:CPL A       ;complement A register
DJNZ R2,AGAIN     ;repeat it 70 times(inner loop)
DJNZ R3,NEXT
```

5. Multiply ECH by 25H using the technique of repeated addition:

MOV R1,#0	;R1=0,this 1s the register to store the MSB
MOV A,#0	;clear ACC
MOV R0,#25H	;the multiplier is placed in R0
AGAIN: ADD A,#0ECH	;add the multiplicand to the Acc
JNC HERE	;if no carry, then repeat the addition
INC R1	;increment R1 for each carry generated
HERE: DJNZ R0,AGAIN	;repeat until R0=0
MOV R0,A	;the LSB of the product is moved to R0
	;the MSB of the product is in R1
	; now R1=22H and R0=1CH

Output:

1. Sample of Assembly language program:

System Clock (MHz) 12.0 Update Freq.

Time: 376us - Instructions: 191

```
ORG 0H
0000 MOV R5, #25H
0002 MOV R7, #34H
0004 MOV A, #0
0006 ADD A, R5
0007 ADD A, R7
0008 ADD A, #12H
000A HERE: SJMP HERE
END
```

8051

Remove All Breakpoints

DI LD

AND Gate Disabled

Key Bounce Disabled

Standard

No Parity 8-bit UART @ 4800 Baud

Rx Reset

Tx Send

0.0V input

11111111

ADC

MAX

MIN

Motor Enabled

0.0V output

Scope

DAC

BF 0 AC 0x00 IR 0x00 DR 0x00

8888

2. Multiply 25 by 10 using the technique of repeated addition:

System Clock (MHz) 12.0 Update Freq.

Time: 189us - Instructions: 179

```
MOV A, #0
MOV R2, #10
AGAIN: ADD A, #25
DJNZ R2, AGAIN
MOV R5, A
```

8051

Remove All Breakpoints

DI LD

AND Gate Disabled

Key Bounce Disabled

Standard

No Parity 8-bit UART @ 4800 Baud

Rx Reset

Tx Send

0.0V input

11111111

ADC

MAX

MIN

Motor Enabled

0.0V output

Scope

DAC

BF 0 AC 0x00 IR 0x00 DR 0x00

8888

3.Program to add first 10 natural number:

The screenshot shows the Proteus 8.10 SP3 environment. The assembly code is as follows:

```

00000| MOV A,#0
00002| MOV R2,#10
00004| MOV R0,#0
00006| AGAIN:INC R0
00007| ADD A,R0
00008| DJNZ R2,AGAIN
0000A| MOV 46H,A
  
```

The hardware interface includes a keypad, a display showing 8051, and various control buttons like "AND Gate Disabled", "Key Bounce Disabled", "Standard", "No Parity", "8-bit UART @ 4800 Baud", "Rx Reset", "Tx Send", "0.0 V input", "ADC", "Motor Enabled", "Scope", "DAC", and "Remove All Breakpoints".

4.Program to load the accumulator with the value of 55H and complement the ACC 700 times:

The screenshot shows the Proteus 8.10 SP3 environment. The assembly code is as follows:

```

00000| MOV A,#55H
00002| MOV R3,#10
00004| NEXT:MOV R3,#700
00006| AGAIN:CPL A
00007| DJNZ R2,AGAIN
00009| DJNZ R3,NEXT
  
```

The hardware interface includes a keypad, a display showing 55H, and various control buttons like "AND Gate Disabled", "Key Bounce Disabled", "Standard", "No Parity", "8-bit UART @ 4800 Baud", "Rx Reset", "Tx Send", "0.0 V input", "ADC", "Motor Enabled", "Scope", "DAC", and "Remove All Breakpoints".

5. Multiply ECH by 25H using the technique of repeated addition:

System Clock (MHz) 12.0 50000 Update Freq.

RST Step Run New Load Save Copy Paste

Time: 650ms 740us - Instructions: 650000

8051

MOV R1, #0
MOV A, #0
MOV R0, #25H
AGAIN: ADD A, #0ECH
JNC HERE
INC R1
HERE: DJNZ R0, AGAIN
MOV R0, A

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

DI LD
AND Gate Disabled
Key Bounce Disabled
Standard
0.0 V output
Scope
DAC
BF 0 AC 0x00 IR 0x00 DR 0x00
0.0 V input
1111111
ADC
MAX
MIN
Motor Enabled

Result:

Thus the 8051 assembly level program was written and implemented successfully

Ex No:2	To Text Data Transfer Between Register and Memory
Date:	

Aim:

The purpose of this experiment is to text data transfer between register and memory.

Software Required:

Edsim51 simulator.

Program Logic:

In 8051, the action of comparing and jumping are combined into single instruction called CJNE. The CJNE instruction compares two operands and jump if they are not equal. In addition, it changes the CY flag to indicate if the destination operand is larger or smaller. It is important to notice that the operand themselves remains unchanged.

Procedure:

Step-1: Enter the opcode in Edsim51 simulator.

Step-2: Execute the program.

Step-3: Check the result in register A.

Program:

```
MOV R0,#50H    ;R0 is the pointer to the data
MOV R1,#10H    ;R1 is the counter
MOV B,#0       ;B=0
BACK:MOV A,@R0  ;move a number to A
CJNE A,B,LOOP  ;compare with B
LOOP:JC LOOP1   ;if A<B, jump to loop1
MOV B,A        ;if A>B, move it to B ie., the biggest number should be in B
INC R0         ;increment the pointer
DJNZ R1,BACK    ;repeat until the counter=0
SJMP NEXT      ;jump to EXIT, the biggest number is in B
LOOP1:INC R0    ;this is another loop, taken when the biggest number was already in buffer
              ;as comparison
DJNZ R1,BACK    ;repeat until the counter=0
NEXT:MOV A,B    ;transfer the biggest number to A register
MOV 60H,A      ;transfer the result to ROM location 6011
END            ;end of asm source file
```

Output:

System Clock (MHz) 12.0 Update Freq. 1

R/W TH0 TL0 R7 0x00 B 0x00
0x00 0x00 0x00 0x00 R6 0x00 ACC 0x00
R/D TXD TCON 0x00 R5 0x00 PSW 0x00
1 1 TMOD 0x00 R4 0x00 IP 0x00
SOCON 0x00 TCON 0x00 R3 0x10 IE 0x00
pins bits TH1 TL1 R2 0x50 PCON 0x00
0xFF 0xFF P3 0x00 0x00 R1 0x00 DPH 0x00
0xFF 0xFF P2 0x00 0x00 R0 0x60 DPL 0x00
0xFF 0xFF P1 0x00 0x00 SP 0x07
0xFF 0xFF P0 0x00 0x00

8051

Data Memory

addr	0x00	0x00	value
0	0	1	2
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
A	0	0	0
B	0	0	0
C	0	0	0
D	0	0	0
E	0	0	0
F	0	0	0

copyright ©2005-2022 James Rogers Remove All Breakpoints

RST Step Pause New Load Save Copy Paste X

Time: 530us - Instructions: 492

```
0000 MOV R2,#50H
0002 MOV R3,#10H
0004 MOV B,#0
0007 BACK:MOV A,@R0
0008 CJNE A,B,LOOP
000B LOOP:JC LOOP1
000D MOV B,A
000F INC R0
0010 DJNZ R1,BACK
0012 SJMP NEXT
0014 LOOP1:INC R0
0015 DJNZ R1,BACK
0017 NEXT:MOV A,B
0019 MOV 60H,A
END
```

P0.7 1 Display-select Decoder CS/LMC
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(Seg. d)(DAC DB3)LCD DB3
P1.2 1 LED 2(Seg. c)(DAC DB2)LCD DB2
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...
P3.2 1 ADC INTR
P3.1 1 Motor Control Bit 1(Ext. UART
P3.0 1 Motor Control Bit 0(Ext. UART

DI i LD

1 2 3 AND Gate Disabled
4 5 6 Key Bounce Disabled
7 8 9 Standard
* 0 #

U No Parity 8-bit UART @ 4800 Baud
Rx Rx Reset
Tx Tx Send

0.0 V output
Scope DAC

BF 0 AC 0x00 IR 0x00 DR 0x00

0.0 V input
11111111
ADC

MAX
MIN
Motor Enabled

Result:

Thus the program to text data transfer between Register and Memory was written and executed successfully.

Ex No:3	Perform ALU operation using 8051 microcontroller
Date:	

Aim:

The purpose of this experiment is to add, subtract, multiply and divide the given two 8 bit numbers and store them in a memory location.

Software Requirement:

Edsim 51 Simulator.

Program Logic:

To perform addition in 8051 one of the data should be in accumulator, another data can be in any of the general purpose register or in memory or immediate data. After addition the sum will be in accumulator. The sum of two 8-bit data can be either 8-bits(sum only) or 9-bits(sum and carry). The accumulator can accumulate only the sum and there is a carry the 8051 will indicate by setting carry flag. Hence one of the register is used to account for carry.

The 8051 has MUL instruction unlike many other 8-bit processors. MUL instruction multiplies the unsigned 8-bit integers in A and B. The lower order byte of the product is left in A and the higher order byte in B.

The 8051 has DIV instruction unlike many other 8-bit processors. DIV instruction divides the unsigned 8-bit integers in A and B. The accumulator receives the integer part of the quotient and the register B receives the remainder.

Procedure:

Step-1: Enter the opcodes from memory location 4200.

Step-2:Execute the program.

Step-3:Check for the result at 4100 and 4101.Using the accumulator, subtraction is performed and the result is stored. Immediate addressing is employed. The SUBB instruction drives the result in the accumulator.

Program:

1.ADDITION:

CLR C	;make CY=0
MOV A,#45H	;load the low byte into A
ADD A,#0ECH	;add the low byte ,now A=31,CY=1
MOV R0,A	;Save the low byte of sum in R0
MOV A,#02H	;load the high byte into A
ADD A,#0FCH	;add the high bytes with carry
MOV R1,A	;save the high byte of result n R1

2.SUBTRACTION:

i) Subtraction with CY=0:

CLR C	;make CY=0
MOV A,#3FH	;load 3FH into A(A=3FH)
MOV R3,#23H	;load 23H into R3(R3=23H)
SUBB A,R3	;subtract A-R3,place the result in A

ii) Subtraction with CY=1:

MOV A,62H	;A=62H
SUBB A,#96H	;62H-96H=CCH with CY=1
MOV R7,A	;save the result
MOV A,#27H	;A=27H
SUBB A,#12H	;27H-12H-1=14H
MOV R6,A	;save the result

3)MULTIPLICATION:

MOV A,#25H	;load 25H to reg A
MOV B,65H	;load 65H to reg B
MUL AB	;25H*65H=E99 where ;B=0EH and A=99H

4)DIVISION:

MOV A,#95H	;load 95 into A
MOV B,#0AH	;move the divisor into B
DIV AB	;divide the hex number by 10
MOV R0,B	;the remainder is in B, move it to R0
MOV B,#0AH	;reload the divisor into B
DIV AB	;divide the quotient by 10
MOV R1,B	;move the remainder to R1
MOV R2,A	;move the last quotient to R2

Output:

1.ADDITION:

System Clock (MHz): 12.0
SBUF
R/O W/O TH0 TL0 R7 0x00 B 0x00
0x00 0x00 0x00 0x00 R6 0x00 ACC 0xFE
R5 0x00 PSW 0x01
R4 0x00 IP 0x00
R3 0x00 IE 0x00
R2 0x00 PCON 0x00
R1 0xFE DPH 0x00
R0 0x31 DPL 0x00
SP 0x07
PC 0x00C0 PSW 0 0 0 0 0 0 0 1
Data Memory
addr 0x00 value
0 0 1 2 3 4 5 6 7 8 9 A B C D E F
00 31 FE 00 00 00 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
20 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
40 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
60 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
Copyright ©2005-2022 James Rogers
Remove All Breakpoints
RST Step Run New Load Save Copy Paste
Time: 188us - Instructions: 188
0000 CLR C
0001 MOV A, #45H
0003 ADD A, #0E0H
0005 MOV R0, A
0006 MOV A, #02H
0008 ADD A, #0F0H
000A MOV R1, A
P0.7 Display-select Decoder CS/DAC WR
P0.6 Keypad Column 2
P0.5 Keypad Column 1
P0.4 Keypad Column 0
P0.3 Keypad Row 3
P0.2 Keypad Row 2
P0.1 Keypad Row 1
P0.0 Keypad Row 0
P1.7 LED 7/Seg. dp/DAC DB7/LCD DB7
P1.6 LED 6/Seg. g/DAC DB6/LCD DB6
P1.5 LED 5/Seg. f/DAC DB5/LCD DB5
P1.4 LED 4/Seg. e/DAC DB4/LCD DB4
P1.3 LED 3/... d/...DB3/...DB3/...RS
P1.2 LED 2/... c/...DB2/...DB2/LCD E
P1.1 LED 1/Seg. b/DAC DB1/LCD DB1
P1.0 LED 0/Seg. a/DAC DB0/LCD DB0
P2.7 SW 7/ADC DB7
P2.6 SW 6/ADC DB6
P2.5 SW 5/ADC DB5
P2.4 SW 4/ADC DB4
P2.3 SW 3/ADC DB3
P2.2 SW 2/ADC DB2
P2.1 SW 1/ADC DB1
P2.0 SW 0/ADC DB0
P3.7 ADC RD/Comparator Output
P3.6 ADC WR
P3.5 Motor Sensor
P3.4 Display-select Input 1
P3.3 AND Gate Output/Display-se..t 0
P3.2 ADC INTR
P3.1 Motor Control Bit 1/Ext. UART Rx
P3.0 Motor Control Bit 0/Ext. UART Tx
DI LD
7 6 5 4 3 2 1 0
0.0V output
Scope
DAC
BF 0 AC 0x00 IR 0x00 DR 0x00
1 2 3 AND Gate Disabled
4 5 6 Key Bounce Disabled
7 8 9 Standard
* 0 #
U No Parity 8-bit UART @ 4800 Baud
Rx Rx Reset
Tx Tx Send
0.0V input
11111111
ADC
MAX
MIN
Motor Enabled
8.8.8.8

2.SUBTRACTION:

i) Subtraction with CY=0:

System Clock (MHz): 12.0
SBUF
R/O W/O TH0 TL0 R7 0x00 B 0x00
0x00 0x00 0x00 0x00 R6 0x00 ACC 0x1C
R5 0x00 PSW 0x01
R4 0x00 IP 0x00
R3 0x23 IE 0x00
R2 0x00 PCON 0x00
R1 0xFE DPH 0x00
R0 0x31 DPL 0x00
SP 0x07
PC 0x0126 PSW 0 0 0 0 0 0 0 1
Data Memory
addr 0x00 value
0 0 1 2 3 4 5 6 7 8 9 A B C D E F
00 31 FE 00 23 00 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
20 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
40 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
60 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
Copyright ©2005-2022 James Rogers
Remove All Breakpoints
RST Step Run New Load Save Copy Paste
Time: 292us - Instructions: 292
0000 MOV A, #3FH
0002 MOV R3, #23H
0004 SUBB A, R3
P0.7 Display-select Decoder CS/DAC WR
P0.6 Keypad Column 2
P0.5 Keypad Column 1
P0.4 Keypad Column 0
P0.3 Keypad Row 3
P0.2 Keypad Row 2
P0.1 Keypad Row 1
P0.0 Keypad Row 0
P1.7 LED 7/Seg. dp/DAC DB7/LCD DB7
P1.6 LED 6/Seg. g/DAC DB6/LCD DB6
P1.5 LED 5/Seg. f/DAC DB5/LCD DB5
P1.4 LED 4/Seg. e/DAC DB4/LCD DB4
P1.3 LED 3/... d/...DB3/...DB3/...RS
P1.2 LED 2/... c/...DB2/...DB2/LCD E
P1.1 LED 1/Seg. b/DAC DB1/LCD DB1
P1.0 LED 0/Seg. a/DAC DB0/LCD DB0
P2.7 SW 7/ADC DB7
P2.6 SW 6/ADC DB6
P2.5 SW 5/ADC DB5
P2.4 SW 4/ADC DB4
P2.3 SW 3/ADC DB3
P2.2 SW 2/ADC DB2
P2.1 SW 1/ADC DB1
P2.0 SW 0/ADC DB0
P3.7 ADC RD/Comparator Output
P3.6 ADC WR
P3.5 Motor Sensor
P3.4 Display-select Input 1
P3.3 AND Gate Output/Display-se..t 0
P3.2 ADC INTR
P3.1 Motor Control Bit 1/Ext. UART Rx
P3.0 Motor Control Bit 0/Ext. UART Tx
DI LD
7 6 5 4 3 2 1 0
0.0V output
Scope
DAC
BF 0 AC 0x00 IR 0x00 DR 0x00
1 2 3 AND Gate Disabled
4 5 6 Key Bounce Disabled
7 8 9 Standard
* 0 #
U No Parity 8-bit UART @ 4800 Baud
Rx Rx Reset
Tx Tx Send
0.0V input
11111111
ADC
MAX
MIN
Motor Enabled
8.8.8.8

ii) Subtraction with CY=1:

The screenshot displays the Proteus ISIS simulation environment. The main window shows the assembly code for a subtraction operation with the carry flag (CY) set to 1. The code is as follows:

```
0000| MOV A,#62H
0002| SUBB A,#96H
0004| MOV R7,A
0005| MOV A,#27H
0007| SUBB A,#12H
0009| MOV R6,A
```

The register window shows the following values:

Register	Value
R7	0xCC
R6	0x14
R5	0x00
R4	0x00
R3	0x23
R2	0x00
R1	0xFF
R0	0x31

The PC register is 0x00AC, and the PSW register is 0x0000. The data memory window shows the following values:

Address	Value
00	FE
01	00
02	23
03	00
04	14
05	CC
06	00
07	00
08	00
09	00
0A	00
0B	00
0C	00
0D	00
0E	00
0F	00

The hardware components at the bottom include a keyboard, a display showing '8888', an ADC input set to 0.0V, and a motor control section with 'Motor Enabled' checked.

3.MULTIPLICATION:

The screenshot displays the Proteus ISIS simulation environment. The main window shows the assembly code for a multiplication operation. The code is as follows:

```
0000| MOV A,#25H
0002| MOV B,#65H
0005| MUL AB
```

The register window shows the following values:

Register	Value
R7	0xCC
R6	0x14
R5	0x00
R4	0x00
R3	0x23
R2	0x00
R1	0xFF
R0	0x31

The PC register is 0x0141, and the PSW register is 0x0000. The data memory window shows the following values:

Address	Value
00	FE
01	00
02	23
03	00
04	14
05	CC
06	00
07	00
08	00
09	00
0A	00
0B	00
0C	00
0D	00
0E	00
0F	00

The hardware components at the bottom include a keyboard, a display showing '8888', an ADC input set to 0.0V, and a motor control section with 'Motor Enabled' checked.

4.DIVISION

The screenshot displays the Proteus 8.10 SP3 IDE interface, showing a project for an AVR microcontroller. The main window is divided into several panes:

- Register Window:** Shows the status of various registers. The PC (Program Counter) is highlighted at address 0x005A, containing the value 8051. Other registers like R7, R6, R5, R4, R3, R2, R1, R0, ACC, PSW, IP, IE, PCON, DPH, DPL, and SP are also visible.
- Memory Window:** Displays the memory map, showing addresses from 0x00 to 0x0F and their corresponding values.
- Disassembly Window:** Shows the assembly code being executed. The instructions are: `MOV A, #95H`, `MOV B, #0AH`, `DIV AB`, `MOV R0, B`, `MOV B, #0AH`, `DIV AB`, `MOV R1, B`, and `MOV R2, A`.
- Pin List Window:** Lists the pins and their functions, including P0.7 (Display-select Decoder CS/DAC WR), P0.6 (Keypad Column 2), P0.5 (Keypad Column 1), P0.4 (Keypad Column 0), P0.3 (Keypad Row 3), P0.2 (Keypad Row 2), P0.1 (Keypad Row 1), P0.0 (Keypad Row 0), P1.7 (LED 7/Seg. dp/DAC DB7/LCD DB7), P1.6 (LED 6/Seg. g/DAC DB6/LCD DB6), P1.5 (LED 5/Seg. f/DAC DB5/LCD DB5), P1.4 (LED 4/Seg. e/DAC DB4/LCD DB4), P1.3 (LED 3/... d/...DB3/...DB3... RS), P1.2 (LED 2/... c/...DB2/...DB2/LCD E), P1.1 (LED 1/Seg. b/DAC DB1/LCD DB1), P1.0 (LED 0/Seg. a/DAC DB0/LCD DB0), P2.7 (SW 7/ADC DB7), P2.6 (SW 6/ADC DB6), P2.5 (SW 5/ADC DB5), P2.4 (SW 4/ADC DB4), P2.3 (SW 3/ADC DB3), P2.2 (SW 2/ADC DB2), P2.1 (SW 1/ADC DB1), P2.0 (SW 0/ADC DB0), P3.7 (ADC RD/Comparator Output), P3.6 (ADC WR), P3.5 (Motor Sensor), P3.4 (Display-select Input 1), P3.3 (AND Gate Output/Display-se..t 0), P3.2 (ADC INTR), P3.1 (Motor Control Bit 1/Ext. UART Rx), and P3.0 (Motor Control Bit 0/Ext. UART Tx).
- Hardware Window:** Shows the physical components of the circuit, including a DAC, a scope, a motor, and a keypad. The DAC is set to 0.0V output, the scope is set to 0.0V input, the motor is set to 11111111, and the keypad is set to 0.0V input.

Result:

Thus the ALU operation using 8051 microcontroller was perform successfully.