EMBEDDED SYSTEMS AND IOT LAB

EXP 1 WRITE 8051 ASSEMBLY LANGUAGE EXPERIMENTS USING SIMULATOR.

1.A PROGRAM FOR BLINKING AN LED USING EdSim51

AIM: To write an assembly level language program for Blinking an LED light using EdSim51

ALGORITHM

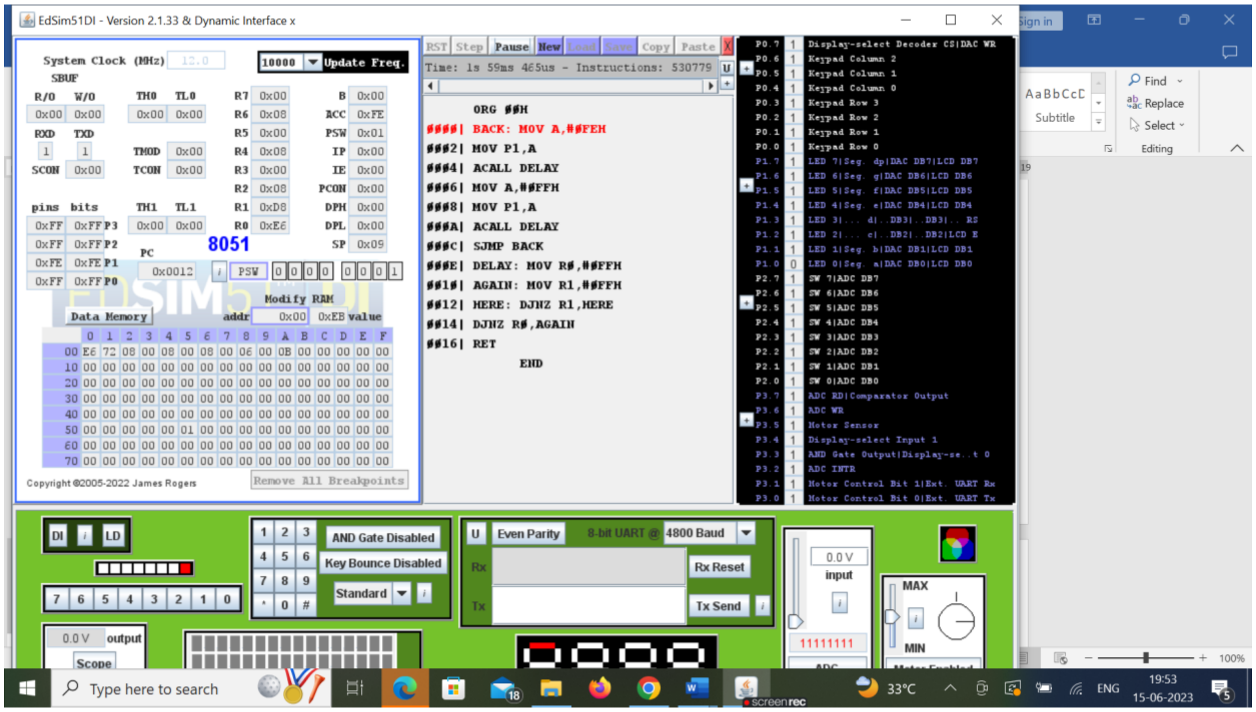
1. Start at address 0x0000 (ORG 0x0000).
2. Set up a continuous loop labeled as "BACK."
3. Move the value 0xFE (254 in decimal) to the accumulator register (A).
4. Move the value in the accumulator (A) to Port 1 (P1) to turn on the LED.
5. Call the delay subroutine by using the instruction "ACALL DELAY."
6. Move the value 0xFF (255 in decimal) to the accumulator register (A).
7. Move the value in the accumulator (A) to Port 1 (P1) to turn off the LED.
8. Call the delay subroutine by using the instruction "ACALL DELAY."
9. Jump back to the "BACK" loop using the instruction "SJMP BACK."
10. Define the delay subroutine labeled as "DELAY."
11. Move the value 0xFF (255 in decimal) to register R0.
12. Use a nested loop to create a delay:
    a. Move the value 0xFF (255 in decimal) to register R1.
    b. Create a loop labeled as "HERE" and decrement R1 using the instruction "DJNZ R1, HERE."
    c. Check if R1 is zero. If not zero, repeat the loop.
    d. Decrement R0 using the instruction "DJNZ R0, AGAIN."
    e. Check if R0 is zero. If not zero, repeat the nested loop.
    Return from the subroutine using the instruction "RET."

CODE :

```
ORG 00H
BACK: MOV A,#0FEH
MOV P1,A
ACALL DELAY
MOV A,#0FFH
MOV P1,A
ACALL DELAY
SJMP BACK
DELAY: MOV R0,#0FFH
AGAIN: MOV R1,#0FFH
HERE: DJNZ R1,HERE
DJNZ R0,AGAIN
RET
```

END

OUTPUT:



```
Time: 1s 59ms 465us - Instructions: 530779

        ORG ØØH
ØØØØ| BACK: MOV A,#ØFEH
ØØØ2| MOV P1,A
ØØØ4| ACALL DELAY
ØØØ6| MOV A,#ØFFH
ØØØ8| MOV P1,A
ØØØA| ACALL DELAY
ØØØC| SJMP BACK
ØØØE| DELAY: MOV RØ,#ØFFH
ØØ1Ø| AGAIN: MOV R1,#ØFFH
ØØ12| HERE: DJNZ R1,HERE
ØØ14| DJNZ RØ,AGAIN
ØØ16| RET
              END
```

RESULT:

1.b LOGICAL AND USING EdSim51

AIM:

To write an assembly level language program Logical AND using EdSim51

ALGORITHM:

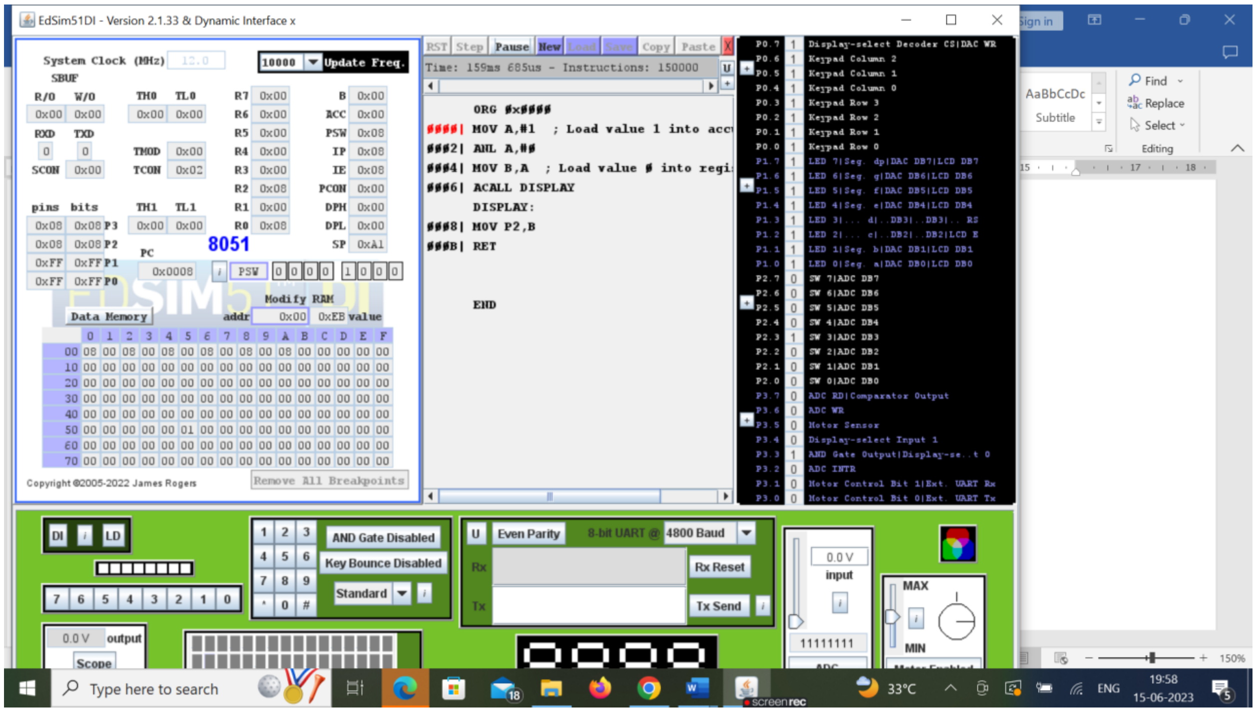
1. Start at address 0x0000 (ORG 0x0000).
2. Move the value 1 into the accumulator register A using the instruction "MOV A, #1".
3. Perform a logical AND operation between the accumulator A and the value 0 using the instruction "ANL A, #0". This operation will result in 0, as any value ANDed with 0 gives 0.
4. Move the value in the accumulator A to register B using the instruction "MOV B, A". This stores the result of the logical AND operation in register B.
5. Call the DISPLAY subroutine using the instruction "ACALL DISPLAY".
6. Define the DISPLAY subroutine: a. Move the value in register B to Port 2 (P2) using the instruction "MOV P2, B". This will display the value stored in register B on Port 2. b. Return from the subroutine using the instruction "RET".

7. End the program using the "END" directive.

CODE:

```
ORG 0x0000
MOV A,#1  ; Load value 1 into accumulator A
ANL A,#0
MOV B,A  ; Load value 0 into register B
ACALL DISPLAY
DISPLAY:
MOV P2,B
RET
END
```

OUTPUT:



EdSim51DI - Version 2.1.33 & Dynamic Interface x

```
Time: 159ms 685us - Instructions: 150000

      ORG ØxØØØØ
ØØØØ| MOV A,#1  ; Load value 1 into acc
ØØØ2| ANL A,#Ø
ØØØ4| MOV B,A  ; Load value Ø into regi
ØØØ6| ACALL DISPLAY
      DISPLAY:
ØØØ8| MOV P2,B
ØØØB| RET


      END
```

```
System Clock (MHz) 12.0     10000 Update Freq.
SBUF
R/O   W/O     TH0   TL0     R7 0x00    B    0x00
0x00  0x00    0x00  0x00    R6 0x00    ACC  0x00
RXD   TXD                   R5 0x00    PSW  0x08
0     0       TMOD  0x00    R4 0x00    IP   0x08
SCON  0x00    TCON  0x02    R3 0x00    IE   0x08
                            R2 0x08    PCON 0x00
pins  bits    TH1   TL1     R1 0x00    DPH  0x00
0x08  0x08 P3 0x00  0x00    R0 0x08    DPL  0x00
0x08  0x08 P2          8051            SP   0xA1
0xFF  0xFF P1   PC
0xFF  0xFF P0   0x0008   PSW 0 0 0 0  1 0 0 0
                              Modify RAM
Data Memory             addr 0x00   0xEB value
```

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	08	00	08	00	08	00	08	00	08	00	08	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	01	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

```
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 0 SW 7|ADC DB7
P2.6 0 SW 6|ADC DB6
P2.5 0 SW 5|ADC DB5
P2.4 0 SW 4|ADC DB4
P2.3 1 SW 3|ADC DB3
P2.2 0 SW 2|ADC DB2
P2.1 0 SW 1|ADC DB1
P2.0 0 SW 0|ADC DB0
P3.7 0 ADC RD|Comparator Output
P3.6 0 ADC WR
P3.5 0 Motor Sensor
P3.4 0 Display-select Input 1
P3.3 1 AND Gate Output|Display-se..t 0
P3.2 0 ADC INTR
P3.1 0 Motor Control Bit 1|Ext. UART Rx
P3.0 0 Motor Control Bit 0|Ext. UART Tx
```

RESULT:

EXP 2A TRANSFER N BYTES OF DATA FROM ONE LOCATION TO ANOTHER

AIM:

To write an assembly level language program to transfer N bytes of data from location A to location B .

ALGORITHM:

1. Move the value 0x30 (48 in decimal) into register R0 using the instruction
2. Move the value 0x40 (64 in decimal) into register R1 using the instruction
3. Move the value 0x05 (5 in decimal) into register R7 using the instruction
4. Start a loop labeled as "BACK":
5. a. Move the value at the memory location pointed to by R0 into the accumulator A using the instruction

   b. Move the value in the accumulator A to the memory location pointed to by R1 using the instruction

   c. Increment the value in register R0 using the instruction

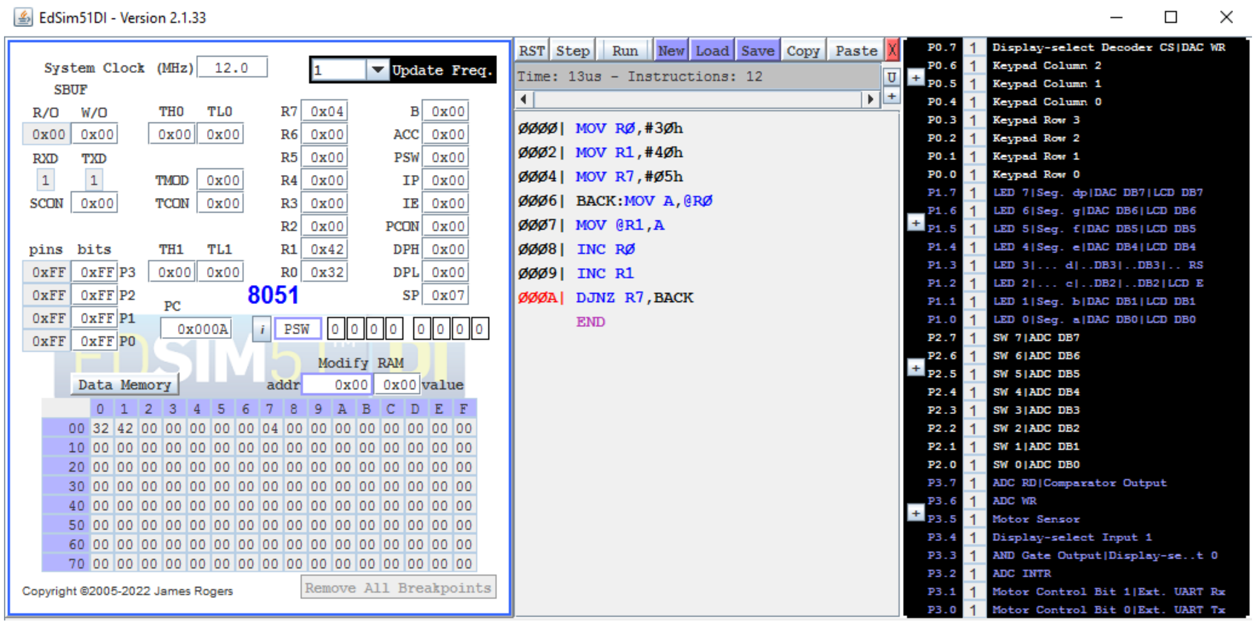
   d. Increment the value in register R1 using the instruction

   e. Decrement the value in register R7 and check if it is not zero using the instruction
6. End the program .

CODE:

```
MOV R0,#30h
 MOV R1,#40h
MOV R7,#05h
 BACK:MOV A,@R0
MOV @R1,A
 INC R0
INC R1
 DJNZ R7,BACK
      END
```

OUTPUT



EdSim51DI - Version 2.1.33

```
System Clock (MHz) 12.0        1  Update Freq.
SBUF
R/O   W/O      TH0   TL0      R7 0x04        B 0x00
0x00  0x00     0x00  0x00     R6 0x00      ACC 0x00
RXD   TXD                     R5 0x00      PSW 0x00
 1     1       TMOD  0x00     R4 0x00       IP 0x00
SCON  0x00     TCON  0x00     R3 0x00       IE 0x00
                              R2 0x00     PCON 0x00
pins  bits     TH1   TL1      R1 0x42      DPH 0x00
0xFF  0xFF P3  0x00  0x00     R0 0x32      DPL 0x00
0xFF  0xFF P2              8051             SP 0x07
0xFF  0xFF P1  PC
0xFF  0xFF P0  0x000A    i  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	32	42	00	00	00	00	00	04	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 ©2005-2022 James Rogers

Remove All Breakpoints

```
Time: 13us - Instructions: 12

0000| MOV R0,#30h
0002| MOV R1,#40h
0004| MOV R7,#05h
0006| BACK:MOV A,@R0
0007| MOV @R1,A
0008| INC R0
0009| INC R1
000A| DJNZ R7,BACK
      END
```

```
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
```

RESULT:

EXP 2B EXCHANGING N BYTES OF DATA FROM ONE LOCATION TO ANOTHER

AIM:

To write an assembly language program to exchange N=___h bytes of data at location A:____h and at location B:______h.

ALGORITHM:

Step 1: Initialize the variables

Step 2: Set up pointers

Step 3: Start the change loop

Step 4: Change a byte

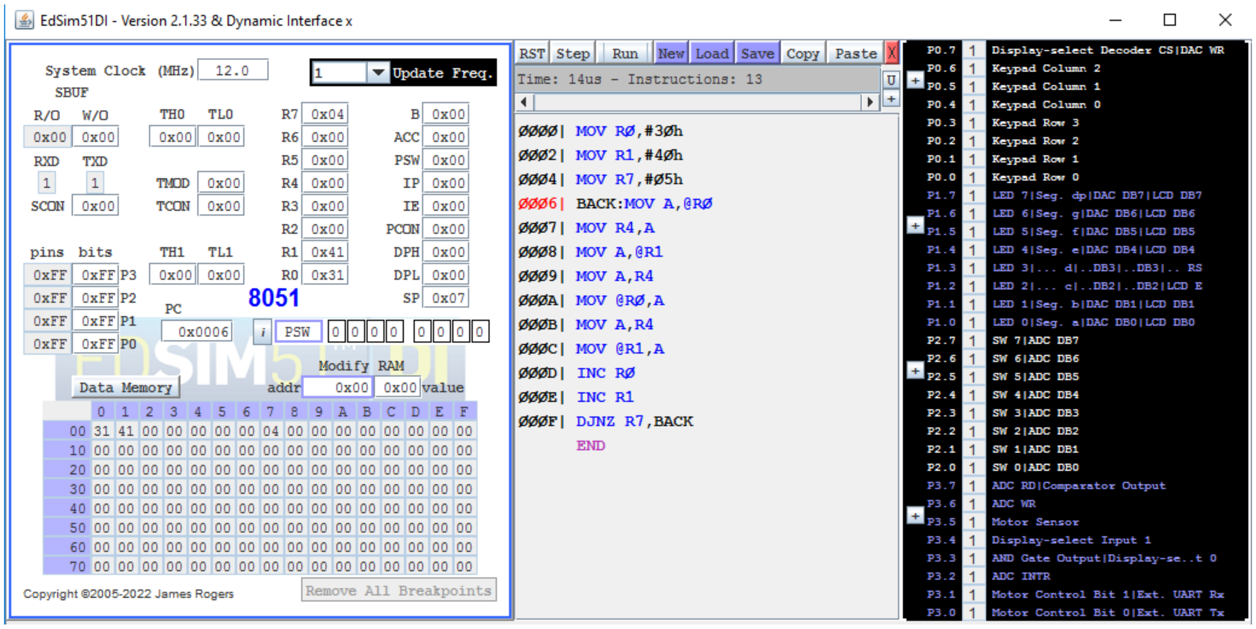
Step 5: Update pointers and counter

Step 6: Check if the change is complete

CODE:

```
MOV R0,#30h
MOV R1,#40h
MOV R7,#05h
BACK: MOV A,@R0
MOV R4,A
MOV A,@R1
MOV A,R4
MOV @R0,A
MOV A,R4
MOV @R1,A
INC R0
INC R1
DJNZ R7,BACK
END
```

OUTPUT:



EdSim51DI - Version 2.1.33 & Dynamic Interface x

```
System Clock (MHz) 12.0        1  Update Freq.
SBUF
R/O   W/O      TH0   TL0      R7 0x04      B 0x00
0x00  0x00     0x00  0x00     R6 0x00    ACC 0x00
RXD   TXD                     R5 0x00    PSW 0x00
1     1        TMOD  0x00     R4 0x00     IP 0x00
SCON  0x00     TCON  0x00     R3 0x00     IE 0x00
                              R2 0x00   PCON 0x00
pins  bits     TH1   TL1      R1 0x41    DPH 0x00
0xFF  0xFF P3  0x00  0x00     R0 0x31    DPL 0x00
0xFF  0xFF P2                 8051        SP 0x07
0xFF  0xFF P1   PC
0xFF  0xFF P0   0x0006      i 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 31 41 00 00 00 00 00 04 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 ©2005-2022 James Rogers          Remove All Breakpoints
```

```
RST Step Run New Load Save Copy Paste X
Time: 14us - Instructions: 13

ØØØØ| MOV RØ,#3Øh
ØØØ2| MOV R1,#4Øh
ØØØ4| MOV R7,#Ø5h
ØØØ6| BACK:MOV A,@RØ
ØØØ7| MOV R4,A
ØØØ8| MOV A,@R1
ØØØ9| MOV A,R4
ØØØA| MOV @RØ,A
ØØØB| MOV A,R4
ØØØC| MOV @R1,A
ØØØD| INC RØ
ØØØE| INC R1
ØØØF| DJNZ R7,BACK
      END
```

```
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
```

RESULT:

EXP 2C FIND LARGEST ELEMENT IN THE GIVEN ARRAY

AIM:

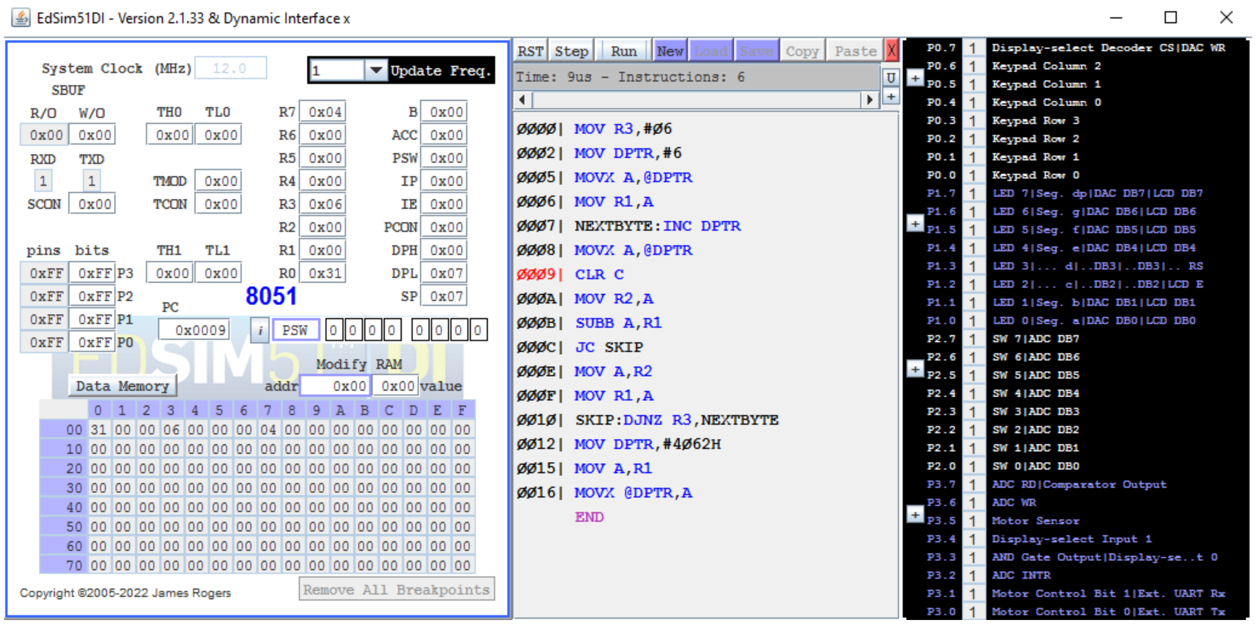
To write an assembly level language program to find the largest element in the given array of N=___h bytes at location 4000h

1. **ALGORITHM:**
   Move the value 0x06 (6 in decimal) into register R3 using the instruction
2. Move the value 0x06 into the Data Pointer (DPTR) using the instruction
3. Move the value at the external memory location pointed to by DPTR into the accumulator A using the instruction
4. Move the value in the accumulator A to register R1 using the instruction
5. Start a loop labeled as "NEXTBYTE": a. Increment the value in the Data Pointer (DPTR) using the instruction "INC DPTR". b. Move the value at the external memory location pointed to by DPTR into the accumulator A using the instruction "MOVX A, @DPTR". c. Clear the carry flag using the instruction "CLR C". d. Move the value in the accumulator A to register R2 using the instruction "MOV R2, A". e. Subtract the value in register R1 from the accumulator A using the instruction "SUBB A, R1". f. Jump to the label "SKIP" if the carry flag is set, indicating a borrow, using the instruction "JC SKIP". g. Move the value in the accumulator A to register R1 using the instruction "MOV A, R2". h. Move the value in the accumulator A to register R1 using the instruction "MOV R1, A". i. Decrement the value in register R3 and check if it is not zero using the instruction
6. Move the value 0x4062 into the Data Pointer (DPTR) using the instruction
7. Move the value in register R1 to the accumulator A using the instruction
8. Move the value in the accumulator A to the external memory location pointed to by DPTR using the instruction ".
9. End the program.

CODE:

```
MOV R3,#06

MOV DPTR,#6

MOVX A,@DPTR

MOV R1,A

NEXTBYTE:INC DPTR

MOVX A,@DPTR

CLR C

MOV R2,A

SUBB A,R1

JC SKIP

MOV A,R2

MOV R1,A

SKIP:DJNZ R3,NEXTBYTE

MOV DPTR,#4062H

MOV A,R1

MOVX @DPTR,A

    END
```

OUTPUT:



EdSim51DI - Version 2.1.33 & Dynamic Interface x

```
System Clock (MHz) 12.0        1  Update Freq.
SBUF
R/O   W/O      TH0   TL0      R7 0x04     B    0x00
0x00  0x00     0x00  0x00     R6 0x00     ACC  0x00
RXD   TXD                     R5 0x00     PSW  0x00
1     1        TMOD  0x00     R4 0x00     IP   0x00
SCON  0x00     TCON  0x00     R3 0x06     IE   0x00
                              R2 0x00     PCON 0x00
pins  bits     TH1   TL1      R1 0x00     DPH  0x00
0xFF  0xFF P3  0x00  0x00     R0 0x31     DPL  0x07
0xFF  0xFF P2               8051          SP   0x07
0xFF  0xFF P1  PC
0xFF  0xFF P0  0x0009    i  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	31	00	00	06	00	00	00	04	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 ©2005-2022 James Rogers

```
Time: 9us - Instructions: 6

ØØØØ| MOV R3,#Ø6
ØØØ2| MOV DPTR,#6
ØØØ5| MOVX A,@DPTR
ØØØ6| MOV R1,A
ØØØ7| NEXTBYTE:INC DPTR
ØØØ8| MOVX A,@DPTR
ØØØ9| CLR C
ØØØA| MOV R2,A
ØØØB| SUBB A,R1
ØØØC| JC SKIP
ØØØE| MOV A,R2
ØØØF| MOV R1,A
ØØ1Ø| SKIP:DJNZ R3,NEXTBYTE
ØØ12| MOV DPTR,#4Ø62H
ØØ15| MOV A,R1
ØØ16| MOVX @DPTR,A
      END
```

```
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
```

RESULT:

EXP 3A.TEST DATA TRANSFER BETWEEN REGISTERS AND MEMORY USING EDSIM51

AIM:

To write a test data transfer between registers and memory using EdSim51

ALGORITHM:

1. Start at address 0x0000 (ORG 0x0000).
2. Load the value 10 into register R0 using the instruction "MOV R0, #10".
3. Initialize register R1 with the value 0 using the instruction "MOV R1, #0".
4. Move the value from register R0 to the accumulator A using the instruction "MOV A, R0".
5. Move the value from the accumulator A to the memory location pointed to by R1 using the instruction "MOV @R1, A".
6. Increment the value in register R1 to access the next memory location using the instruction "INC R1".
7. Move the value from the memory location pointed to by R1 to the accumulator A using the instruction "MOV A, @R1".
8. Move the value from the accumulator A to register R2 using the instruction "MOV R2, A".
9. End the program using the "END" directive.

CODE:

```
ORG 0x0000  ; Starting address


MOV R0, #10  ; Load value 10 into register R0

MOV R1, #0   ; Initialize register R1 with 0


MOV A, R0    ; Move the value from register R0 to accumulator A

MOV @R1, A   ; Move the value from accumulator A to the memory location pointed by
           R1
```

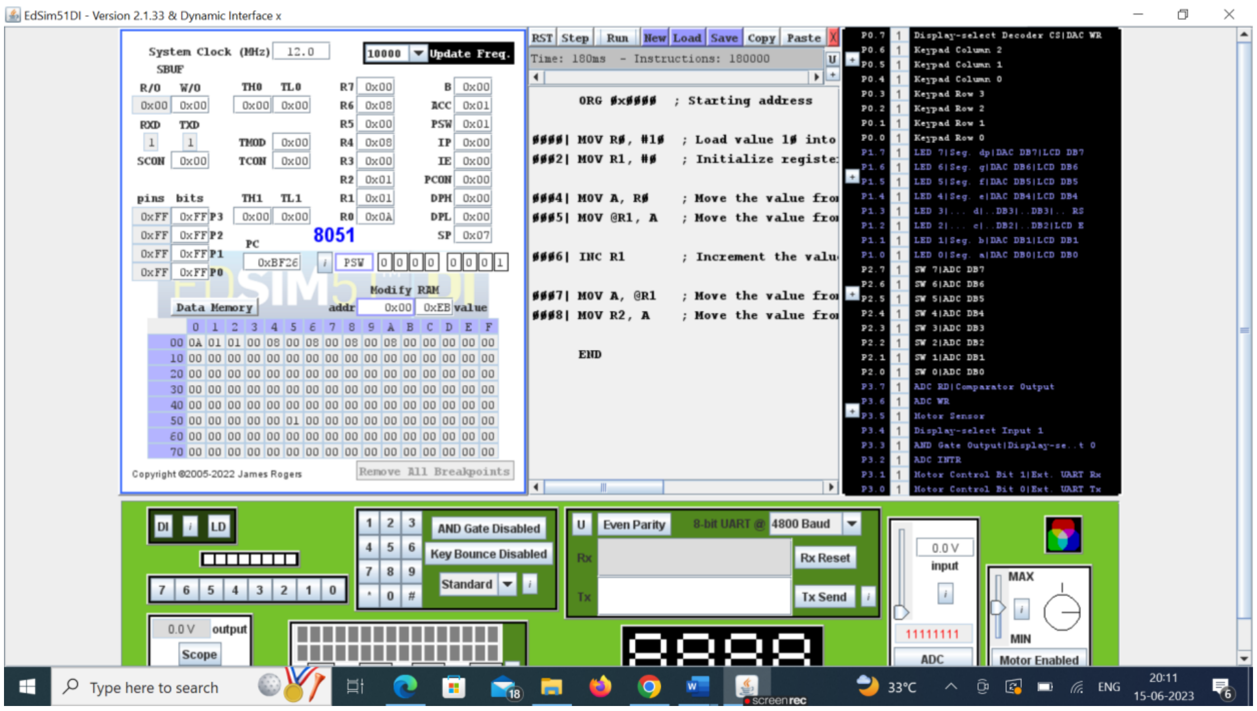
INC R1 ; Increment the value in register R1 to access the next memory location

MOV A, @R1 ; Move the value from the memory location pointed by R1 to accumulator A

MOV R2, A ; Move the value from accumulator A to register R2

END

OUTPUT:



EdSim51DI - Version 2.1.33 & Dynamic Interface x

```
Time: 180ms  - Instructions: 180000

      ORG ØxØØØØ   ; Starting address

ØØØØ| MOV RØ, #1Ø   ; Load value 1Ø into
ØØØ2| MOV R1, #Ø    ; Initialize registe

ØØØ4| MOV A, RØ     ; Move the value fro
ØØØ5| MOV @R1, A    ; Move the value fro

ØØØ6| INC R1        ; Increment the valu

ØØØ7| MOV A, @R1    ; Move the value fro
ØØØ8| MOV R2, A     ; Move the value fro

      END
```

System Clock (MHz) 12.0 — Update Freq. 10000

R7	0x00	B	0x00
R6	0x08	ACC	0x01
R5	0x00	PSW	0x01
R4	0x08	IP	0x00
R3	0x00	IE	0x00
R2	0x01	PCON	0x00
R1	0x01	DPH	0x00
R0	0x0A	DPL	0x00
		SP	0x07

SBUF R/O 0x00, W/O 0x00; TH0 0x00, TL0 0x00; TH1 0x00, TL1 0x00; TMOD 0x00; TCON 0x00; SCON 0x00; RXD 1, TXD 1; P3, P2, P1, P0 pins 0xFF, bits 0xFF; PC 0xBF26; PSW 0 0 0 0 0 0 0 1

Modify RAM addr 0x00 value 0xEB

Data Memory

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	0A	01	01	00	08	00	08	00	08	00	08	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	01	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

RESULT:

EXP.3B. PERFORM ALU OPERATIONS

AIM:

To perform ALU operations in Embedded Systems using Simulator.

ALGORITHM:

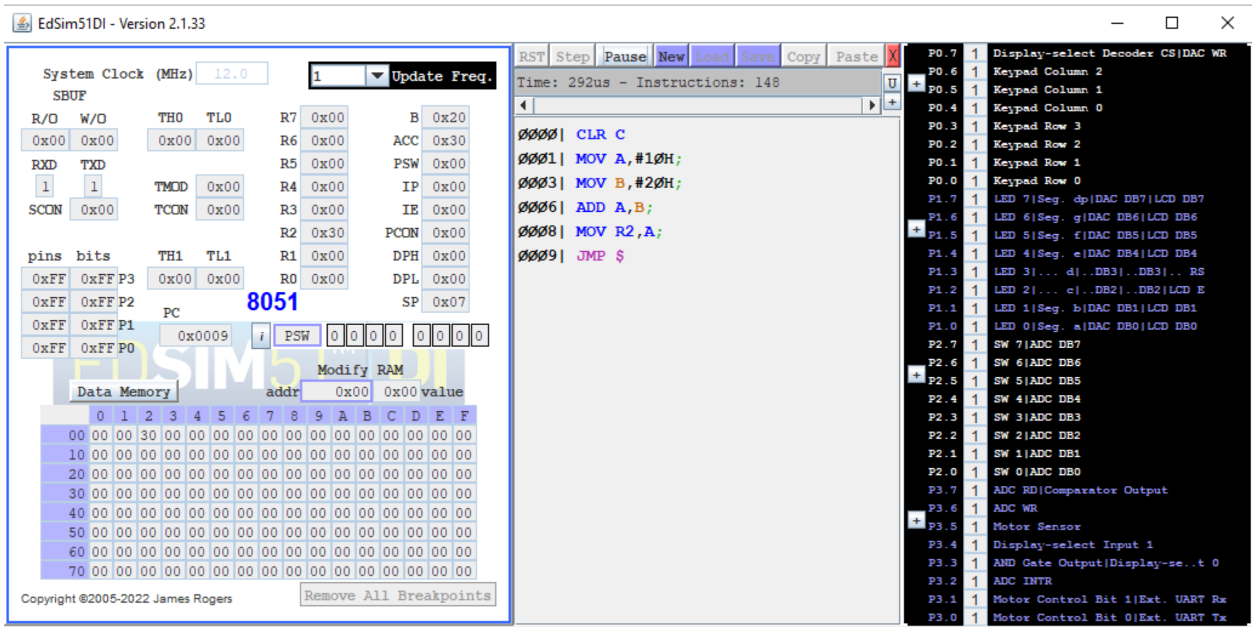
```
Step 1: Read input operands and control signal.
Step 2: Determine the type of operation based on the control signal.
Step 3: Execute the selected operation:
   - For arithmetic operations, perform the calculation.
   - For logical operations, perform the logical operation.
   - For other operations, follow the respective algorithm.
Step 4: Store the result in the ALU output register.
Step 5: Display the result on the output device.
Step 6: End the ALU operation module.
```

Code:

1. **Addition:**

```
 ORG 0x0000
MOV A, #10H
MOV B, #20H
ADD A, B
MOV R2,A

   JMP $
```

OUTPUT:



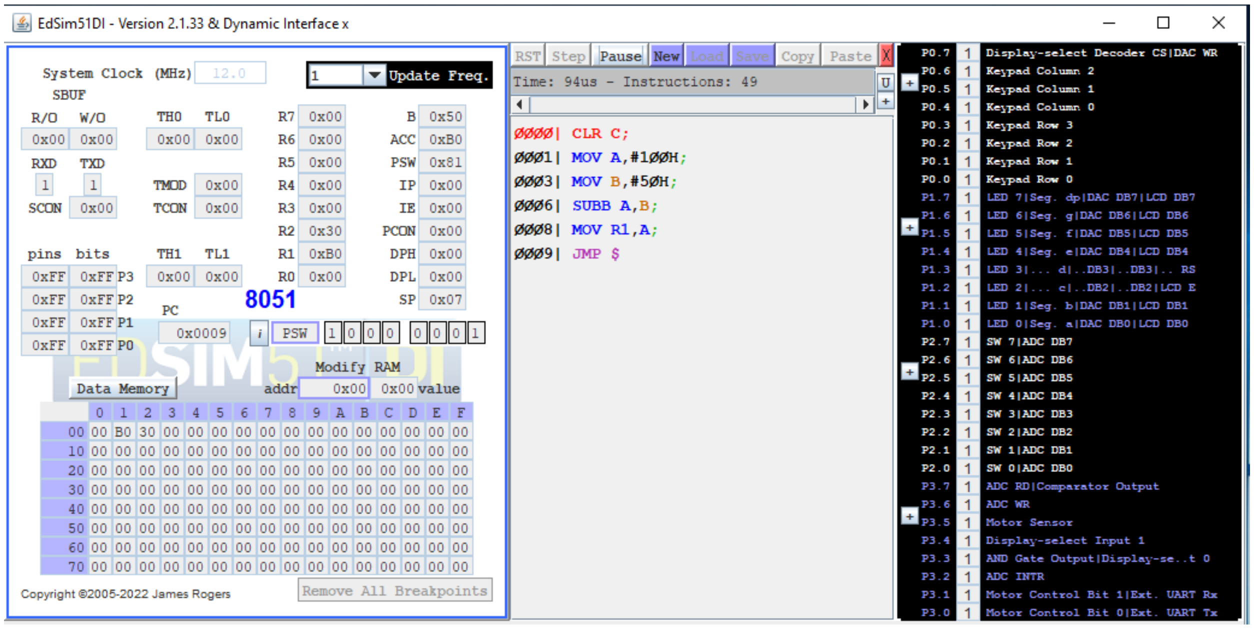
```
EdSim51DI - Version 2.1.33

System Clock (MHz) 12.0      1   Update Freq.
SBUF
R/O   W/O    TH0   TL0    R7 0x00    B    0x20
0x00  0x00   0x00  0x00   R6 0x00    ACC  0x30
RXD   TXD                 R5 0x00    PSW  0x00
1     1      TMOD  0x00   R4 0x00    IP   0x00
SCON  0x00   TCON  0x00   R3 0x00    IE   0x00
                          R2 0x30    PCON 0x00
pins  bits   TH1   TL1    R1 0x00    DPH  0x00
0xFF  0xFF P3  0x00  0x00 R0 0x00    DPL  0x00
0xFF  0xFF P2         8051           SP   0x07
0xFF  0xFF P1  PC
0xFF  0xFF P0  0x0009   i  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 00 30 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 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 ©2005-2022 James Rogers     Remove All Breakpoints

RST Step Pause New Load Save Copy Paste
Time: 292us - Instructions: 148

0000| CLR C
0001| MOV A,#10H;
0003| MOV B,#20H;
0006| ADD A,B;
0008| MOV R2,A;
0009| JMP $

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
```

Fig addition

2.SUBTRACTION

```
 ORG 0x0000
MOV A, #50H
MOV B, #30H
SUBB A, B
MOV R1,A
 JMP $
```

Output:



```
EdSim51DI - Version 2.1.33 & Dynamic Interface x

System Clock (MHz) 12.0      1   Update Freq.
SBUF
R/O   W/O     TH0   TL0    R7 0x00      B    0x50
0x00  0x00    0x00  0x00   R6 0x00      ACC  0xB0
RXD   TXD                  R5 0x00      PSW  0x81
1     1       TMOD  0x00   R4 0x00      IP   0x00
SCON  0x00    TCON  0x00   R3 0x00      IE   0x00
                           R2 0x30      PCON 0x00
pins  bits    TH1   TL1    R1 0xB0      DPH  0x00
0xFF  0xFF P3 0x00  0x00   R0 0x00      DPL  0x00
0xFF  0xFF P2              8051         SP   0x07
0xFF  0xFF P1   PC
0xFF  0xFF P0   0x0009   PSW 1 0 0 0  0 0 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 00 B0 30 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 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 ©2005-2022 James Rogers      Remove All Breakpoints

RST Step Pause New Load Save Copy Paste
Time: 94us - Instructions: 49

0000| CLR C;
0001| MOV A,#100H;
0003| MOV B,#50H;
0006| SUBB A,B;
0008| MOV R1,A;
0009| JMP $

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
```

Fig subtraction

3.Multiplication

ORG 0X000

MOV A,#06H

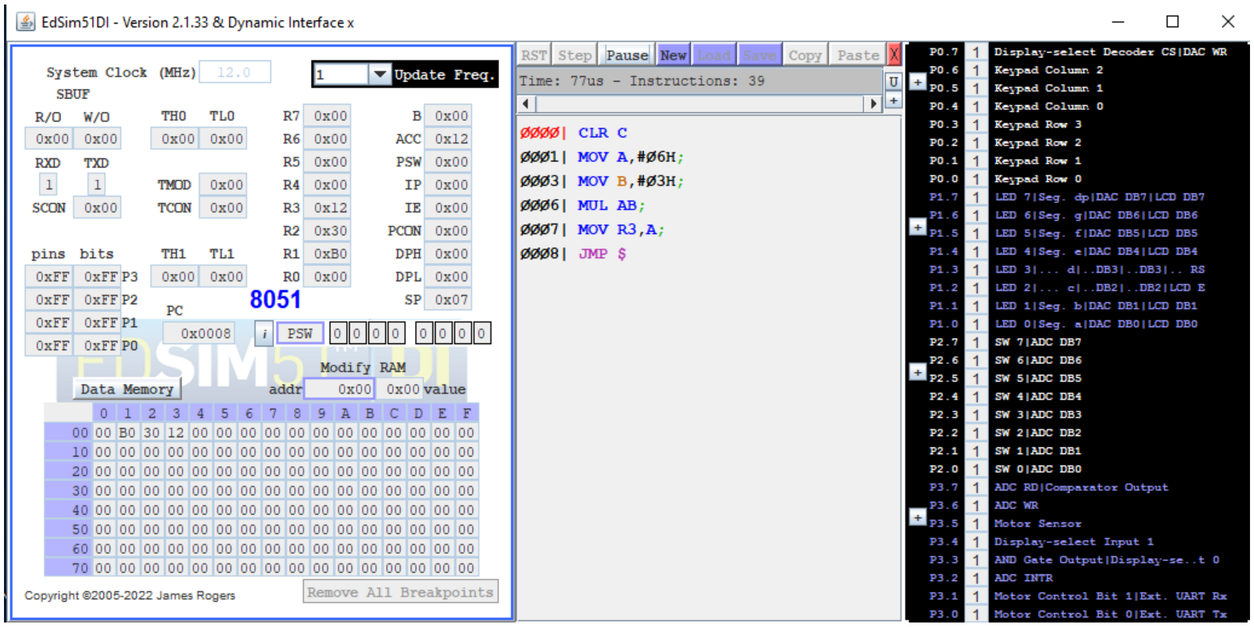
MOV B,#03H

MUL AB

MOV R3,A

JMP $

Output:



EdSim51DI - Version 2.1.33 & 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  0x12
RXD   TXD                    R5 0x00      PSW  0x00
1     1        TMOD  0x00    R4 0x00      IP   0x00
SCON  0x00     TCON  0x00    R3 0x12      IE   0x00
                             R2 0x30      PCON 0x00
pins  bits     TH1   TL1     R1 0xB0      DPH  0x00
0xFF  0xFF P3  0x00  0x00    R0 0x00      DPL  0x00
0xFF  0xFF P2            8051             SP   0x07
0xFF  0xFF P1  PC
0xFF  0xFF P0  0x0008    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	B0	30	12	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	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 ©2005-2022 James Rogers   Remove All Breakpoints

RST Step Pause New Load Save Copy Paste

```
Time: 77us - Instructions: 39

ØØØØ| CLR C
ØØØ1| MOV A,#Ø6H;
ØØØ3| MOV B,#Ø3H;
ØØØ6| MUL AB;
ØØØ7| MOV R3,A;
ØØØ8| JMP $
```

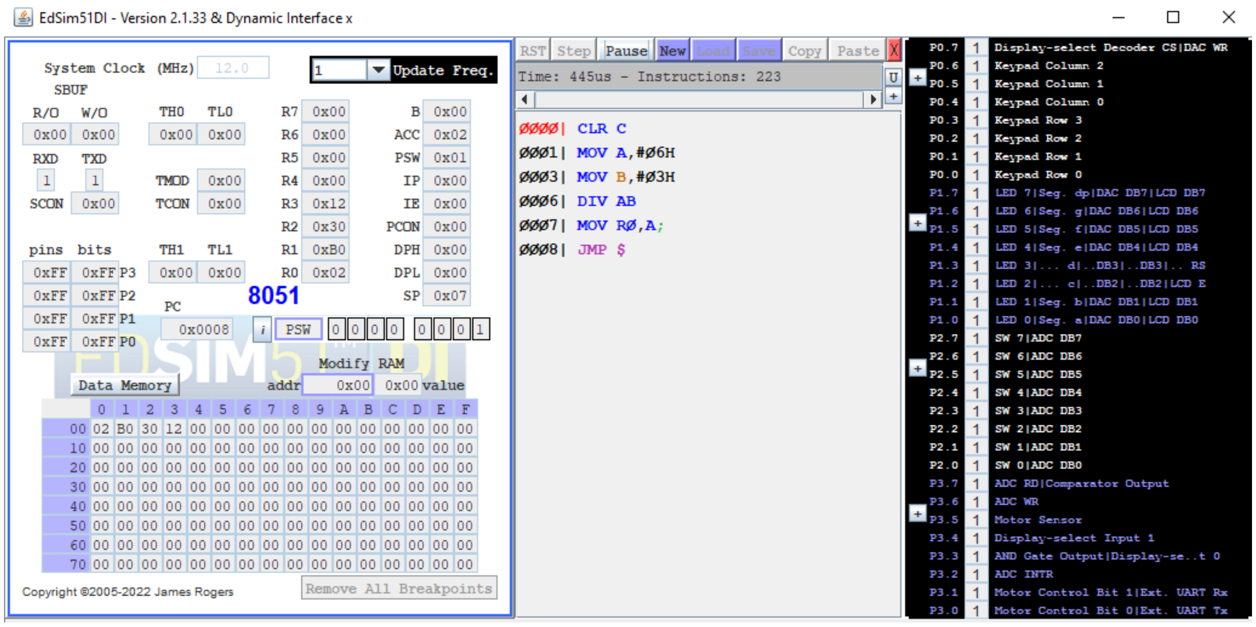
```
P0.7 1 Display-select Decoder CS|DAC WR
P0.6 1 Keypad Column 2
P0.5 1 Keypad Column 1
P0.4 1 Keypad Column 0
P0.3 1 Keypad Row 3
P0.2 1 Keypad Row 2
P0.1 1 Keypad Row 1
P0.0 1 Keypad Row 0
P1.7 1 LED 7|Seg. dp|DAC DB7|LCD DB7
P1.6 1 LED 6|Seg. g|DAC DB6|LCD DB6
P1.5 1 LED 5|Seg. f|DAC DB5|LCD DB5
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
```

Fig multiplication

4.Division

```
ORG 0X000
MOV A,#06H
MOV B,#03H
DIV AB
MOV R0,A
JMP $
```

OUTPUT:


EdSim51DI - Version 2.1.33 & Dynamic Interface x

System Clock (MHz) 12.0 — Update Freq. 1

SBUF R/O	0x00	W/O	0x00	TH0	0x00	TL0	0x00	R7	0x00	B	0x00
RXD	1	TXD	1	TMOD	0x00			R6	0x00	ACC	0x02
SCON	0x00			TCON	0x00			R5	0x00	PSW	0x01
								R4	0x00	IP	0x00
								R3	0x12	IE	0x00
								R2	0x30	PCON	0x00
				TH1	0x00	TL1	0x00	R1	0xB0	DPH	0x00
								R0	0x02	DPL	0x00
										SP	0x07

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

8051 — PC 0x0008 — PSW 0 0 0 0 0 0 0 1

Modify RAM: addr 0x00, value 0x00

Data Memory

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	02	B0	30	12	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	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 ©2005-2022 James Rogers

```
Time: 445us - Instructions: 223

ØØØØ| CLR C
ØØØ1| MOV A,#Ø6H
ØØØ3| MOV B,#Ø3H
ØØØ6| DIV AB
ØØØ7| MOV RØ,A;
ØØØ8| JMP $
```

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

Fig Division

RESULT:

EXP . 4A . Interfacing of common anode 7- segment display to display decimal numbers 0 t0 9 using Edsim-51 simulator.

AIM : To perform the given program which displays decimal numbers 0-9 on the 1st 7-segment display.

ALGORITHM:

1. Set the segment pins (P1) to display the pattern for 0 (0xC0).
2. Call the delay subroutine.
3. Set the segment pins (P1) to display the pattern for 1 (0xF9).
4. Call the delay subroutine.
5. Set the segment pins (P1) to display the pattern for 2 (0xA4).
6. Call the delay subroutine.
7. Set the segment pins (P1) to display the pattern for 3 (0xB0).
8. Call the delay subroutine.
9. Set the segment pins (P1) to display the pattern for 4 (0x99).
10. Call the delay subroutine.

11. Set the segment pins (P1) to display the pattern for 5 (0x92).
12. Call the delay subroutine.
13. Set the segment pins (P1) to display the pattern for 6 (0x82).
14. Call the delay subroutine.
15. Set the segment pins (P1) to display the pattern for 7 (0xF8).
16. Call the delay subroutine.
17. Set the segment pins (P1) to display the pattern for 8 (0x80).
18. Call the delay subroutine.
19. Set the segment pins (P1) to display the pattern for 9 (0x90).
20. Call the delay subroutine.
21. Jump back to the start of the code (JMP start).
22. Implement the delay subroutine:

    a. Initialize register R0 with a value of 0x10 (16).

    b. Decrement the value of R0 using the DJNZ instruction and jump back to the previous instruction if R0 is not zero.

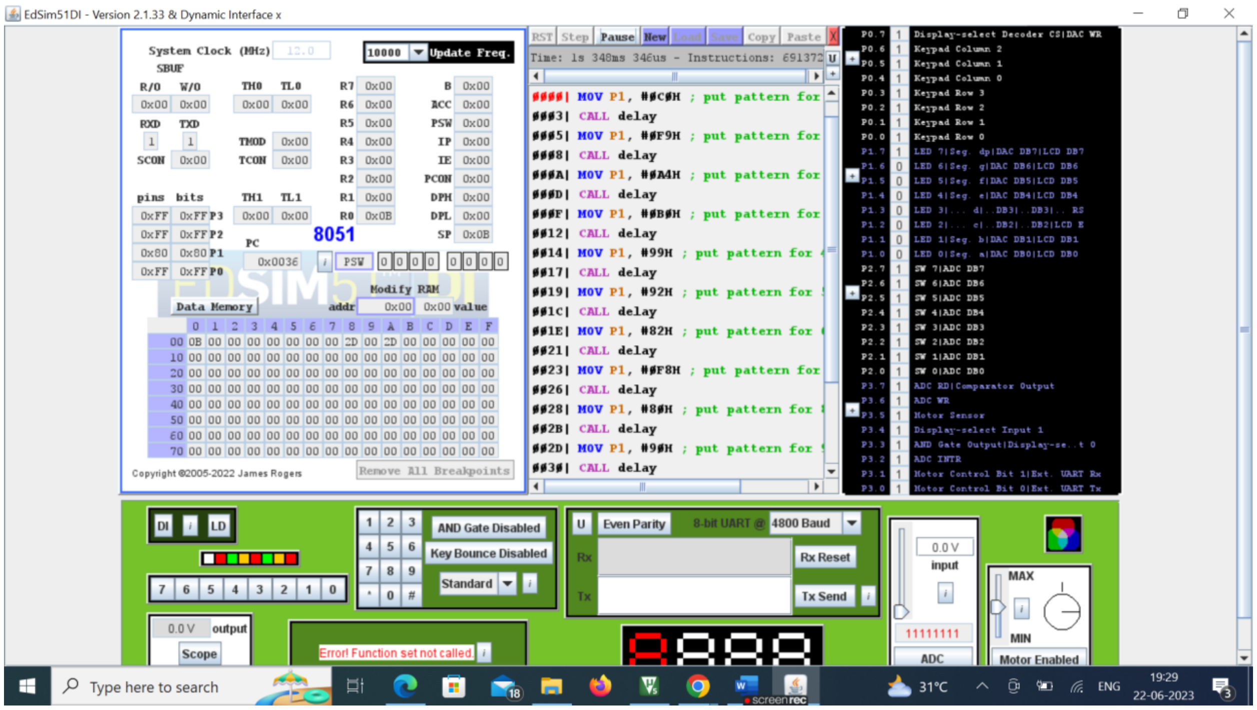
    c. Return from the subroutine (RET).

CODE:

```
start:
MOV P1, #0C0H ; put pattern for 0 on display
CALL delay
MOV P1, #0F9H ; put pattern for 1 on display
CALL delay
MOV P1, #0A4H ; put pattern for 2 on display
CALL delay
MOV P1, #0B0H ; put pattern for 3 on display
CALL delay
MOV P1, #99H ; put pattern for 4 on display
CALL delay
MOV P1, #92H ; put pattern for 5 on display
```

```
CALL delay
MOV P1, #82H ; put pattern for 6 on display
CALL delay
MOV P1, #0F8H ; put pattern for 7 on display
CALL delay
MOV P1, #80H ; put pattern for 8 on display
CALL delay
MOV P1, #90H ; put pattern for 9 on display
CALL delay
JMP start ; jump back to start
delay:
MOV R0, #10H
DJNZ R0, $
RET
```

OUTPUT:



EdSim51DI - Version 2.1.33 & Dynamic Interface x

```
Time: 1s 348ms 346us - Instructions: 691372
0000| MOV P1, #0C0H ; put pattern for
0003| CALL delay
0005| MOV P1, #0F9H ; put pattern for
0008| CALL delay
000A| MOV P1, #0A4H ; put pattern for
000D| CALL delay
000F| MOV P1, #0B0H ; put pattern for
0012| CALL delay
0014| MOV P1, #99H ; put pattern for
0017| CALL delay
0019| MOV P1, #92H ; put pattern for
001C| CALL delay
001E| MOV P1, #82H ; put pattern for
0021| CALL delay
0023| MOV P1, #0F8H ; put pattern for
0026| CALL delay
0028| MOV P1, #80H ; put pattern for
002B| CALL delay
002D| MOV P1, #90H ; put pattern for
0030| CALL delay
```

```
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  0  LED 6|Seg. g|DAC DB6|LCD DB6
P1.5  0  LED 5|Seg. f|DAC DB5|LCD DB5
P1.4  0  LED 4|Seg. e|DAC DB4|LCD DB4
P1.3  0  LED 3|... d|..DB3|..DB3|.. RS
P1.2  0  LED 2|... c|..DB2|..DB2|LCD E
P1.1  0  LED 1|Seg. b|DAC DB1|LCD DB1
P1.0  0  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
```

System Clock (MHz) 12.0 · 10000 Update Freq. · PC 0x0036 · 8051

Error! Function set not called

RESULT:

EXP.4B. Multiplexing the number 1234 on the four 7-segment displays using Edsim-51 simulator.

AIM : To implement the program for Multiplexing the number 1234 on the four 7-segment displays using Edsim-51 simulator.

ALGORITHM:

1. Set the P3.3 pin to high to enable display 3.
2. Set the P3.4 pin to high to enable display 2.
3. Set the segment pins (P1) to display the pattern for 1 (0b11111001).
4. Call the delay subroutine.
5. Clear the P3.3 pin to disable display 3.
6. Set the P3.4 pin to high to enable display 2.
7. Set the segment pins (P1) to display the pattern for 2 (0b10100100).
8. Call the delay subroutine.
9. Clear the P3.4 pin to disable display 2.
10. Set the P3.3 pin to high to enable display 1.
11. Set the segment pins (P1) to display the pattern for 3 (0b10110000).
12. Call the delay subroutine.
13. Clear the P3.3 pin to disable display 1.
14. Clear the P3.4 pin to disable display 0.
15. Set the segment pins (P1) to display the pattern for 4 (0b10011001).
16. Call the delay subroutine.
17. Jump back to the start of the code (JMP start).
18. Implement the delay subroutine:

    a. Initialize register R0 with a value of 0x0F (15).

    b. Decrement the value of R0 using the DJNZ instruction and jump back to the previous instruction if R0 is not zero.

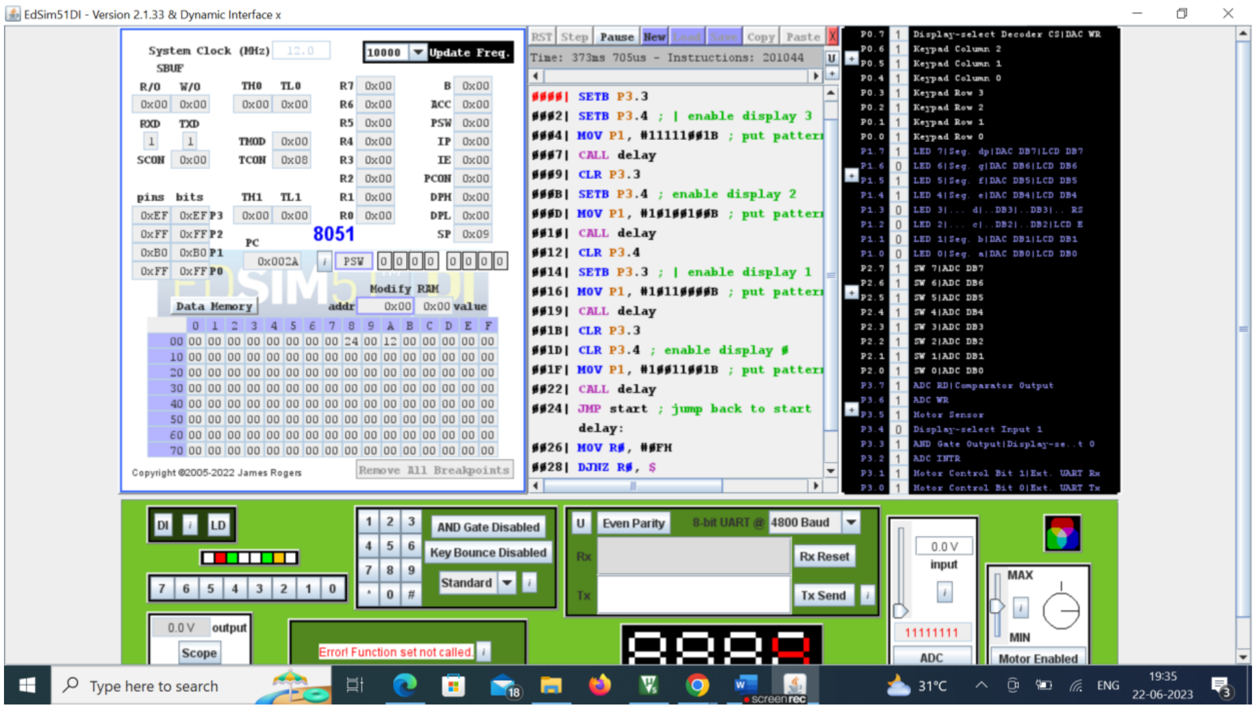
    c. Return from the subroutine (RET).

CODE :

```
start:

SETB P3.3

SETB P3.4 ; | enable display 3

MOV P1, #11111001B ; put pattern for 1 on display
```

```
CALL delay
CLR P3.3
SETB P3.4 ; enable display 2
MOV P1, #10100100B ; put pattern for 2 on display
CALL delay
CLR P3.4
SETB P3.3 ; | enable display 1
MOV P1, #10110000B ; put pattern for 3 on display
CALL delay
CLR P3.3
CLR P3.4 ; enable display 0
MOV P1, #10011001B ; put pattern for 4 on display
CALL delay
JMP start ; jump back to start
delay:
MOV R0, #0FH
DJNZ R0, $
RET
```

OUTPUT:



```
Time: 373ms 705us - Instructions: 201044
0000| SETB P3.3
0002| SETB P3.4 ; | enable display 3
0004| MOV P1, #11111001B ; put pattern
0007| CALL delay
0009| CLR P3.3
000B| SETB P3.4 ; enable display 2
000D| MOV P1, #10100100B ; put pattern
0010| CALL delay
0012| CLR P3.4
0014| SETB P3.3 ; | enable display 1
0016| MOV P1, #10110000B ; put pattern
0019| CALL delay
001B| CLR P3.3
001D| CLR P3.4 ; enable display 0
001F| MOV P1, #10011001B ; put pattern
0022| CALL delay
0024| JMP start ; jump back to start
      delay:
0026| MOV R0, #0FH
0028| DJNZ R0, $
```

```
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 0 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 0 LED 3|... d|..DB3|..DB3|.. RS
P1.2 0 LED 2|... c|..DB2|..DB2|LCD E
P1.1 0 LED 1|Seg. b|DAC DB1|LCD DB1
P1.0 0 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 0 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
```

Error! Function set not called.

RESULT:

EMBEDDED WITH C PROGRAMS

EXP.5 Toggles an LED connected to pin P1.0 of the 8051 microcontroller.

AIM: To Perform Toggles an LED connected to pin P1.0 of the 8051 microcontroller.

ALGORITHM:

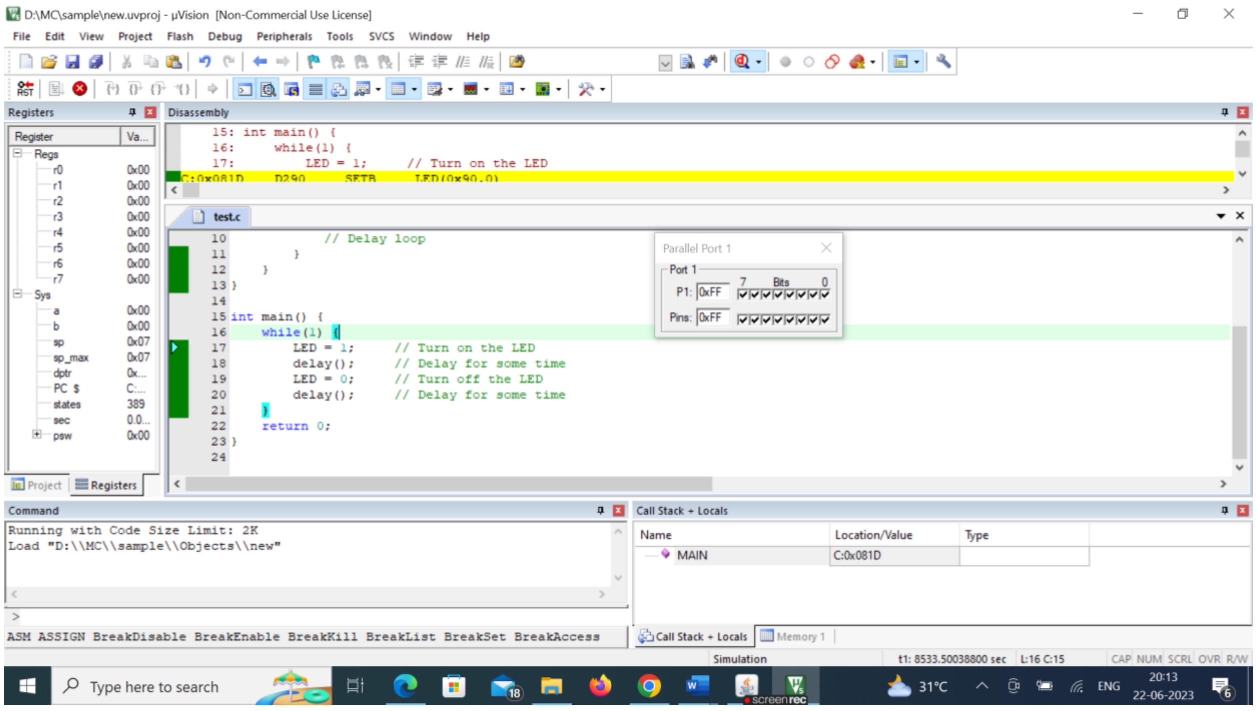
1. Include the header file "reg52.h" to access the register definitions for the 8051 microcontroller.
2. Define the necessary I/O port and pin configurations using the "sbit" keyword. For example, here the LED is connected to pin P1.0.
3. Define the delay function.
4. Declare two unsigned int variables "i" and "j" for loop counters.
5. Use nested for loops to introduce a delay. The outer loop iterates 500 times, and the inner loop iterates 500 times. Adjust the loop counts based on the desired delay duration.
6. Return from the delay function.

7. Define the main function.
8. Enter an infinite loop using the "while(1)" condition.
9. Set the LED pin to high (1) to turn on the LED.
10. Call the delay function to introduce a delay.
11. Set the LED pin to low (0) to turn off the LED.
12. Call the delay function to introduce a delay.
13. Repeat steps 9 to 12 indefinitely.
14. Return 0 to indicate successful program execution.

CODE :

```
#include <reg52.h>   // Include the header file for 8051 microcontroller

// Define the necessary I/O port and pin configurations
sbit LED = P1^0;     // Example: LED connected to P1.0 pin

void delay() {
    unsigned int i, j;
    for(i = 0; i < 500; i++) {
        for(j = 0; j < 500; j++) {
            // Delay loop
        }
    }
}

int main() {
    while(1) {
        LED = 1;     // Turn on the LED
        delay();     // Delay for some time
        LED = 0;     // Turn off the LED
        delay();     // Delay for some time
    }
    return 0;
}
```

OUTPUT:



D:\MC\sample\new.uvproj - µVision [Non-Commercial Use License]

Registers

```
Register    Va...
Regs
  r0        0x00
  r1        0x00
  r2        0x00
  r3        0x00
  r4        0x00
  r5        0x00
  r6        0x00
  r7        0x00
Sys
  a         0x00
  b         0x00
  sp        0x07
  sp_max    0x07
  dptr      0x...
  PC $      C:...
  states    389
  sec       0.0...
  psw       0x00
```

Disassembly

```
    15: int main() {
    16:     while(1) {
    17:         LED = 1;      // Turn on the LED
C:0x081D    D290      SETB     LED(0x90.0)
```

test.c

```
                // Delay loop
            }
        }
}

int main() {
    while(1) {
        LED = 1;      // Turn on the LED
        delay();      // Delay for some time
        LED = 0;      // Turn off the LED
        delay();      // Delay for some time
    }
    return 0;
}
```

Parallel Port 1

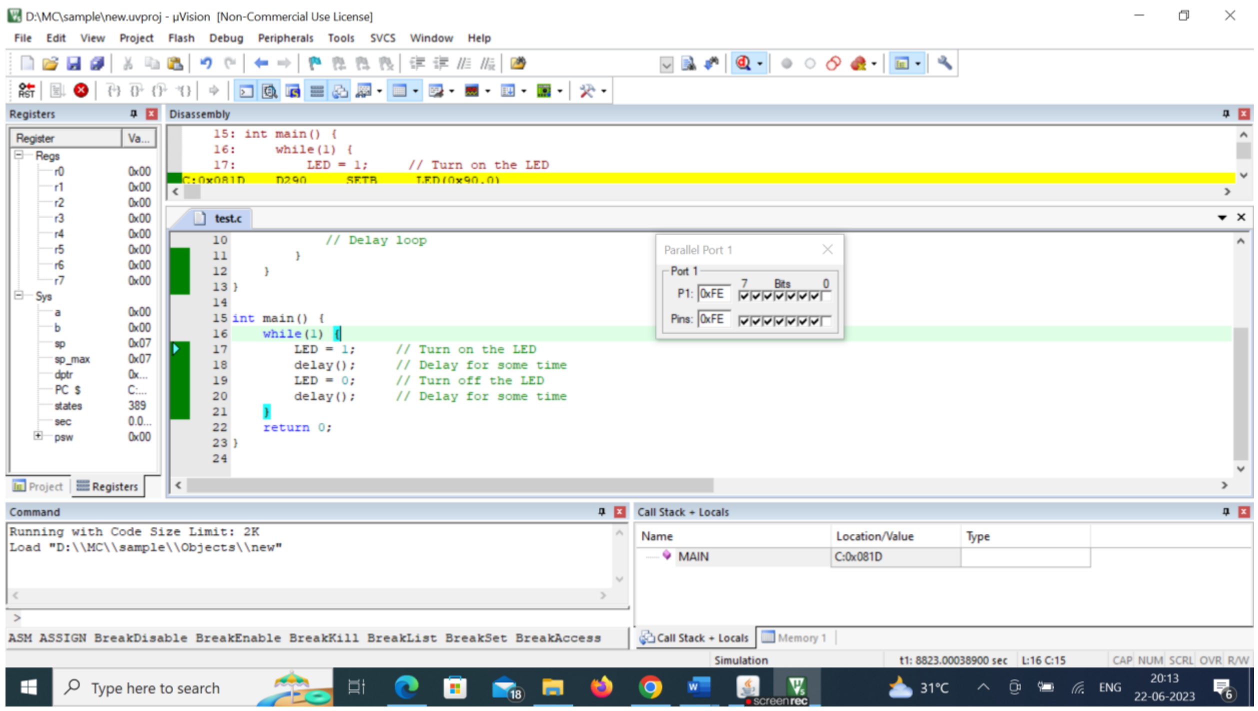
Port 1 — P1: 0xFF, Bits 7…0: ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓
Pins: 0xFF, ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓

Command

```
Running with Code Size Limit: 2K
Load "D:\\MC\\sample\\Objects\\new"
>
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess
```

Call Stack + Locals

Name	Location/Value	Type
MAIN	C:0x081D	


D:\MC\sample\new.uvproj - µVision [Non-Commercial Use License]
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
test.c
Parallel Port 1
Project
Registers
Command
Call Stack + Locals
Memory 1
Simulation
Type here to search

Register	Va...
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x07
sp_max	0x07
dptr	0x...
PC $	C:...
states	389
sec	0.0...
psw	0x00

```
    15: int main() {
    16:     while(1) {
    17:         LED = 1;      // Turn on the LED
C:0x081D    D290      SETB      LED(0x90.0)
```

```
                // Delay loop
            }
        }
}

int main() {
    while(1) {
        LED = 1;      // Turn on the LED
        delay();      // Delay for some time
        LED = 0;      // Turn off the LED
        delay();      // Delay for some time
    }
    return 0;
}
```

Port 1
P1: 0xFE 7 Bits 0
Pins: 0xFE

```
Running with Code Size Limit: 2K
Load "D:\\MC\\sample\\Objects\\new"
```

```
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess
```

Name	Location/Value	Type
MAIN	C:0x081D	

t1: 8823.00038900 sec L:16 C:15 CAP NUM SCRL OVR R/W

31°C ENG 20:13 22-06-2023

RESULT:

EXP.6 Write Basic and arithmetic Programs Using Embedded C in keil

AIM: To write the Embedded C programs using in keil software

ALGORITHM:

1. Start the program.
2. Declare the necessary variables for the arithmetic operation (e.g., num1, num2, result).
3. Prompt the user to enter the input values (e.g., num1, num2).
4. Read the input values from the user.
5. Perform the desired arithmetic operation (e.g., addition, subtraction, multiplication, division) using the input values.
6. Store the result in the 'result' variable.
7. Display the result to the user.
8. End the program.

CODE 1 : Program to find the sum of two numbers:

```
#include <stdio.h>
```

```
int main() {
    int num1, num2, sum;

    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);

    sum = num1 + num2;

    printf("Sum: %d\n", sum);

    while (1) {}  // To keep the program running in Keil

    return 0;
}
```

OUTPUT:

RESULT:

CODE 2: Program to check if a number is even or odd:

```
#include <stdio.h>

int main() {
    int num;

    printf("Enter a number: ");
    scanf("%d", &num);

    if (num % 2 == 0) {
        printf("%d is even.\n", num);
    } else {
        printf("%d is odd.\n", num);
    }

    while (1) {}  // To keep the program running in Keil

    return 0;
}
```

OUTPUT:

RESULT:

CODE 3: Program to find the factorial of a number:

```
#include <stdio.h>

int main() {
    int num, factorial = 1;

    printf("Enter a number: ");
    scanf("%d", &num);

    for (int i = 1; i <= num; i++) {
        factorial *= i;
    }

    printf("Factorial: %d\n", factorial);

    while (1) {}  // To keep the program running in Keil

    return 0;
```

}

OUTPUT:

RESULT:

EXP NO: 7.A. Write a C Program to send values 00-FF to port P1.

AIM : To Implement a C Program to send values 00-FF to port P1.

ALGORITHM:

1. Include the necessary header file:
2. Define the main function:
3. Declare an unsigned char variable **z**:
4. Execute a for loop from 0 to 255:
5. Assign the value of **z** to the P1 register:

PROGRAM:

```
# include <reg51.h>
void main (void)
{
unsigned char z;
for(z=0;z<=255;z++)
P1=z;
}
```

OUTPUT:

P1 displays values 00-FFH in binary.

RESULT:

EXP NO: 7.B. Write an 8051 C Program to send hex values for ASCII characters of 0,1,2,3,4,5,A,B,C,D to Port P1.

AIM: To Write an 8051 C Program to send hex values for ASCII characters of 0,1,2,3,4,5,A,B,C,D to Port P1.

ALGORITHM:

1. Include the necessary header file:

2. Define the main function:
3. Declare an unsigned char array **mynum** and initialize it with a string of characters:
4. Declare an unsigned char variable **z**:
5. Execute a for loop from 0 to 10:
6. Assign the value at index **z** of the **mynum** array to the P1 register:

CODE :

```
#include <reg51.h>
void main(void)
{
unsigned char mynum[] =”012345ABCD”;
unsigned char z;
for (z=0;z<=10;z++)
P1=mynum[z];
}
```

OUTPUT:

P1 displays values 30H,31H,32H,33H,34H,35H,41H,42H,43H and 44H, the hex values for ASCII 0,1,2,and so on.

RESULT:

EXP NO: 7.C. Write an 8051 C Program to toggle all the bits of P1 continuously.

AIM: To Write an 8051 C Program to toggle all the bits of P1 continuously.

ALGORITHM:

1. Include the necessary header file reg51.h for the 8051 microcontroller.
2. Implement the main() function.
3. Start an infinite loop using for (;;) or while(1).
4. Inside the loop, assign the hex value 0x55 to Port P1.
    - Set P1 to 0x55 by assigning it the value 0*55. The 0* prefix indicates that the value is in hexadecimal (binary).
5. Assign the hex value 0xAA to Port P1.

- Set P1 to 0xAA by assigning it the value 0*AA. Again, the 0* prefix indicates the hexadecimal representation.

6. Repeat the loop indefinitely.

CODE:

```
#include <reg51.h>
void main (void)
{
for (;;)  //repeat forever
{
P1=0*55;   //0* indicates the data is in hex (binary)
P1=0*AA;
}
}
```

OUTPUT:

Port P1 toggles continuously.

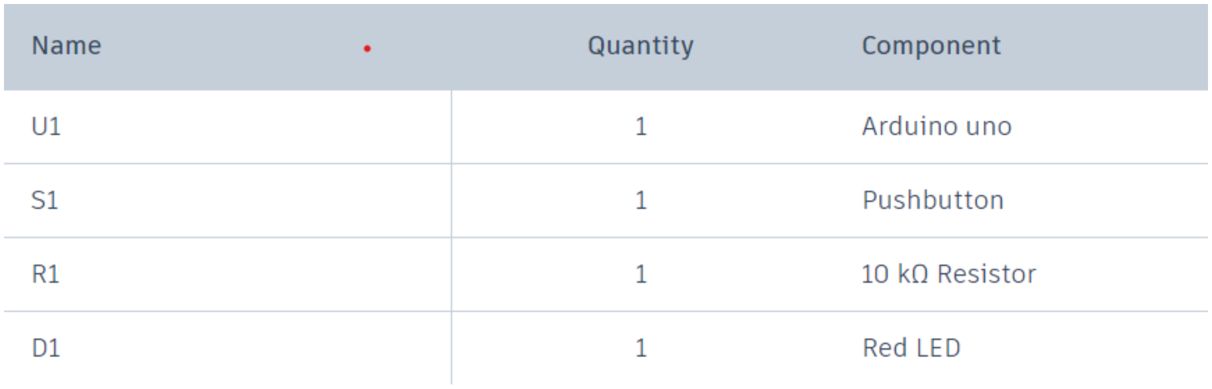
RESULT:

EXP. 8. BLINKING OF LED USING ARDUINO PROGRAMMING

AIM:

To create a Circuit using Arduino Board to control the LED On/OFF Using Switch.

Components Required:



Name	Quantity	Component
U1	1	Arduino uno
S1	1	Pushbutton
R1	1	10 kΩ Resistor
D1	1	Red LED

ALGORITHM:

Step 1: Set up the hardware

- Connect the longer leg (anode) of the LED to a digital pin on the Arduino board (e.g., pin 13).
- Connect the shorter leg (cathode) of the LED to the ground (GND) pin on the Arduino board using the current-limiting resistor.

Step 2: Set the pin mode

In the Arduino setup function, set the pin mode for the LED pin to OUTPUT. This will configure the pin to send output signals to the LED.

Step 3: Blink the LED

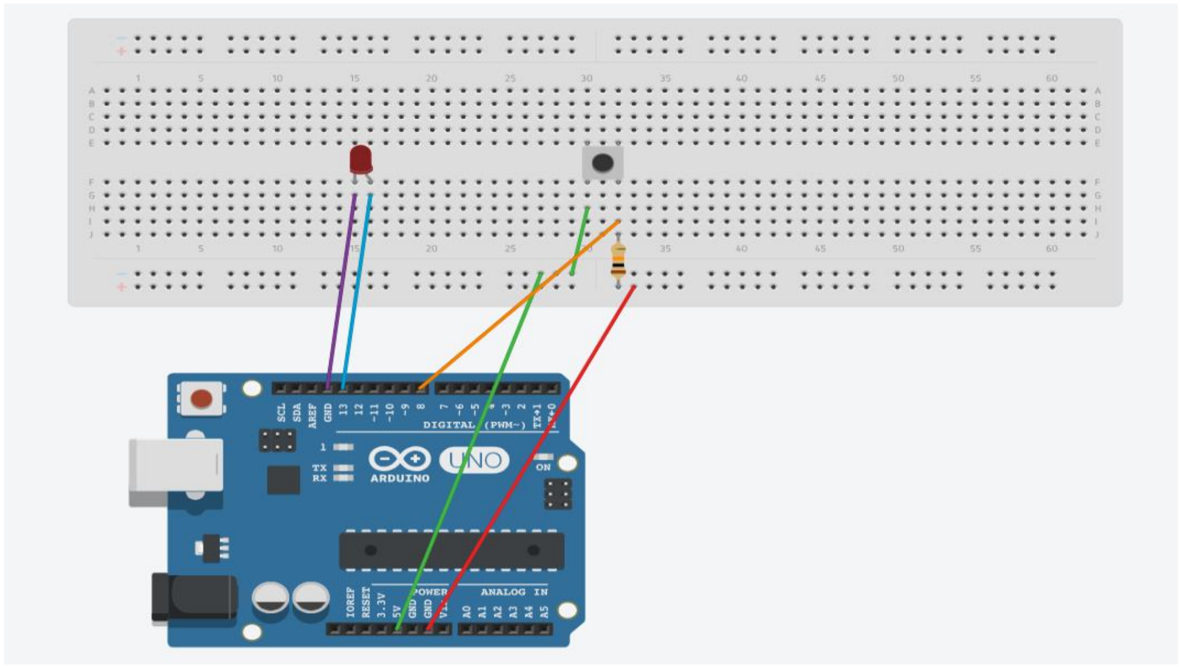
- In the Arduino loop function, turn the LED ON by setting the digital pin HIGH.
- Add a short delay using the delay() function to keep the LED on for a specific duration (e.g., 1000 milliseconds or 1 second).
- Turn the LED OFF by setting the digital pin LOW.
- Add another delay to keep the LED off for the same duration as before.

Step 4: Repeat

- The loop function runs continuously, so the LED will keep blinking ON and OFF in a loop.

CODE :

```
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int ledPin = 13;
int switchPin = 8;
boolean lastButton= LOW;
boolean currentButton= LOW;
boolean ledOn= false;
// the setup routine runs once when you press reset:
void setup()
{
// initialize the digital pin as an output.
pinMode(8, INPUT);
pinMode(13,OUTPUT);
}
boolean debounce(boolean last)
{
boolean current= digitalRead(switchPin);
if(last != current)
{
delay(15);
current=digitalRead(switchPin);
}
return current;
}

// the loop routine runs over and over again forever:
void loop() {
currentButton=debounce(lastButton);
if(lastButton==LOW &&currentButton==HIGH)
{
ledOn = !ledOn;
}
lastButton= currentButton;
digitalWrite(ledPin,ledOn);
}
```



RESULT:

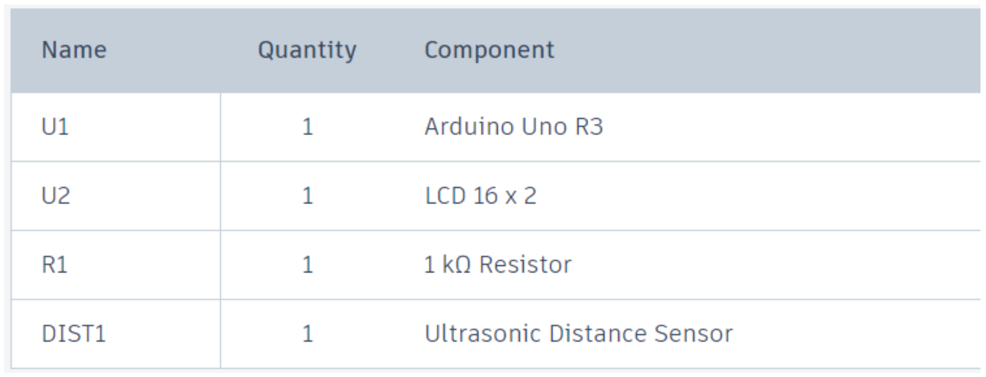
Thus the LED Blink control using Arduino is executed successfully.

EXP NO 9 : MEASURING DISTANCE USING ARDUINO AND ULTRASONIC SENSORS

AIM :

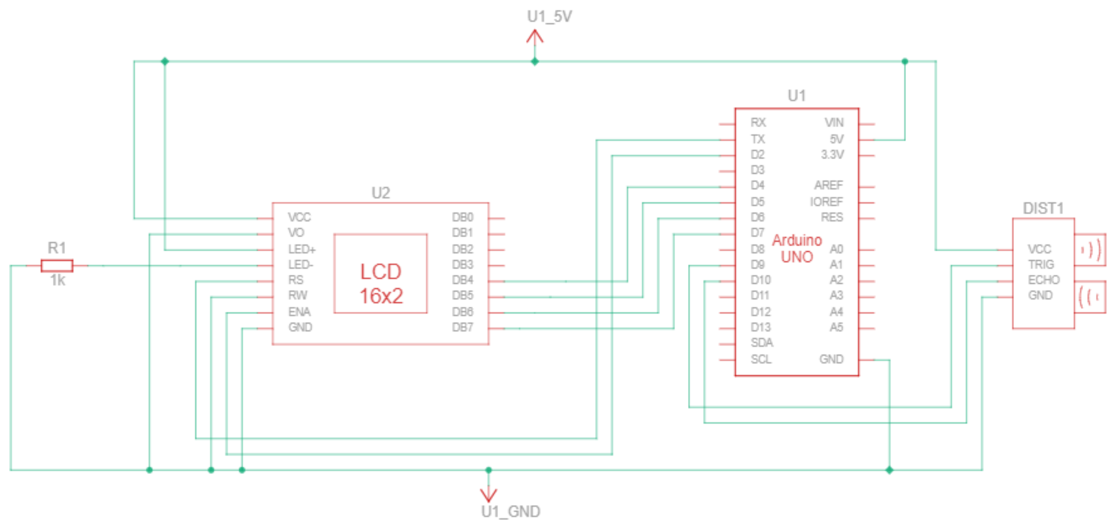
To measure the distance of an object using Ultrasonic Sensors and Arduino.

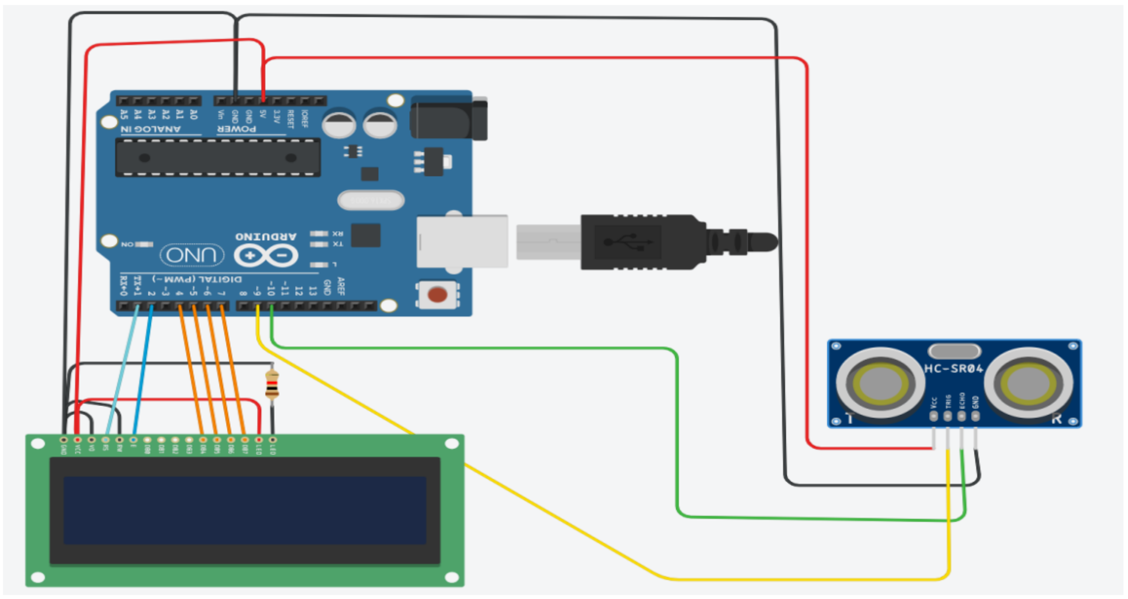
Components Required:



Name	Quantity	Component
U1	1	Arduino Uno R3
U2	1	LCD 16 x 2
R1	1	1 kΩ Resistor
DIST1	1	Ultrasonic Distance Sensor

Circuit Diagram:

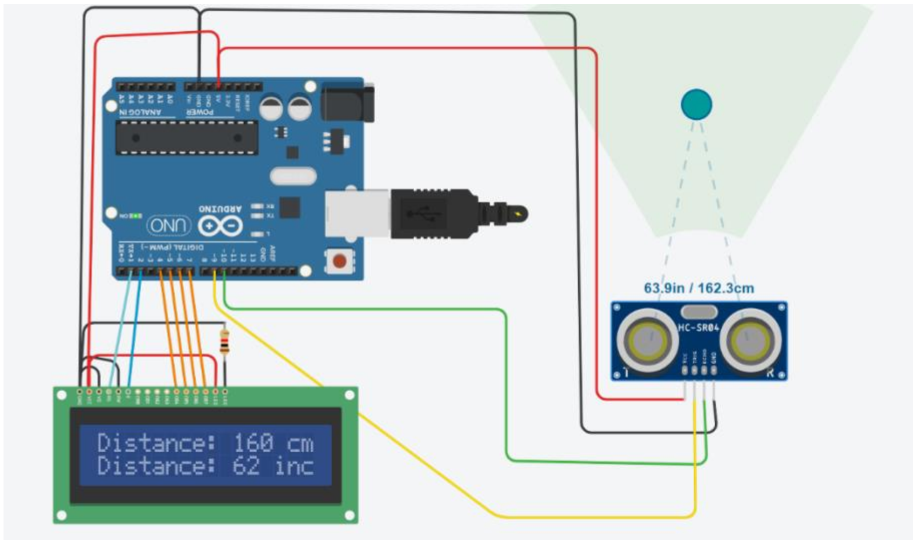




Code:

```
#include <LiquidCrystal.h> // includes the LiquidCrystal Library
LiquidCrystal lcd(1, 2, 4, 5, 6, 7); // Creates an LCD object. Parameters: (rs, enable, d4, d5, d6,
d7)
const int trigPin = 9;
const int echoPin = 10;
long duration;
int distanceCm, distanceInch;
void setup() {
lcd.begin(16,2); // Initializes the interface to the LCD screen, and specifies the dimensions (width
and height) of the display
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
}
void loop() {
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distanceCm= duration*0.034/2;
distanceInch = duration*0.0133/2;
lcd.setCursor(0,0); // Sets the location at which subsequent text written to the LCD will be
displayed
lcd.print("Distance: "); // Prints string "Distance" on the LCD
lcd.print(distanceCm); // Prints the distance value from the sensor
lcd.print(" cm");
delay(10);
lcd.setCursor(0,1);
lcd.print("Distance: ");
lcd.print(distanceInch);
lcd.print(" inch");
delay(10);
}
```

OUTPUT :



```
Distance: 160 cm
Distance: 62 inc
```

RESULT:

Thus the program for Distance calculation is executed Successfully.

EXP.NO 10 : Light and sound

AIM : Task: A LED and a piezo speaker are supposed to blink or beep continuously

ALGORITHM:

```
1. Initialize the LED and piezo speaker pins as OUTPUT.
2. Enter a loop that runs indefinitely (the main loop).
3. Inside the main loop:
    a. Turn ON the LED.
    b. Generate a beep on the piezo speaker.
    c. Wait for a specific duration (blink and beep time).
    d. Turn OFF the LED.
    e. Stop the piezo speaker sound.
```

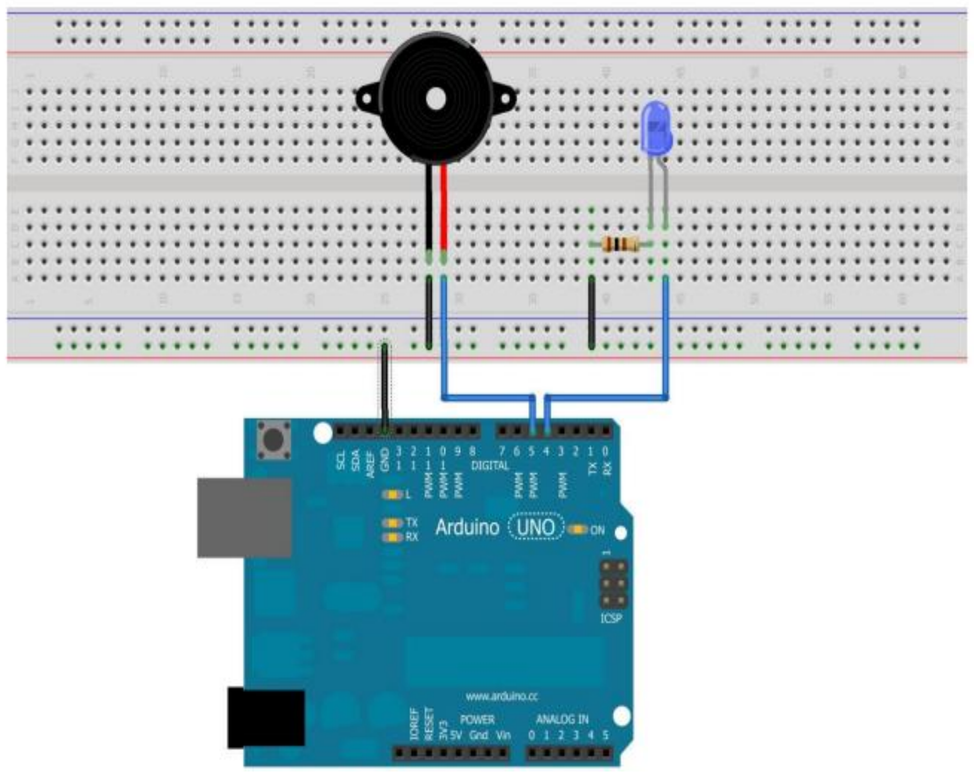
f. Wait for a specific duration (the interval between blinks and beeps).

g. Go back to step 3 (repeat the loop).

COMPONENTS REQUIRED:

Required equipment: Microcontroller / one LED / resistor with 200 Ohm / Breadboard / piezo speaker / cables

Setup:



CODE :

```
int LED=4; //this time we also going to use the first part of the program. Here
//we are going to put in variables. This means that there will be a letter or a
//word standing for a number. In this example the LED is connected to pin 4 and
//the speaker to pin 5, so we rename pin 4 and pin 5, to avoid confusion. The
//word “LED” now stands for the number 4 and the word “beep” for the number 5.
int beep=5;
void setup()
{ //We are starting with the setup
pinMode(LED, OUTPUT); //pin 4 (pin “LED”) is supposed to be an output
pinMode(beep, OUTPUT); //Pin 5 (pin “beep”) is supposed to be an output
}
void loop()
{ //The main part starts
digitalWrite(LED, HIGH); //turn on the LED
digitalWrite(beep, HIGH); //turn on the speaker
delay(1000); //wait for 1000 milliseconds (sound and light)
digitalWrite(LED, LOW); //turn off the LED
digitalWrite(beep, LOW); //turn off the speaker
delay(1000); //wait for 1000 milliseconds (no sound and no light)
} //Here at the end of the loop the program starts again from the beginning of
//the loop. So it will beep and light up again. If you change the break (delay)
//it will be either beep and light up faster or slower.
```

RESULT: