

## 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

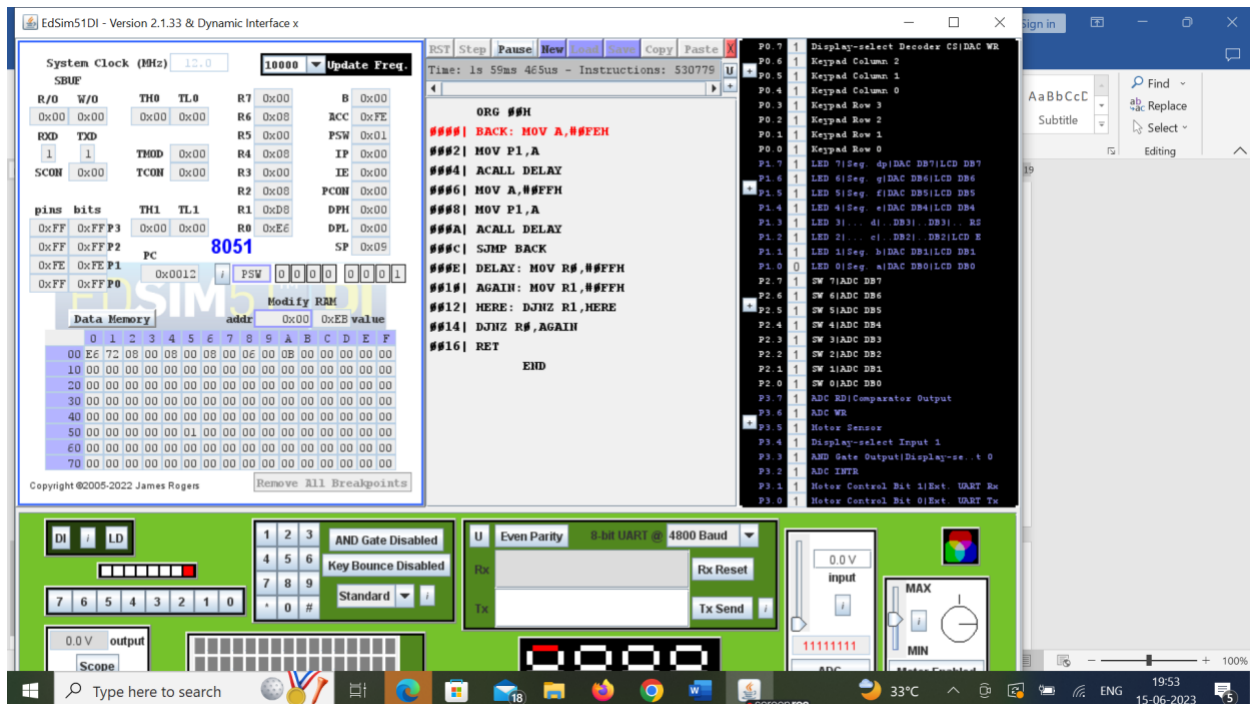
1. Start at address 0x0000 (ORG 0x0000).
2. Set up a continuous loop labeled as "BACK."
3. Move the value 0xFE (254 in decimal) to the accumulator register (A).
4. Move the value in the accumulator (A) to Port 1 (P1) to turn on the LED.
5. Call the delay subroutine by using the instruction "ACALL DELAY."
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:



## 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
```

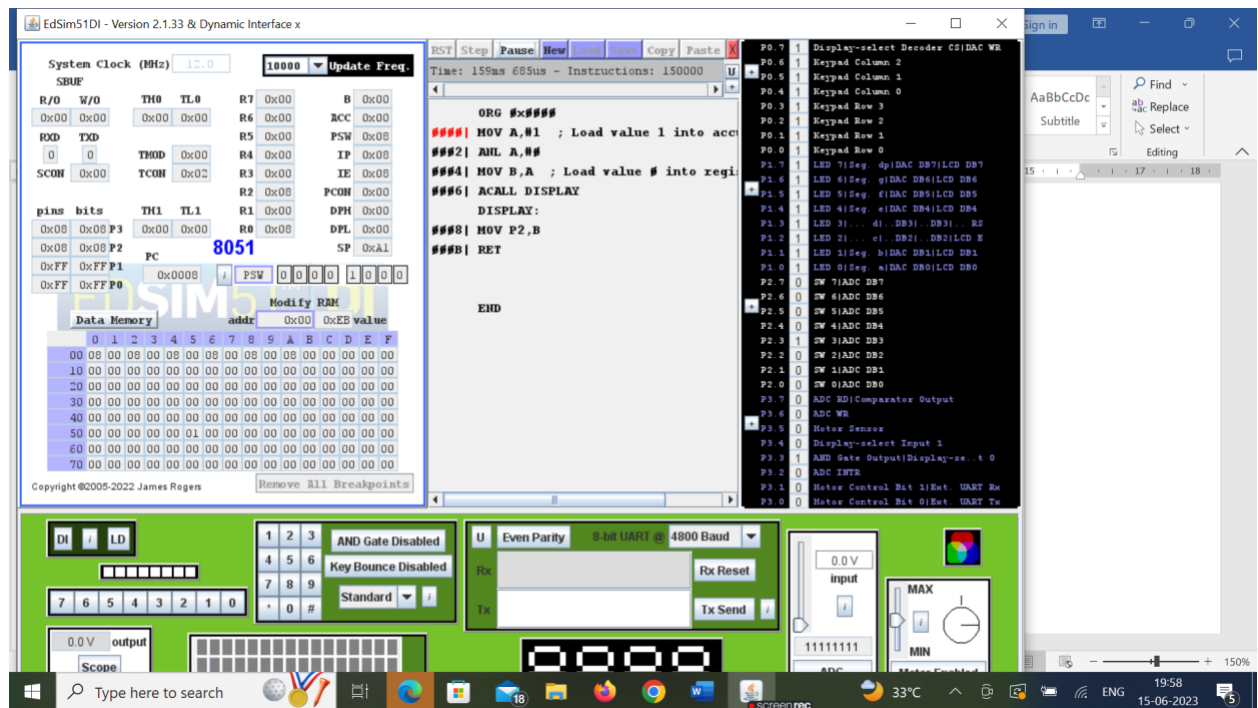
```
DISPLAY:
```

```
MOV P2,B
```

```
RET
```

```
END
```

### OUTPUT:



### 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":
  - 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

Update Freq. 1

RST Step Run New Load Save Copy Paste

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

PC 8051

Modify RAM

Data Memory

addr	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
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

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. g|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:

The screenshot displays the EdSim51DI software interface, Version 2.1.33 & Dynamic Interface x. The main window is divided into several sections:

- System Clock (MHz):** Set to 12.0.
- SBUF:** A table showing R/O and W/O values for various ports (RXD, TXD, SC0N, P0, P1, P2, P3).
- Registers:** A table showing the values of various registers (R0-R7, ACC, PSW, IP, IE, PCON, DPH, DPL, SP).
- PC:** Program Counter value is 8051.
- Modify RAM:** A table showing the values of RAM locations (0-70).
- Assembly Code:** A list of instructions being executed, including MOV, INC, and DJNZ.
- I/O Devices:** A list of I/O devices and their status (P0.0-P0.7, P1.0-P1.7, P2.0-P2.7, P3.0-P3.7).

The assembly code being executed is:

```
0000 MOV R0,#30h
0002 MOV R1,#40h
0004 MOV R7,#05h
0006 BACK:MOV A,@R0
0007 MOV R4,A
0008 MOV A,@R1
0009 MOV A,R4
000A MOV @R0,A
000B MOV A,R4
000C MOV @R1,A
000D INC R0
000E INC R1
000F DJNZ R7,BACK
END
```

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
- Move the value 0x06 into the Data Pointer (DPTR) using the instruction
- Move the value at the external memory location pointed to by DPTR into the accumulator A using the instruction
- Move the value in the accumulator A to register R1 using the instruction
- 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 "MOVB 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
- Move the value 0x4062 into the Data Pointer (DPTR) using the instruction
- Move the value in register R1 to the accumulator A using the instruction
- Move the value in the accumulator A to the external memory location pointed to by DPTR using the instruction "
- 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:

The screenshot displays the EdSim51DI software interface, which is used for simulating 8051 microcontroller programs. The main window is divided into several sections:

- System Configuration:** Shows the System Clock (MHz) set to 12.0 and the Update Freq. set to 1.
- Registers and Special Function Registers (SFRs):** A table showing the current values of various registers. For example, R0 is 0x31, R1 is 0x00, R2 is 0x00, R3 is 0x06, and the Program Counter (PC) is 8051.
- Assembly Code Window:** Displays the assembly code being executed. The code includes instructions like MOV R3,#06, MOV DPTR,#6, MOVX A,@DPTR, 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, and END.
- Hardware Status Window:** Shows the status of various hardware components, including P0, P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11, P12, P13, P14, P15, P16, P17, P18, P19, P20, P21, P22, P23, P24, P25, P26, P27, P28, P29, P30, P31, P32, P33, P34, P35, P36, P37, P38, P39, P40, P41, P42, P43, P44, P45, P46, P47, P48, P49, P50, P51, P52, P53, P54, P55, P56, P57, P58, P59, P60, P61, P62, P63, P64, P65, P66, P67, P68, P69, P70, P71, P72, P73, P74, P75, P76, P77, P78, P79, P80, P81, P82, P83, P84, P85, P86, P87, P88, P89, P90, P91, P92, P93, P94, P95, P96, P97, P98, P99, P100, P101, P102, P103, P104, P105, P106, P107, P108, P109, P110, P111, P112, P113, P114, P115, P116, P117, P118, P119, P120, P121, P122, P123, P124, P125, P126, P127, P128, P129, P130, P131, P132, P133, P134, P135, P136, P137, P138, P139, P140, P141, P142, P143, P144, P145, P146, P147, P148, P149, P150, P151, P152, P153, P154, P155, P156, P157, P158, P159, P160, P161, P162, P163, P164, P165, P166, P167, P168, P169, P170, P171, P172, P173, P174, P175, P176, P177, P178, P179, P180, P181, P182, P183, P184, P185, P186, P187, P188, P189, P190, P191, P192, P193, P194, P195, P196, P197, P198, P199, P200, P201, P202, P203, P204, P205, P206, P207, P208, P209, P210, P211, P212, P213, P214, P215, P216, P217, P218, P219, P220, P221, P222, P223, P224, P225, P226, P227, P228, P229, P230, P231, P232, P233, P234, P235, P236, P237, P238, P239, P240, P241, P242, P243, P244, P245, P246, P247, P248, P249, P250, P251, P252, P253, P254, P255, P256, P257, P258, P259, P260, P261, P262, P263, P264, P265, P266, P267, P268, P269, P270, P271, P272, P273, P274, P275, P276, P277, P278, P279, P280, P281, P282, P283, P284, P285, P286, P287, P288, P289, P290, P291, P292, P293, P294, P295, P296, P297, P298, P299, P300, P301, P302, P303, P304, P305, P306, P307, P308, P309, P310, P311, P312, P313, P314, P315, P316, P317, P318, P319, P320, P321, P322, P323, P324, P325, P326, P327, P328, P329, P330, P331, P332, P333, P334, P335, P336, P337, P338, P339, P340, P341, P342, P343, P344, P345, P346, P347, P348, P349, P350, P351, P352, P353, P354, P355, P356, P357, P358, P359, P360, P361, P362, P363, P364, P365, P366, P367, P368, P369, P370, P371, P372, P373, P374, P375, P376, P377, P378, P379, P380, P381, P382, P383, P384, P385, P386, P387, P388, P389, P390, P391, P392, P393, P394, P395, P396, P397, P398, P399, P400, P401, P402, P403, P404, P405, P406, P407, P408, P409, P410, P411, P412, P413, P414, P415, P416, P417, P418, P419, P420, P421, P422, P423, P424, P425, P426, P427, P428, P429, P430, P431, P432, P433, P434, P435, P436, P437, P438, P439, P440, P441, P442, P443, P444, P445, P446, P447, P448, P449, P450, P451, P452, P453, P454, P455, P456, P457, P458, P459, P460, P461, P462, P463, P464, P465, P466, P467, P468, P469, P470, P471, P472, P473, P474, P475, P476, P477, P478, P479, P480, P481, P482, P483, P484, P485, P486, P487, P488, P489, P490, P491, P492, P493, P494, P495, P496, P497, P498, P499, P500, P501, P502, P503, P504, P505, P506, P507, P508, P509, P510, P511, P512, P513, P514, P515, P516, P517, P518, P519, P520, P521, P522, P523, P524, P525, P526, P527, P528, P529, P530, P531, P532, P533, P534, P535, P536, P537, P538, P539, P540, P541, P542, P543, P544, P545, P546, P547, P548, P549, P550, P551, P552, P553, P554, P555, P556, P557, P558, P559, P560, P561, P562, P563, P564, P565, P566, P567, P568, P569, P570, P571, P572, P573, P574, P575, P576, P577, P578, P579, P580, P581, P582, P583, P584, P585, P586, P587, P588, P589, P590, P591, P592, P593, P594, P595, P596, P597, P598, P599, P600, P601, P602, P603, P604, P605, P606, P607, P608, P609, P610, P611, P612, P613, P614, P615, P616, P617, P618, P619, P620, P621, P622, P623, P624, P625, P626, P627, P628, P629, P630, P631, P632, P633, P634, P635, P636, P637, P638, P639, P640, P641, P642, P643, P644, P645, P646, P647, P648, P649, P650, P651, P652, P653, P654, P655, P656, P657, P658, P659, P660, P661, P662, P663, P664, P665, P666, P667, P668, P669, P670, P671, P672, P673, P674, P675, P676, P677, P678, P679, P680, P681, P682, P683, P684, P685, P686, P687, P688, P689, P690, P691, P692, P693, P694, P695, P696, P697, P698, P699, P700, P701, P702, P703, P704, P705, P706, P707, P708, P709, P710, P711, P712, P713, P714, P715, P716, P717, P718, P719, P720, P721, P722, P723, P724, P725, P726, P727, P728, P729, P730, P731, P732, P733, P734, P735, P736, P737, P738, P739, P740, P741, P742, P743, P744, P745, P746, P747, P748, P749, P750, P751, P752, P753, P754, P755, P756, P757, P758, P759, P760, P761, P762, P763, P764, P765, P766, P767, P768, P769, P770, P771, P772, P773, P774, P775, P776, P777, P778, P779, P780, P781, P782, P783, P784, P785, P786, P787, P788, P789, P790, P791, P792, P793, P794, P795, P796, P797, P798, P799, P800, P801, P802, P803, P804, P805, P806, P807, P808, P809, P810, P811, P812, P813, P814, P815, P816, P817, P818, P819, P820, P821, P822, P823, P824, P825, P826, P827, P828, P829, P830, P831, P832, P833, P834, P835, P836, P837, P838, P839, P840, P841, P842, P843, P844, P845, P846, P847, P848, P849, P850, P851, P852, P853, P854, P855, P856, P857, P858, P859, P860, P861, P862, P863, P864, P865, P866, P867, P868, P869, P870, P871, P872, P873, P874, P875, P876, P877, P878, P879, P880, P881, P882, P883, P884, P885, P886, P887, P888, P889, P890, P891, P892, P893, P894, P895, P896, P897, P898, P899, P900, P901, P902, P903, P904, P905, P906, P907, P908, P909, P910, P911, P912, P913, P914, P915, P916, P917, P918, P919, P920, P921, P922, P923, P924, P925, P926, P927, P928, P929, P930, P931, P932, P933, P934, P935, P936, P937, P938, P939, P940, P941, P942, P943, P944, P945, P946, P947, P948, P949, P950, P951, P952, P953, P954, P955, P956, P957, P958, P959, P960, P961, P962, P963, P964, P965, P966, P967, P968, P969, P970, P971, P972, P973, P974, P975, P976, P977, P978, P979, P980, P981, P982, P983, P984, P985, P986, P987, P988, P989, P990, P991, P992, P993, P994, P995, P996, P997, P998, P999, P1000, P1001, P1002, P1003, P1004, P1005, P1006, P1007, P1008, P1009, P1010, P1011, P1012, P1013, P1014, P1015, P1016, P1017, P1018, P1019, P1020, P1021, P1022, P1023, P1024, P1025, P1026, P1027, P1028, P1029, P1030, P1031, P1032, P1033, P1034, P1035, P1036, P1037, P1038, P1039, P1040, P1041, P1042, P1043, P1044, P1045, P1046, P1047, P1048, P1049, P1050, P1051, P1052, P1053, P1054, P1055, P1056, P1057, P1058, P1059, P1060, P1061, P1062, P1063, P1064, P1065, P1066, P1067, P1068, P1069, P1070, P1071, P1072, P1073, P1074, P1075, P1076, P1077, P1078, P1079, P1080, P1081, P1082, P1083, P1084, P1085, P1086, P1087, P1088, P1089, P1090, P1091, P1092, P1093, P1094, P1095, P1096, P1097, P1098, P1099, P1100, P1101, P1102, P1103, P1104, P1105, P1106, P1107, P1108, P1109, P1110, P1111, P1112, P1113, P1114, P1115, P1116, P1117, P1118, P1119, P1120, P1121, P1122, P1123, P1124, P1125, P1126, P1127, P1128, P1129, P1130, P1131, P1132, P1133, P1134, P1135, P1136, P1137, P1138, P1139, P1140, P1141, P1142, P1143, P1144, P1145, P1146, P1147, P1148, P1149, P1150, P1151, P1152, P1153, P1154, P1155, P1156, P1157, P1158, P1159, P1160, P1161, P1162, P1163, P1164, P1165, P1166, P1167, P1168, P1169, P1170, P1171, P1172, P1173, P1174, P1175, P1176, P1177, P1178, P1179, P1180, P1181, P1182, P1183, P1184, P1185, P1186, P1187, P1188, P1189, P1190, P1191, P1192, P1193, P1194, P1195, P1196, P1197, P1198, P1199, P1200, P1201, P1202, P1203, P1204, P1205, P1206, P1207, P1208, P1209, P1210, P1211, P1212, P1213, P1214, P1215, P1216, P1217, P1218, P1219, P1220, P1221, P1222, P1223, P1224, P1225, P1226, P1227, P1228, P1229, P1230, P1231, P1232, P1233, P1234, P1235, P1236, P1237, P1238, P1239, P1240, P1241, P1242, P1243, P1244, P1245, P1246, P1247, P1248, P1249, P1250, P1251, P1252, P1253, P1254, P1255, P1256, P1257, P1258, P1259, P1260, P1261, P1262, P1263, P1264, P1265, P1266, P1267, P1268, P1269, P1270, P1271, P1272, P1273, P1274, P1275, P1276, P1277, P1278, P1279, P1280, P1281, P1282, P1283, P1284, P1285, P1286, P1287, P1288, P1289, P1290, P1291, P1292, P1293, P1294, P1295, P1296, P1297, P1298, P1299, P1300, P1301, P1302, P1303, P1304, P1305, P1306, P1307, P1308, P1309, P1310, P1311, P1312, P1313, P1314, P1315, P1316, P1317, P1318, P1319, P1320, P1321, P1322, P1323, P1324, P1325, P1326, P1327, P1328, P1329, P1330, P1331, P1332, P1333, P1334, P1335, P1336, P1337, P1338, P1339, P1340, P1341, P1342, P1343, P1344, P1345, P1346, P1347, P1348, P1349, P1350, P1351, P1352, P1353, P1354, P1355, P1356, P1357, P1358, P1359, P1360, P1361, P1362, P1363, P1364, P1365, P1366, P1367, P1368, P1369, P1370, P1371, P1372, P1373, P1374, P1375, P1376, P1377, P1378, P1379, P1380, P1381, P1382, P1383, P1384, P1385, P1386, P1387, P1388, P1389, P1390, P1391, P1392, P1393, P1394, P1395, P1396, P1397, P1398, P1399, P1400, P1401, P1402, P1403, P1404, P1405, P1406, P1407, P1408, P1409, P1410, P1411, P1412, P1413, P1414, P1415, P1416, P1417, P1418, P1419, P1420, P1421, P1422, P1423, P1424, P1425, P1426, P1427, P1428, P1429, P1430, P1431, P1432, P1433, P1434, P1435, P1436, P1437, P1438, P1439, P1440, P1441, P1442, P1443, P1444, P1445, P1446, P1447, P1448, P1449, P1450, P1451, P1452, P1453, P1454, P1455, P1456, P1457, P1458, P1459, P1460, P1461, P1462, P1463, P1464, P1465, P1466, P1467, P1468, P1469, P1470, P1471, P1472, P1473, P1474, P1475, P1476, P1477, P1478, P1479, P1480, P1481, P1482, P1483, P1484, P1485, P1486, P1487, P1488, P1489, P1490, P1491, P1492, P1493, P1494, P1495, P1496, P1497, P1498, P1499, P1500, P1501, P1502, P1503, P1504, P1505, P1506, P1507, P1508, P1509, P1510, P1511, P1512, P1513, P1514, P1515, P1516, P1517, P1518, P1519, P1520, P1521, P1522, P1523, P1524, P1525, P1526, P1527, P1528, P1529, P1530, P1531, P1532, P1533, P1534, P1535, P1536, P1537, P1538, P1539, P1540, P1541, P1542, P1543, P1544, P1545, P1546, P1547, P1548, P1549, P1550, P1551, P1552, P1553, P1554, P1555, P1556, P1557, P1558, P1559, P1560, P1561, P1562, P1563, P1564, P1565, P1566, P1567, P1568, P1569, P1570, P1571, P1572, P1573, P1574, P1575, P1576, P1577, P1578, P1579, P1580, P1581, P1582, P1583, P1584, P1585, P1586, P1587, P1588, P1589, P1590, P1591, P1592, P1593, P1594, P1595, P1596, P1597, P1598, P1599, P1600, P1601, P1602, P1603, P1604, P1605, P1606, P1607, P1608, P1609, P1610, P1611, P1612, P1613, P1614, P1615, P1616, P1617, P1618, P1619, P1620, P1621, P1622, P1623, P1624, P1625, P1626, P1627, P1628, P1629, P1630, P1631, P1632, P1633, P1634, P1635, P1636, P1637, P1638, P1639, P1640, P1641, P1642, P1643, P1644, P1645, P1646, P1647, P1648, P1649, P1650, P1651, P1652, P1653, P1654, P1655, P1656, P1657, P1658, P1659, P1660, P1661, P1662, P1663, P1664, P1665, P1666, P1667, P1668, P1669, P1670, P1671, P1672, P1673, P1674, P1675, P1676, P1677, P1678, P1679, P1680, P1681, P1682, P1683, P1684, P1685, P1686, P1687, P1688, P1689, P1690, P1691, P1692, P1693, P1694, P1695, P1696, P1697, P1698, P1699, P1700, P1701, P1702, P1703, P1704, P1705, P1706, P1707, P1708, P1709, P1710, P1711, P1712, P1713, P1714, P1715, P1716, P1717, P1718, P1719, P1720, P1721, P1722, P1723, P1724, P1725, P1726, P1727, P1728, P1729, P1730, P1731, P1732, P1733, P1734, P1735, P1736, P1737, P1738, P1739, P1740, P1741, P1742, P1743, P1744, P1745, P1746, P1747, P1748, P1749, P1750, P1751, P1752, P1753, P1754, P1755, P1756, P1757, P1758, P1759, P1760, P1761, P1762, P1763, P1764, P1765, P1766, P1767, P1768, P1769, P1770, P1771, P1772, P1773, P1774, P1775, P1776, P1777, P1778, P1779, P1780, P1781, P1782, P1783, P1784, P1785, P1786, P1787, P1788, P1789, P1790, P1791, P1792, P1793, P1794, P1795, P1796, P1797, P1798, P1799, P1800, P1801, P1802, P1803, P1804, P1805, P1806, P1807, P1808, P1809, P1810, P1811, P1812, P1813, P1814, P1815, P1816, P1817, P1818, P1819, P1820, P1821, P1822, P1823, P1824, P1825, P1826, P1827, P1828, P1829, P1830, P1831, P1832, P1833, P1834, P1835, P1836, P1837, P1838, P1839, P1840, P1841, P1842, P1843, P1844, P1845, P1846, P1847, P1848, P1849, P1850, P1851, P1852, P1853, P1854, P1855, P1856, P1857, P1858, P1859, P1860, P1861, P1862, P1863, P1864, P1865, P1866, P1867, P1868, P1869, P1870, P1871, P1872, P1873, P1874, P1875, P1876, P1877, P1878, P1879, P1880, P1881, P1882, P1883, P1884, P1885, P1886, P1887, P1888, P1889, P1890, P1891, P1892, P1893, P1894, P1895, P1896, P1897, P1898, P1899, P1900, P1901, P1902, P1903, P1904, P1905, P1906, P1907, P1908, P1909, P1910, P1911, P1912, P1913, P1914, P1915, P1916, P1917, P1918, P1919, P1920, P1921, P1922, P1923, P1924, P1925, P1926, P1927, P1928, P1929, P1930, P1931, P1932, P1933, P1934, P1935, P1936, P1937, P1938, P1939, P1940, P1941, P1942, P1943, P1944, P1945, P1946, P1947, P1948, P1949, P1950, P1951, P1952, P1953, P1954, P1955, P1956, P1957, P1958, P1959, P1960, P1961, P1962, P1963, P1964, P1965, P1966, P1967, P1968, P1969, P1970, P1971, P1972, P1973, P1974, P1975, P1976, P1977, P1978, P1979, P1980, P1981, P1982, P1983, P1984, P1985, P1986, P1987, P1988, P1989, P1990, P1991, P1992, P1993, P1994, P1995, P1996, P1997, P1998, P1999, P2000, P2001, P2002, P2003, P2004, P2005, P2006, P2007, P2008, P2009, P2010, P2011, P2012, P2013, P2014, P2015, P2016, P2017, P2018, P2019, P2020, P2021, P2022, P2023, P2024, P2025, P2026, P2027, P2028, P2029, P2030, P2031, P2032, P2033, P2034, P2035, P2036, P2037, P2038, P2039, P2040, P2041, P2042, P2043, P2044, P2045, P2046, P2047, P2048, P2049, P2050, P2051, P2052, P2053, P2054, P2055, P2056, P2057, P2058, P2059, P2060, P2061, P2062, P2063, P2064, P2065, P2066, P2067, P2068, P2069, P2070, P2071, P2072, P2073, P2074, P2075, P2076, P2077, P2078, P2079, P2080, P2081, P2082, P2083, P2084, P2085, P2086, P2087, P2088, P2089, P2090, P2091, P2092, P2093, P2094, P2095, P2096, P2097, P2098, P2099, P2100, P2101, P2102, P2103, P2104, P2105, P2106, P2107, P2108, P2109, P2110, P2111, P2112, P2113, P2114, P2115, P2116, P2117, P2118, P2119, P2120, P2121, P2122, P2123, P2124, P2125, P2126, P2127, P2128, P2129, P2130, P2131, P2132, P2133, P2134, P2135, P2136, P2137, P2138, P2139, P2140, P2141, P2142, P2143, P2144, P2145, P2146, P2147, P2148, P2149, P2150, P2151, P2152, P2153, P2154, P2155, P2156, P2157, P2158, P2159, P2160, P2



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:**



**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:**

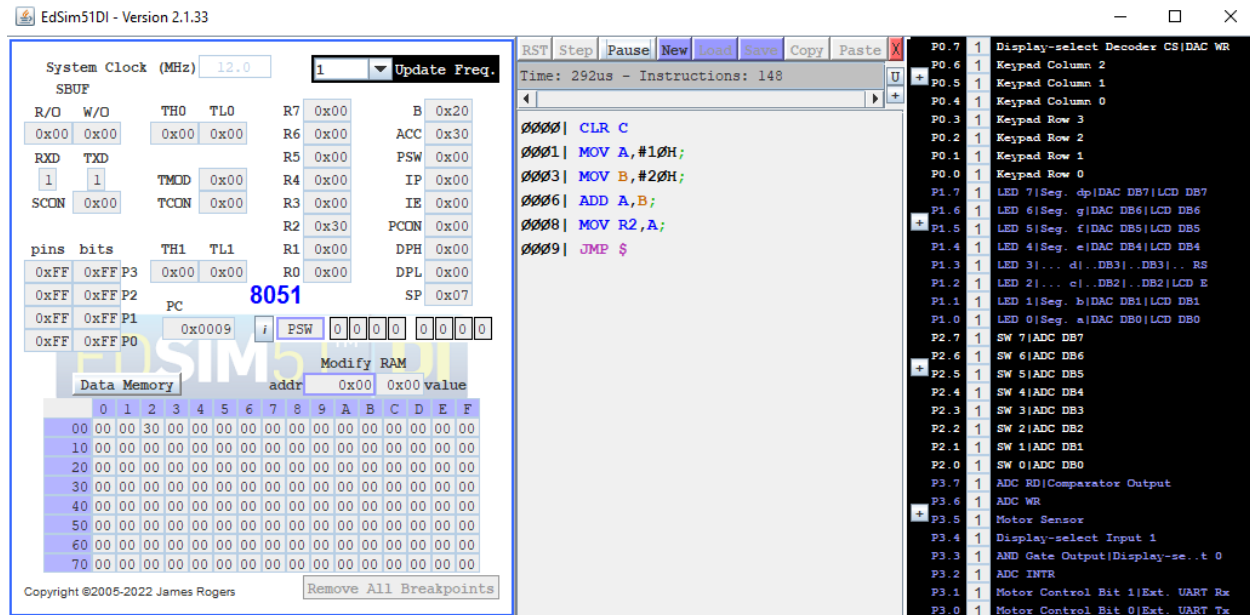


Fig addition

## 2.SUBTRACTION

ORG 0x0000

MOV A, #50H

MOV B, #30H

SUBB A, B

MOV R1,A

JMP \$

**Output:**

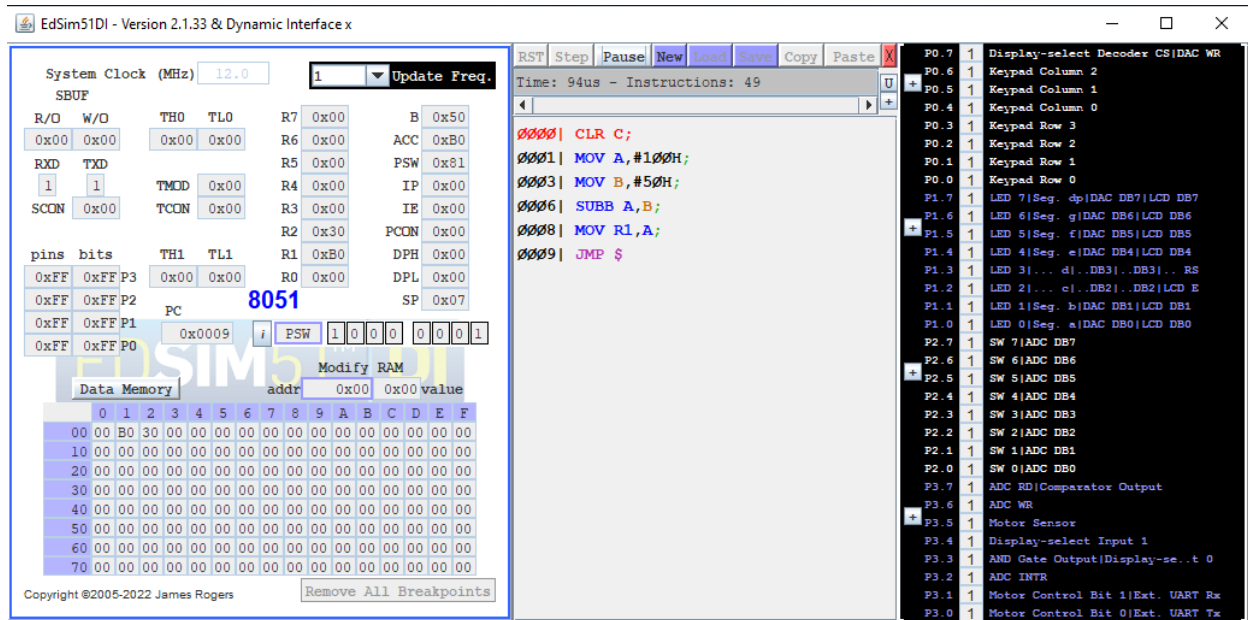


Fig subtraction

### 3.Multiplication

```

ORG 0X000

MOV A,#06H

MOV B,#03H

MUL AB

MOV R3,A

JMP $

```

Output:

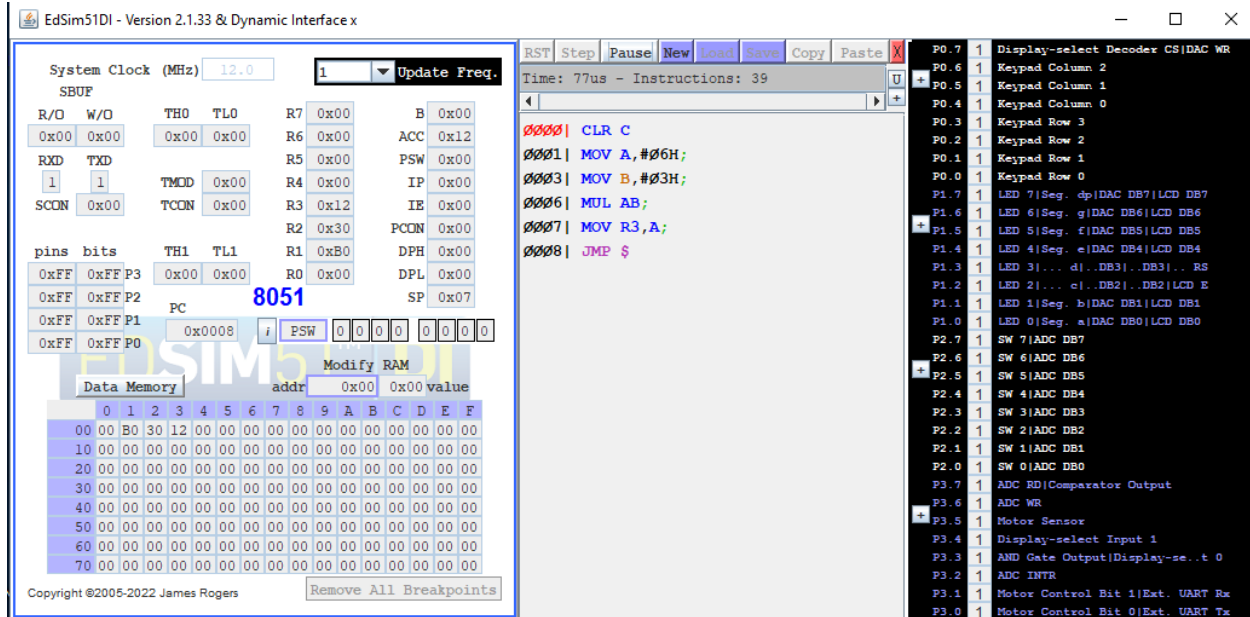


Fig multiplication

#### 4.Division

ORG 0X000

MOV A,#06H

MOV B,#03H

DIV AB

MOV R0,A

JMP \$

**OUTPUT:**

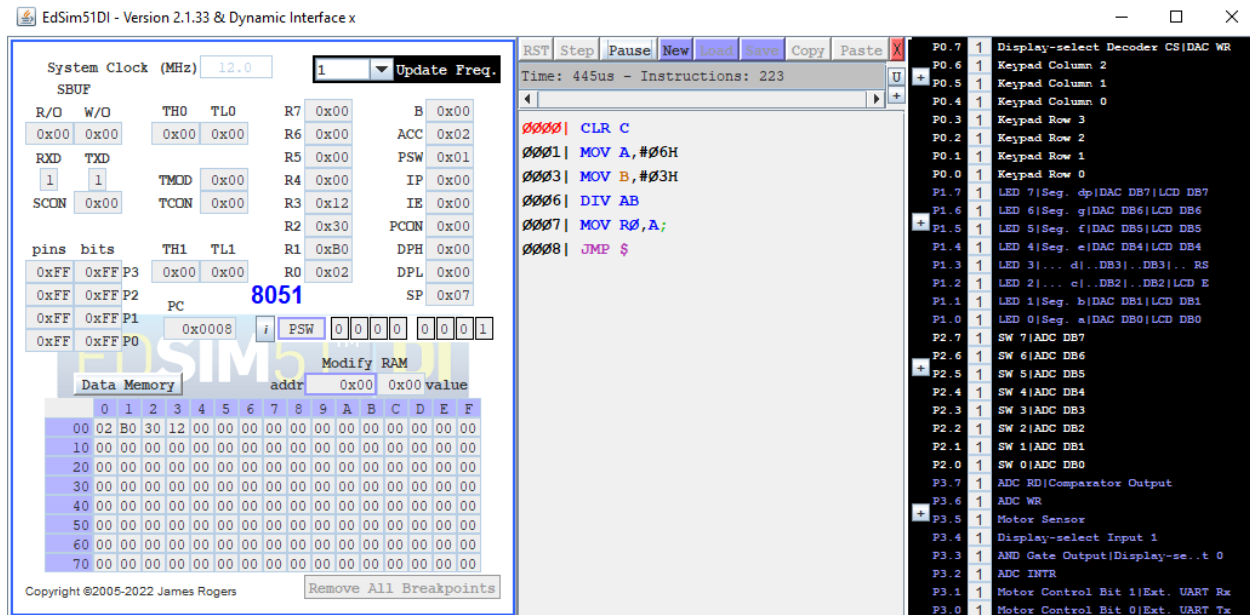


Fig Division

## RESULT:

### EXP . 4A . Interfacing of common anode 7- segment display to display decimal numbers 0 to 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

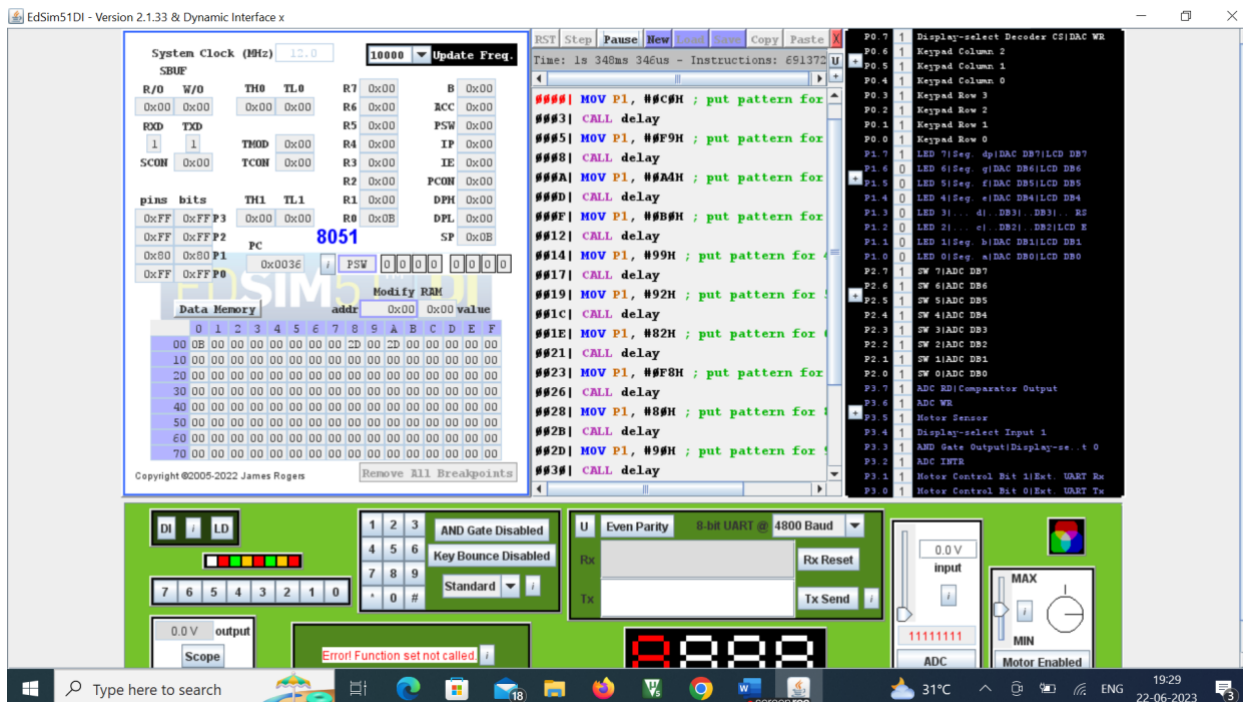
delay:

MOV R0, #10H

DJNZ R0, \$

RET

## OUTPUT:



## 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).

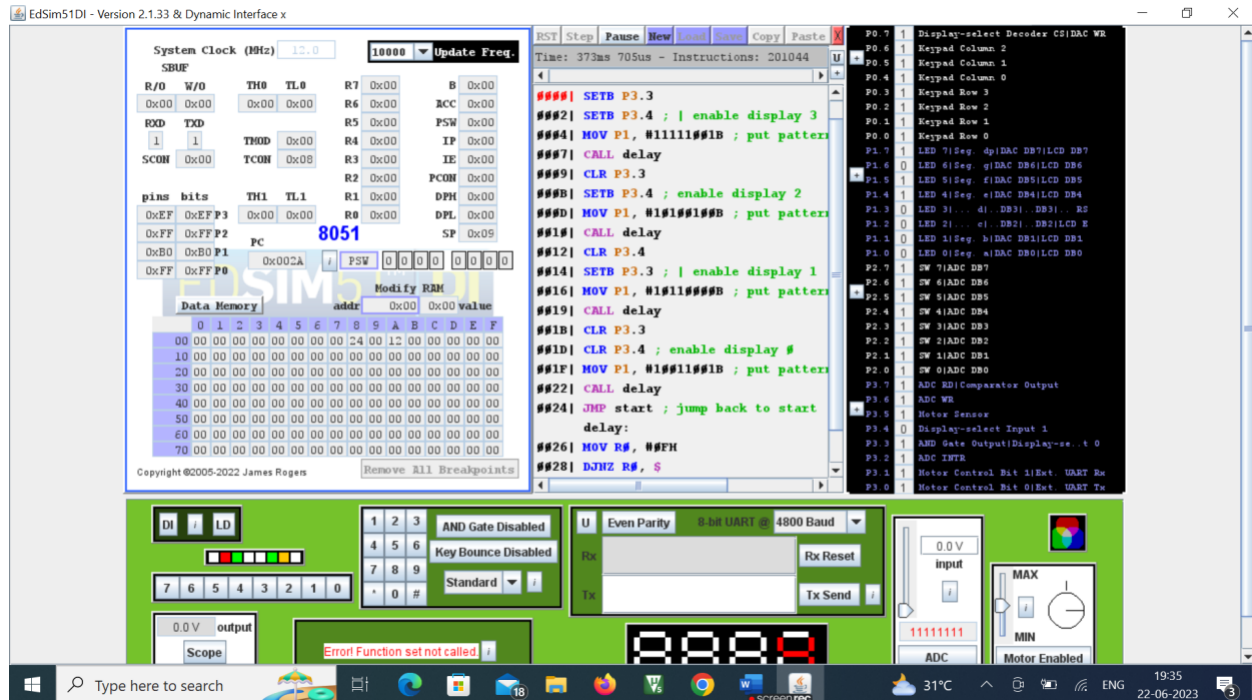
#### **CODE :**

```
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:**



**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

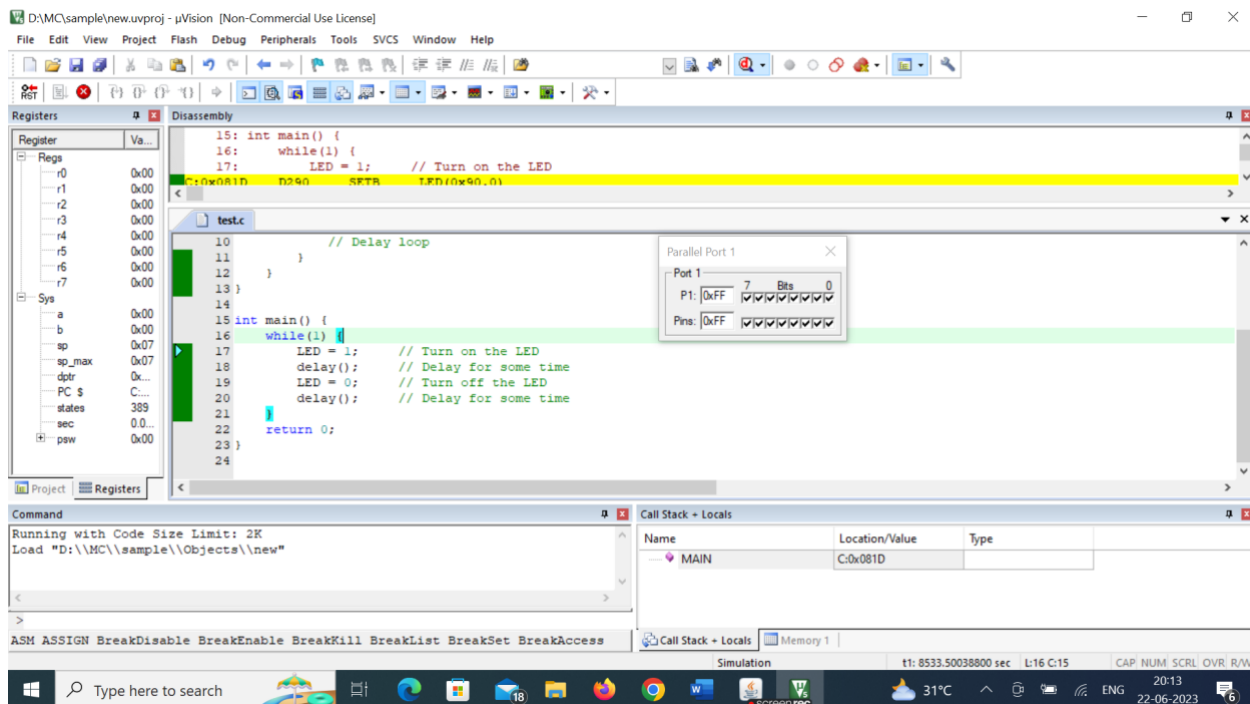
}

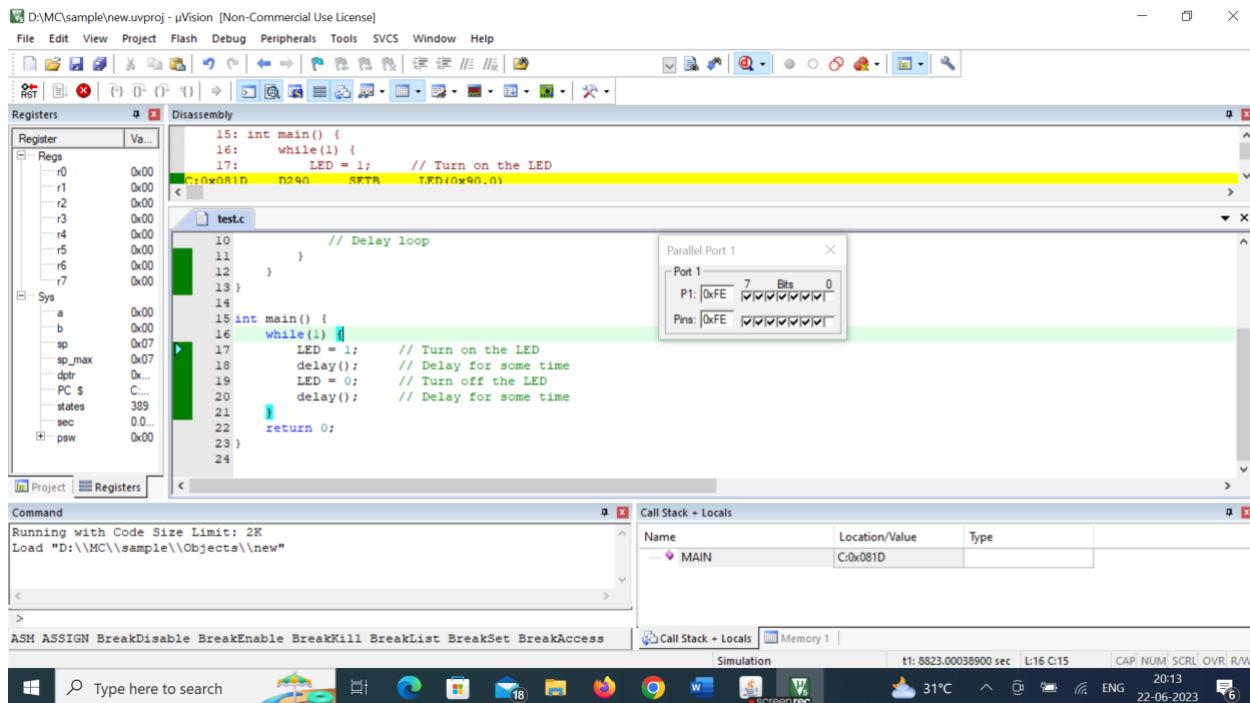
return 0;

}

```

## OUTPUT:





## 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 :**

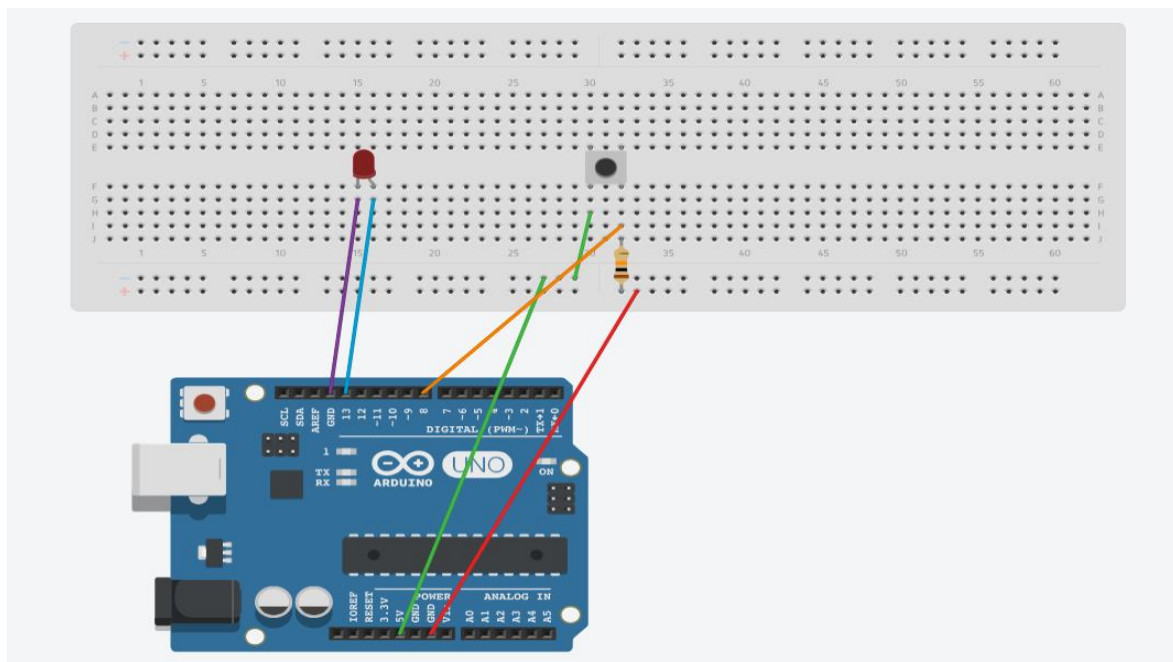
```
// Pin 13 has an LED connected on most Arduino boards.  
// give it a name:  
int ledPin = 13;  
int switchPin = 8;  
boolean lastButton= LOW;  
boolean currentButton= LOW;  
boolean ledOn= false;  
// the setup routine runs once when you press reset:  
void setup()  
{  
  // initialize the digital 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

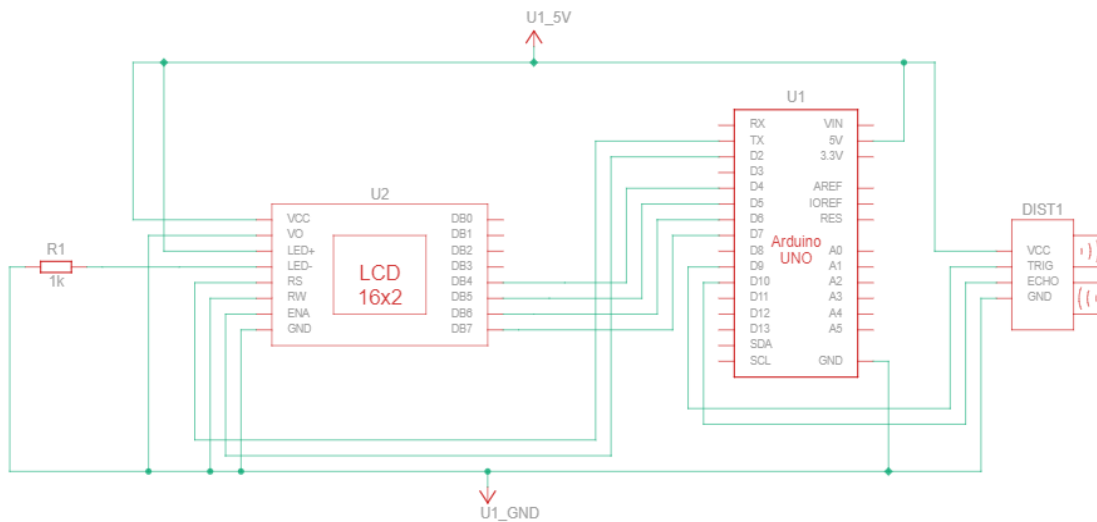
### AIM :

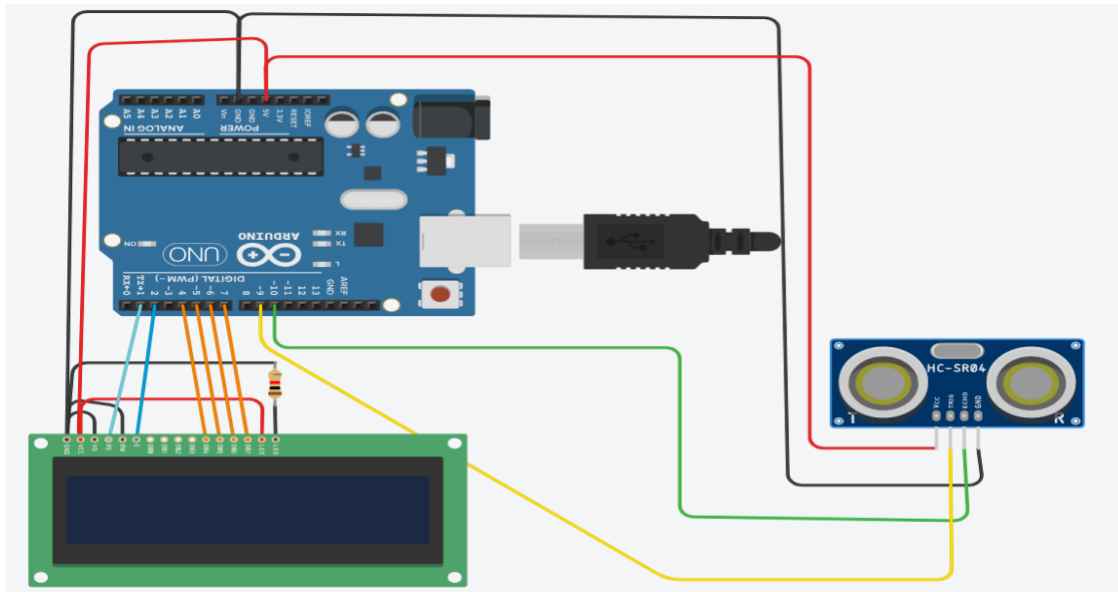
To measure the distance of an object using Ultrasonic Sensors and Arduino.

### Components Required:

Name	Quantity	Component
U1	1	Arduino Uno R3
U2	1	LCD 16 x 2
R1	1	1 k $\Omega$ 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);
}
```

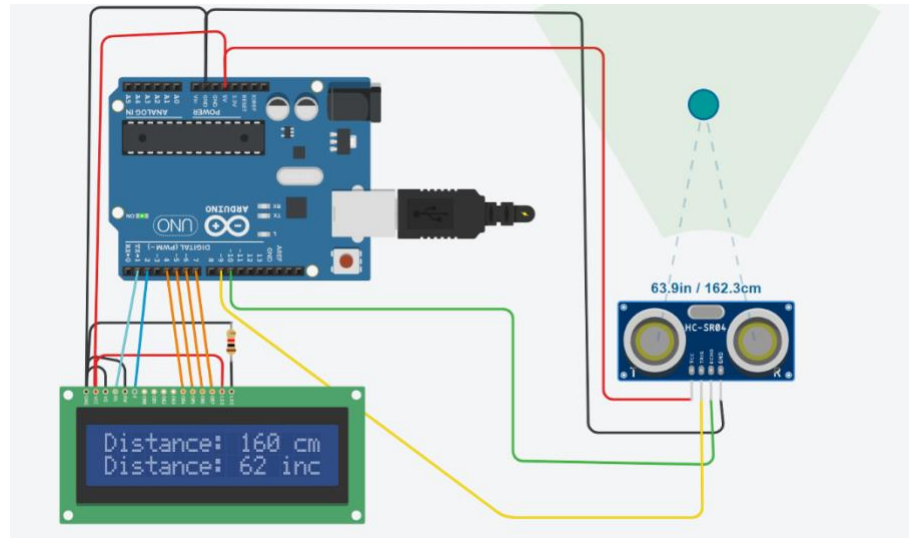


```

lcd.setCursor(0,1);
lcd.print("Distance: ");
lcd.print(distanceInch);
lcd.print(" inch");
delay(10);
}

```

## OUTPUT :



## 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.

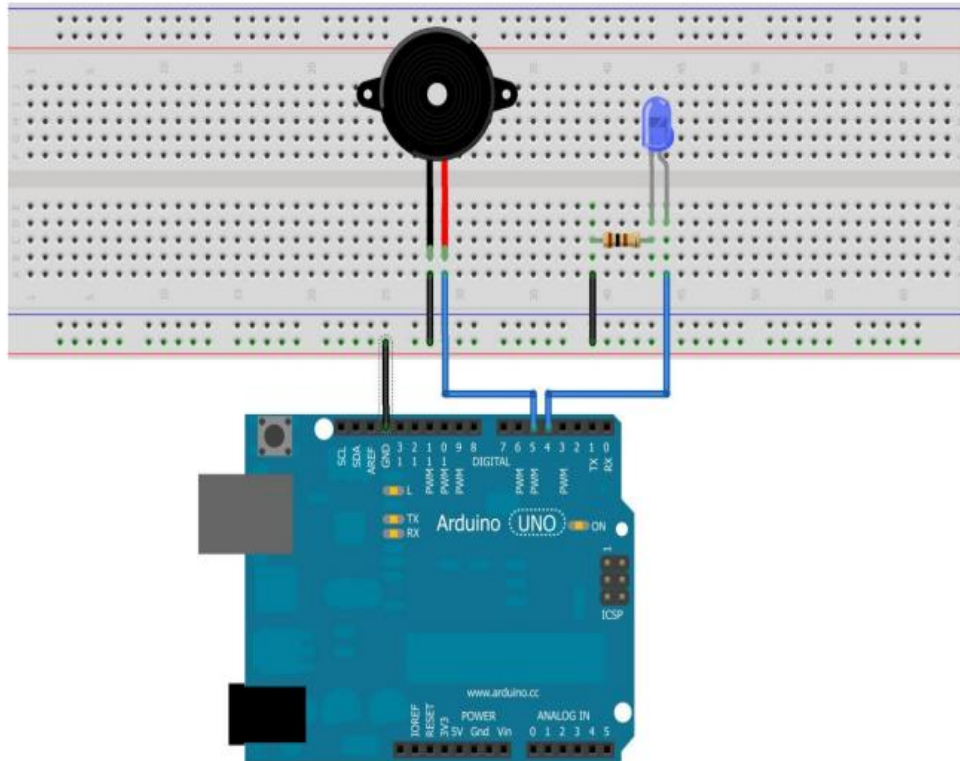
f. Wait for a specific duration (the interval between blinks and beeps).

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:**