4X4 CALCULATOR USING 8051, KEYPAD AND LCD

OBJECTIVE:

The calculator is built using AT89C51 microprocessor, 4X4 Keypad (SmallCalc), LM016L 16x2 LCD which will compute basic operations Addition, Subtraction, Division And Multiplication.

ABSTRACT:

The Calculator is virtually assembled in Proteus Simulation Software with AT89C51 8051 Microprocessor.

The Code is executed in a such way that when the user presses the key in the Simulation Software, the HEX file loaded into the microprocessor interacts with the input and it returns the output associated with it.

INTRODUCTION:

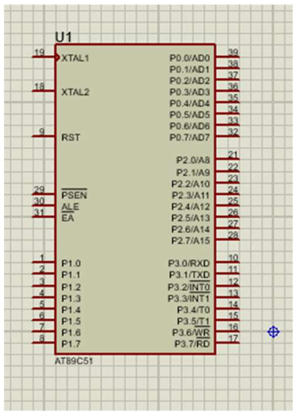
The Code Used here for this project is Embedded C, the code is executed and debugged in the Keil UVision Software . Then the output is generated in the HEX format and then it is loaded in the 8051 Processor.

SOFTWARE REQUIREMENTS:

AT89C51 microprocessor, 4X4 Keypad (SmallCalc), LM016L 16x2 LCD, Keil UVision, Proteus 8.1 Simulation Software.

CONCEPTS/WORKING PRINCIPLE

1. The **AT89C51** Processor is an age old 8-bit microcontroller from the Atmel family. It is a 40 pin IC package with 4Kb flash memory. It has four ports and all together provide 32 Programmable GPIO pins. It does not have in-built ADC module and supports only USART communication.
2. The **AT89C51** is no longer in production and Atmel does not support new design. Instead the new **AT89S51** is recommended for new applications.
3. AT89C51:



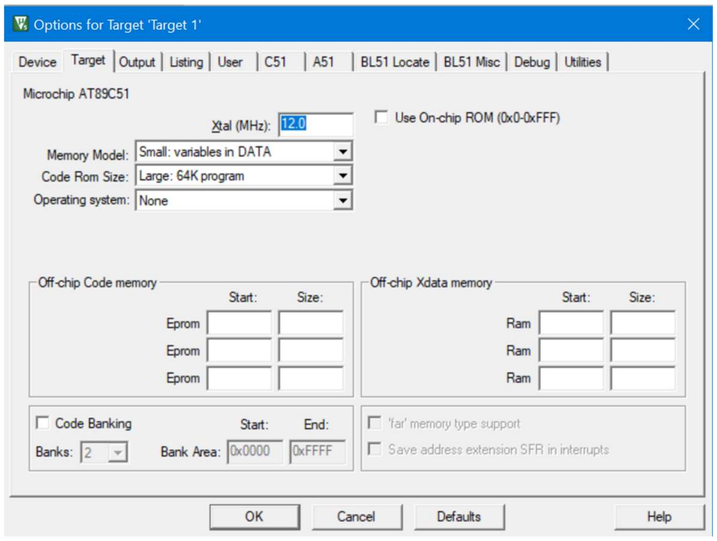
APPROACH/METHODOLOGY/PROGRAMS:

Embedded C Code :

```
#include <regx51.h>

void init();
void command(unsigned int);
void write_data(unsigned char);
void delay(unsigned char );
void msDelay(unsigned int);
char process_key(int,int,int,int,char,char,char,char);
void num_generator(char,int *);
void write_result(int,char);

sbit EN = P3^2; // Enable LCD
sbit RW = P3^3; // Read Write
sbit RS = P3^4; // Register Select

void main()
{
int i,j,* operand ,operand1=0,operand2=0,result=0;
int i_arr[4][4] = {{0,1,1,1},{1,0,1,1},{1,1,0,1},{1,1,1,0}};
char pressed_key,operator1='a',negative='N';
char c_arr[4][4] = {{'7','4','1','o'},{'8','5','2','0'},{'9','6','3','='},{'/','*','-
','+'}},error[10]={'M','A','T','H',' ','E','R','R','O','R'};
P1 = 0xff;
operand = &operand1;
while(1)
{
for(i=0;i<4;i++)
 {
pressed_key =
process_key(i_arr[i][0],i_arr[i][1],i_arr[i][2],i_arr[i][3],c_arr[i][0],c_arr[i][1],c_arr[i][2],c_arr[i][3]);
   if(pressed_key=='/'||pressed_key=='*'||pressed_key=='-'||pressed_key=='+')
   {
    operand = &operand2;
    operator1 = pressed_key;
   }
   if((pressed_key != 'Z')&&(pressed_key!= '=')&&(pressed_key!= '/')&&(pressed_key!=
'*')&&(pressed_key!= '-')&&(pressed_key!= '+'))
   {
    num_generator(pressed_key,operand);
   }
   // Make a function generate_result
   if(pressed_key == '=')
   {
    // Make a function generate_result in which all the mess below this line will be processed..
    if(operator1== '*')
    result = operand1*operand2;

    if(operator1== '/')
    if(operand2==0)
    {
     command(0xC0);
     for(j=0;j<10;j++)
     write_data(error[j]);
     msDelay(500);
     init();
    }
    else
     result = operand1/operand2;

    if(operator1== '-')
    {
    if (operand1>operand2)
    result = operand1-operand2;
    else
    {
    result = operand2-operand1;
    command(0xC0);
    negative = 'Y';
    }
    }

    if(operator1== '+')
    result = operand1+operand2;

    write_result(result,negative);
   }
  }
 }
}

// LCD initilaizer function
void init()
{
 delay(3500);
 command(0x38);
 delay(3500);
 command(0x38);
 delay(3500);
 command(0x38);
 delay(350);
 command(0x38);
 command(0x1C);
 command(0x0E);
 command(0x06);
 command(0x01);
 delay(3500);
 command(0x00);
 main();
}

// Sends different commands to LCD
void command(unsigned int comm)
{
 RW = 0;
 RS = 0;
 P2 = comm;
 EN = 1;
 delay(3500);
 EN = 0;
}

// Process and check the press of key
char process_key(int a,int b,int c,int d,char A,char B,char C,char D)
{
 char ch = 'Z';
 P1_0=a;
 P1_1=b;
 P1_2=c;
 P1_3=d;

 if(P1_4==0)
 {
  ch = A;
 }
 if(P1_5==0)
 {
  ch = B;
 }
 if(P1_6==0)
 {
  ch = C;
 }
 if(P1_7==0)
 {
  if (D == 'o')
  init();
  else
   ch = D;
 }
 if(ch!= 'Z')
 {
  write_data(ch);
  msDelay(100);
 }
 return ch;
}

// Used to write data in char on LCD
void write_data(unsigned char ch)
{
  RW = 0;
  RS = 1;
  P2 = ch;
  EN = 1;
  delay(3500);
  EN = 0;
}

void num_generator(char ch,int *operand)
{
int digit;
digit = ch - '0';
```

```
*operand = digit + (*operand*10);
}

void write_result(int num,char neg)
{
 int i=0,j,rem;
 char rev_num[20];
 command(0xC0);
 if(neg == 'Y')
 {
 write_data('-');
 }
 do
    {
    rem = num%10;
    num = num /10;
    rev_num[i] = (char)rem+'0';
    i++;
    }while(num>0);
  for(j=i-1;j>=0;j--)
    {
      RW = 0;
   RS = 1;
   P2 = rev_num[j];
   EN = 1;
   delay(3500);
   EN = 0;
    }
}
//Function for generation of delay
void delay(unsigned char c)
{
unsigned int i;
unsigned char j;
for(i=0;i<=3;i++)
{
for(j=0;j<=c;j++);
}
}
// Generates delay in milli seconds
void msDelay(unsigned int time)
{
 TL0 = 0xEF;
 TH0 = 0xAF;
 TR0 = 1;
 while(time--)
 {
  while(TF0 == 0);
  TF0 = 0;
  TL0 = 0xEF;
  TH0 = 0xAF;
 }
 TR0 = 0;
}
```

Target Files:


Options for Target 'Target 1'

Device | Target | Output | Listing | User | C51 | A51 | BL51 Locate | BL51 Misc | Debug | Utilities

Microchip AT89C51

Xtal (MHz): 12.0

☐ Use On-chip ROM (0x0-0xFFF)

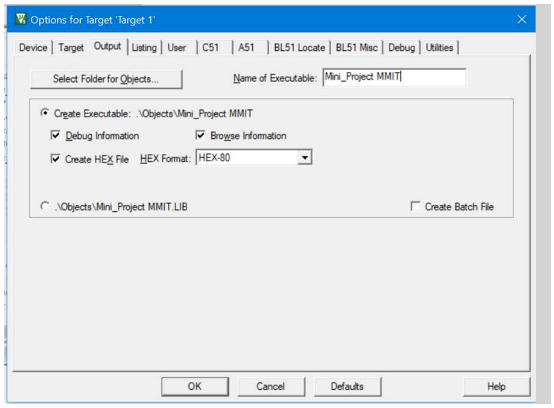
Memory Model: Small: variables in DATA
Code Rom Size: Large: 64K program
Operating system: None

Off-chip Code memory — Start: Size: — Eprom, Eprom, Eprom
Off-chip Xdata memory — Start: Size: — Ram, Ram, Ram

☐ Code Banking — Start: End:
Banks: 2 — Bank Area: 0x0000 0xFFFF
☐ 'far' memory type support
☐ Save address extension SFR in interrupts

OK | Cancel | Defaults | Help

Options for generating HEX file :


Options for Target 'Target 1'

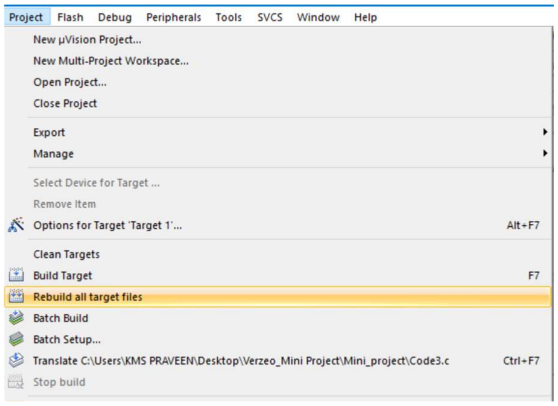
Device | Target | Output | Listing | User | C51 | A51 | BL51 Locate | BL51 Misc | Debug | Utilities

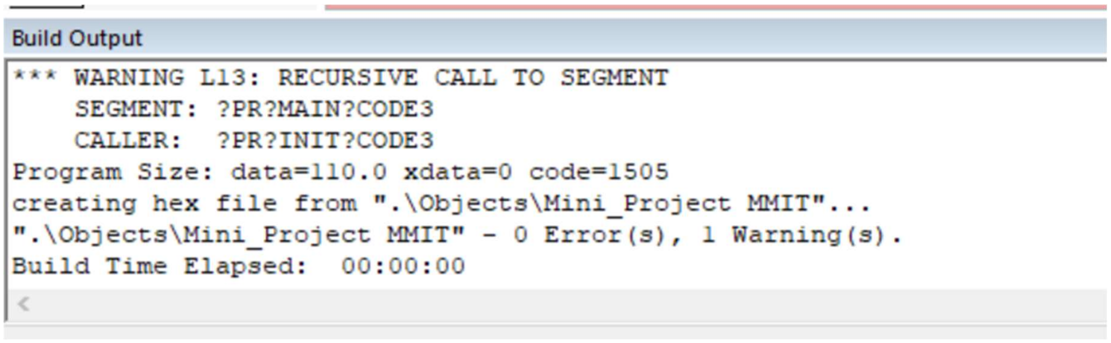
Select Folder for Objects... — Name of Executable: Mini_Project MMIT

◉ Create Executable: .\Objects\Mini_Project MMIT
☑ Debug Information ☑ Browse Information
☑ Create HEX File HEX Format: HEX-80

○ .\Objects\Mini_Project MMIT.LIB ☐ Create Batch File

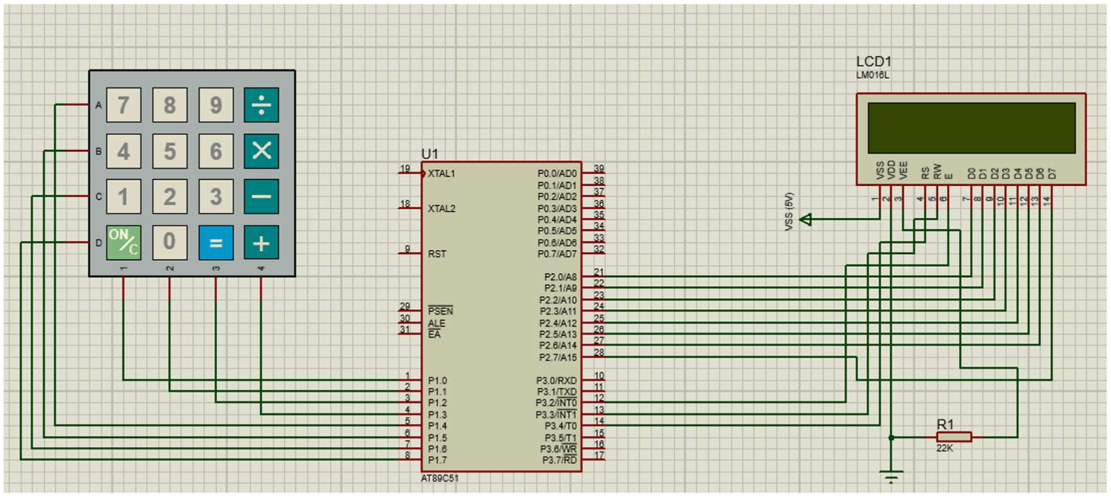
OK | Cancel | Defaults | Help


Project Flash Debug Peripherals Tools SVCS Window Help
New µVision Project...
New Multi-Project Workspace...
Open Project...
Close Project
Export
Manage
Select Device for Target ...
Remove Item
Options for Target 'Target 1'... Alt+F7
Clean Targets
Build Target F7
Rebuild all target files
Batch Build
Batch Setup...
Translate C:\Users\KMS PRAVEEN\Desktop\Verzeo_Mini Project\Mini_project\Code3.c Ctrl+F7
Stop build

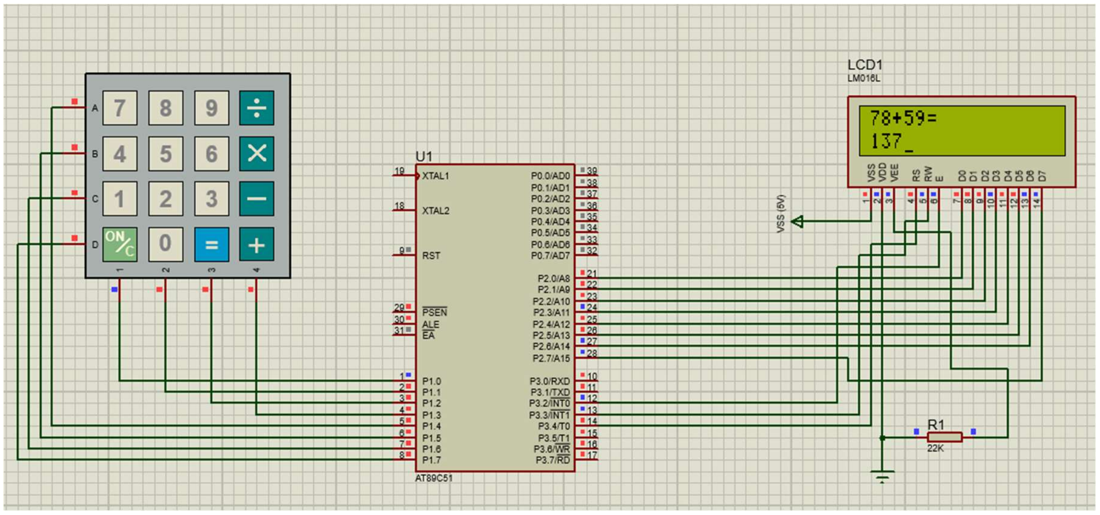

Build Output

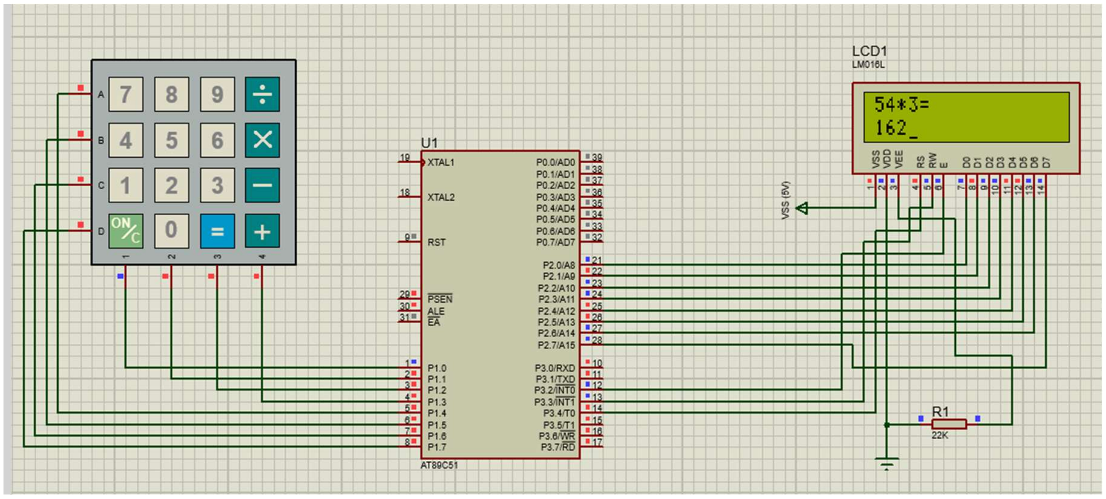
```
*** WARNING L13: RECURSIVE CALL TO SEGMENT
    SEGMENT: ?PR?MAIN?CODE3
    CALLER:  ?PR?INIT?CODE3
Program Size: data=110.0 xdata=0 code=1505
creating hex file from ".\Objects\Mini_Project MMIT"...
".\Objects\Mini_Project MMIT" - 0 Error(s), 1 Warning(s).
Build Time Elapsed:  00:00:00
```

Proteus Simulation Setup :



WORKING OUTPUTS:





CONCLUSIONS:
Thus the 8051 Calculator was virtually simulated using lcd , keypad , Keil and Proteus softwares.