Experiment no: 1

AIM: Write a 8051 C program to multiply two 16 bit binary numbers.

Design:

This is 16 bit multiplication program in embedded-C in 8051 micro controller with easiest design. Suppose we have two hex numbers num1 value 1234 & num2 value 5678 then multiplication of two numbers result will be 173C44.

num1 = 1234, num2 = 5678

num1 * num2 = result

Program:

```
#include<reg51.h>
void main(void)
{
        unsigned int num1,num2;
        unsigned long result;
        P0 = 0x00;
        P1 = 0x00;
        num1 = P0;
        num2 = P1;
        result = (unsigned long )num1 *  (unsigned long )num2;
}
```

Test Cases:

Enter value for num1 in port P0	P0 = 1234	
Enter value for num2 in port P1	P1 = 5678	Result = 17 3C44
Enter value for num1 in port P0	P0 = 4321	
Enter value for num2 in port P1	P1 = 8765	Result = 241 E79D

Experiment no: 2

Aim: Write a 8051 C program to find the sum of first 10 integer numbers.

Design:

In this program, we need to add first 10 integer numbers. In for loop, the variable i is initialized to 1, and the loop will continue as long as i is less than or equal to 10. In each iteration of the loop, the sum variable will add the value of i to itself, and the sum will be displayed in port1. Finally, the loop will increment the value of i by 1, and the process will repeat until the condition i <= 10 is no longer true.

```
int i, sum = 0
for(i = 1; i <= 10; i++)
sum = sum + i;
```

Program:

```
#include<reg51.h>
void main()
{
   int i, sum = 0;
   for(i = 1; i <= 10; i++)
   {
      sum = sum + i;
      P1 = sum;                                  //Store the sum in Port1
    }
}
```

Test Cases:

sum = 37

Experiment no: 3

Aim: Write a 8051 C program to find factorial of a given number

Design:

Factorial is used in many areas of mathematics but mainly used in permutation and combination. Factorial is the product of the all positive number from 1 to n (user entered number). In simple words, we can say that factorial of n would be 1*2*3*…..*n.

Note: There is no factorial exist for the negative number and the value of !0 is 1

int i, n, fact = 1

for(i = 1;i <= n; i++)

fact = fact * i

Program:

```
#include<reg51.h>

void main()

{

    int i, n, fact = 1;

    P0 = 0x00;   P1 = 0x00;

    n = P0;

        for (i = 1; i <= n ; i++)

        {

            fact = fact * i;
```

```
            P1 = fact;                    //Store the factorial in Port1
        }
}
```

Test Cases:

Enter value of n: n = 5	Enter value of n: n = 7
Fact = 78	Fact = 13B0

Experiment no: 4

Aim: Write a 8051 C program to add an array of 16 bit numbers and store the 32 bit result in internal RAM.

Design:

In this program, we need to add an array of 16 bit numbers. In for loop, the variable i is initialized to 0, and the loop will continue as long as i is less than 4. In each iteration of the loop, the sum variable will add the value of numbers from array. Finally, the loop will increment the value of i by 1, and the process will repeat until the condition i < 4 is no longer true.

unsigned int numbers[] = {0x1234, 0x5678, 0x9ABC, 0xDEFF}

unsigned long sum = 0

for(i = 0; i<4; i++)

sum += numbers[i]

((unsigned long)RESULT_ADDR) = sum

Program:

```
#include<reg51.h>
#define RESULT_ADDR 0x30

void main()
{
   int i;
   unsigned int numbers[] = {0x1234, 0x5678, 0x9ABC, 0xDEFF};
   unsigned long sum = 0;

         for(i = 0; i<4; i++)
         {
               sum += numbers[i];
         }

        *((unsigned long*)RESULT_ADDR) = sum;
}
```

Test Cases: Sum = 1235 D033

Experiment no: 5

Aim: Write a 8051 C program to find the square of a number (1 to 10) using look-up table.

Design:

Finding the Square of a Number is a simple method. We need to multiply the given number by itself to find its square number. The square term is always represented by a number raised to the power of 2. For example, the square of 6 is 6 multiplied by 6, i.e., 6×6 = 62 = 36. The following program uses look up table concept. Instead of finding the squares by arithmetic operations, the calculated values of squares of 0 to 9 are stored in memory locations. These memory locations are accessed using their addresses, the last digits of which are from zero to nine.

```
const unsigned char lookup[] = {1, 4, 9, 16, 25, 36, 49, 64, 81, 100}
if (num  >= 1 && num  <= 10)

square = lookup[num - 1]
```

Program:

```
#include <reg51.h>
const unsigned char lookup[] = {1, 4, 9, 16, 25, 36, 49, 64, 81, 100};
void main()
{
        unsigned char num = 5;
        unsigned char square = 0;
        P1 = 0x00;
```

```
    P2 = 0x00;
    num = P1;                                   // write the number to port 1
    if (num  >= 1 && num  <= 10)
    {
        square = lookup[num - 1];
    }
    P2 = square;                                // write the square to port 2
}
```

Test Cases:

Enter value of num: num = 5	Enter value of num: num = 9
Square = 25	Square = 81

Experiment no: 6

Aim: Write a 8051 C program to find the largest/smallest number in an array of 32 numbers

Design:

This embedded-C program shows how to find the largest and the smallest number from within an array. Inside the main (), the integer type array is declared and initialized. The integer type array is used to store consecutive values all of them having type integer.

The statement is: int numbers[] = {};

Then two integer type variable, name smallest and largest are declared and initialized with the 0th index value of the array. Then a 'for loop' is used which goes from 1 to the array length. Within this loop the largest and the smallest value is detected and

initialized to the smallest and largest value using if()

When …. numbers[i] is greater than largetst

largetst = numbers[i];

when numbers[i] greater than smallest

smallest = numbers[i];

unsigned int i

unsigned int numbers[array_size] = {23, 54, 6, 78, 89, 12, 45, 67, 9, 11, 3, 1, 32,

54, 76, 89, 12, 65, 43, 21, 98, 76, 54, 32, 10, 43, 65, 87, 99, 65, 21, 43}

for (i = 1; i < array_size; i++)

if (numbers[i] > largest)

largest = numbers[i]

if (numbers[i] < smallest)

smallest = numbers[i]

Program:

```
#include <reg51.h>
#define array_size 32
void main()
{
   unsigned int i;
   unsigned int numbers[array_size] = {23, 54, 6, 78, 89, 12, 45, 67, 9, 11, 3, 1,
   32, 54, 76, 89,12, 65, 43, 21, 98, 76, 54, 32, 10, 43, 65, 87, 99, 65, 21, 43};
Unsigned int largest = numbers[0];
   unsigned int smallest = numbers[0];
        for (i = 1; i < array_size; i++)
        {
              if (numbers[i] > largest)
              {
               largest = numbers[i];
              }
              if (numbers[i] < smallest)
              {
              smallest = numbers[i];
              }
        }
        P1 = largest;
        P0 = smallest;
}
```

Test cases:

Largest number from port1 = 99

Smallest number from port1 = 1

Experiment no: 7

Aim: Write a 8051 C program to arrange a series of 32 bit numbers in ascending/descending order

Design:

Arranging numbers in ascending order and descending order. We know, while arranging numbers from the smallest number to the largest number, then the numbers are arranged in ascending order. Suppose for example, 81, 97, 123, 137 and 201 are arranged in ascending order. Vice-versa while arranging numbers from the largest number to the smallest number then the numbers are arranged in descending order.

Suppose for example, 187, 121, 117, 103 and 99 are arranged in descending order.

numbers[NUM_NUMBERS]={0xDEADBEEF,0xCAFEBABE,0xBAADF00D, 0xFEEDFAC}

bubble_sort(numbers, NUM_NUMBERS)

void bubble_sort(unsigned long* numbers, int num_numbers)

int i, int j

for(i = 0; i<num_numbers - 1; i++)

for(j = 0; j<num_numbers - i - 1; j++)

if (numbers[j] > numbers[j+1])

swap(&numbers[j], &numbers[j+1])

Program:

```
#include<reg51.h>
#define NUM_NUMBERS 4
#define NUMBERS_ADDR 0x20
void swap(unsigned long* x, unsigned long* y)
{
        unsigned long temp = *x;
```

```
        *x = *y;
        *y = temp;
}
void bubble_sort(unsigned long* numbers, int num_numbers)
{
        int i;
        int j;
        for(i = 0; i < num_numbers - 1; i++)
        {
                for(j = 0; j < num_numbers - i - 1; j++)
                {
                        if (numbers[j] > numbers[j+1])
                        {
                        swap(&numbers[j],    &numbers[j+1]);
                        }
                }
        }
}
void main()
{
        unsigned long numbers[NUM_NUMBERS] = {0xDEADBEEF,
        0xCAFEBABE, 0xBAADF00D,0xFEEDFACE};
        bubble_sort(numbers, NUM_NUMBERS);
        while(1);
}
```

Test cases:

Enter numbers: 0xDEADBEEF, 0xCAFEBABE, 0xBAADF00D, 0xFEEDFACE
Sorted numbers: 0xBAADF00D, 0xCAFEBABE, 0xDEADBEEF, 0xFEEDFACE

Experiment no: 8

Aim: Write a 8051 C program to count the number of ones and zeros in two consecutive memory locations.

Design:

In this program, first memory location value taken in value1 variable. value1 bits are performed logical AND operation bitwise 8 times by using shift operation and counter1 or counter0 is incremented based on logical condition true or false.

```
unsigned char* ptr = 0x1000;
unsigned char value1 = *ptr;
unsigned char value2 = *(ptr+1);

if (value1 & (1 << i))          if (value2 & (1 << i))
count1++;                       count1++;
else                            else
Count0++;                       Count0++;
                                                        } for (i = 0; i < 8; i++)
```

Program:

```
#include <reg51.h>
void main()
{
   unsigned char i, count1, count0;
   unsigned char* ptr = 0x1000;                 // Starting memory location
   unsigned char value1 = *ptr;                  // First memory location
   unsigned char value2 = *(ptr+1);             // Second memory location
   count1 = count0 = 0;

        for (i = 0; i < 8; i++)
            {
                if (value1 & (1 << i))
```

```
                {
                        count1++;
                }
                else
                {
                        Count0++;
            }
 }
   P1 = count1;
   P1 = count0;
   count1 = count0 = 0;

        for (i = 0; i < 8; i++)
            {
                if (value2 & (1 << i))
                {
                        count1++;
                 }
                else
                {
                        count0++;
                 }
            }
   P2 = count1;
   P2 = count0;
}
```

Test cases:

Enter the value for value1: value1 = 3	Enter the value for value1: value2 =
Count1 = 2	4
Count0 = 6	Count1 = 1
	Count0 = 7

Experiment no: 9

Aim: Write an 8051 C program to scan a series of 32 bit numbers to find how many are negative.

Design:

In the below program, to find how many negative numbers in a given array, first the number is taken from an array and then logical AND operation with 80000000 using if statements. Repeat same procedure for remaining numbers in an array.

unsigned long numbers[NUM_NUMBERS] = {0xDEADBEEF, 0xCAFEBABE, 0xBAADF00D, 0xFEEDFACE}

int negative_count = 0

for (int i = 0; i<NUM_NUMBERS; i++)

if (numbers[i] & 0x80000000)

negative_count++

Program:

```
#include<reg51.h>


#define NUM_NUMBERS 4

#define NUMBERS_ADDR 0x20

void main()

{
```

```
    int i;

    unsigned long numbers[NUM_NUMBERS] = {0xDEADBEEF,
    0x0AFEBABE, 0xBAADF00D,0xFEEDFACE};
    int negative_count = 0;
    for(i = 0; i < NUM_NUMBERS; i++)
    {
        if (numbers[i] & 0x80000000)
            {
                negative_count++;
            }
    }
    while(1);
}
```

Test Cases:

Enter numbers: 0xDEADBEEF, 0x0AFEBABE, 0xBAADF00D, 0xFEEDFACE

Negative count: 3

Enter numbers: 0x987654321, 0xBCDEFBCD, 0x36781234, 0x56472348

Negative count: 2

Experiment no: 10

Aim: Write a 8051 C program to display “Hello World” message.

Program:

```
#include<reg51.h>
void SEND (unsigned char);
void main()
{
TMOD = 0x20;
TH1 = 0xFD;
SCON = 0x50;
TR1 = 1;
SEND('H');
SEND('E');
SEND('L');
SEND('L');
SEND('O');
SEND('W');
SEND('O');
SEND('R');
SEND('L');
SEND('D');
}
void send(unsigned char x)
        {
            SBUF = x;
             while(TI == 0);
              TI = 0;
          }
```

Experiment no: 11

Aim: Write a 8051 C program to convert the hexadecimal data 0xCFh to decimal and display the digits on ports P0, P1 and P2 (port window in simulator).

Design:

In this program, the hexadecimal number is taken in x variable. x is divided by 10 using % modulus, remainder is saved in d0 variable, again x is divided by 10 using / integer division and save the quotient in x variable only. Quotient value x is divided by 10 using modulus, remainder is saved in d1 variable. x is divided by 10 using / integer division and save the quotient in d2. The decimal number will be available in d0, d1 and d2 variable.

P0 = input, P1, P2, P3 = output

x = input

d0 = x % 10;

x = x / 10;

d1 = x % 10;

d2 = x / 10;

LSB = d0;

MB = d1;

MSB = d2;

Program:

```
#include<reg51.h>

#define input P0

#define LSB P1

#define MB P2
```

```
#define MSB P3
unsigned char x,d0,d1,d2;
void main(void)
{
        input  =  0xff;
        LSB  =  0x00;
        MB  =  0x00;
        MSB = 0x00;
        while (1)
        {
                x = input;
                d0 = x % 10;
                x = x / 10;
                d1 = x % 10;
                d2 = x / 10;
                LSB = d0;
                MB = d1;
                MSB = d2;
        }
}
```

Test cases:

Enter Hexa number: FF	Enter Hexa number: 0E
Decimal equivalent is: 255	Decimal equivalent is: 014

Content beyond syllabus

Experiment No: 12

Aim: Write an 8051 C program is used to swapping two numbers, using bitwise operators.

Program:

In this program, value of i and k taken through port P0 and P1, then swapped two numbers using bitwise XOR operation.

```
#include <reg51.h>

int main()

    {

        int i;

        int k;

        i = P0;

        k = P1;

        i = i ^ k;

        k = i ^ k;

        i = i ^ k;

        P0 = i;

        P1 = k;

    }
```

Test Cases:

Enter value of i and k through port P0 and P1.

Swaping of numbers will be in P0 and P1.

Experiment No: 13

Aim: Write an 8051 C program to toggle bits of P1 ports continuously with a 250ms.

Program:

```
#include <reg51.h>

void MSDelay(unsigned int);
void main( )
{
	while (1)
	{
		P1= 0x55;
		MSDelay(250);

		P1 = 0xAA;
		MSDelay(250);
	}
}

void MSDelay(unsigned int itime)
{
	unsigned int i, j;
	for (i = 0; i < itime; i++)
	for (j = 0; j < 1275; j++);
}
```

Embedded C Basics Lab VIVA-Questions

1) What is Microcontroller?
2) Explain the differences between microprocessor & microcontroller.
3) What are logical operators?
4) What is an Embedded System?
5) List the application of embedded systems.
6) Explain the differences between C & Embedded-C
7) Explain the characteristics of Embedded programming environment.
8) Explain the usage of reg51.h
9) Explain the Bitwise &(AND) operation with an example.
10) Explain the Bitwise |(OR) operation with an example.
11) Explain the Bitwise ^(XOR) operation with an example.
12) Explain the bitwise left shift operation with an example.
13) Explain the bitwise right shift operation with an example.
14) Explain the syntax of for loop structure.
15) Write the program for the infinite loop demonstration.

APPENDIX

PREAMBLE

This laboratory manual contains the details of the laboratory experiment as per the curriculum of B.E under VTU. The laboratory manual helps the student to understand the aim and then procedure. Further the student will also come to know the application of this laboratory in future endeavoring electronics engineering projects.

The Embedded-C basics Engineering Laboratory helps the student to understand the basic concepts of c and embedded C. This laboratory manual also contains the sample viva voce questions and sample external experiments which will be asked frequently during the regular labs. Further the information regarding the experiments to be incorporated in the syllabus is also mentioned.

Earlier, many embedded applications were developed using assembly level programming. However, they did not provide portability. This disadvantage was overcome by the advent of various high-level languages like C, Pascal, and COBOL. However, it was the C language that got extensive acceptance for embedded systems, and it continues to do so. The C code written is more reliable, scalable, and portable; and in fact, much easier to understand. Embedded C Programming is the soul of the processor functioning inside each and every embedded system we come across in our daily life, such as mobile phones, washing machines, and digital cameras. Each processor is associated with embedded software. The first and foremost thing is the embedded software that decides to function of the embedded system. Embedded C language is most frequently used to program the microcontroller.

Objectives:

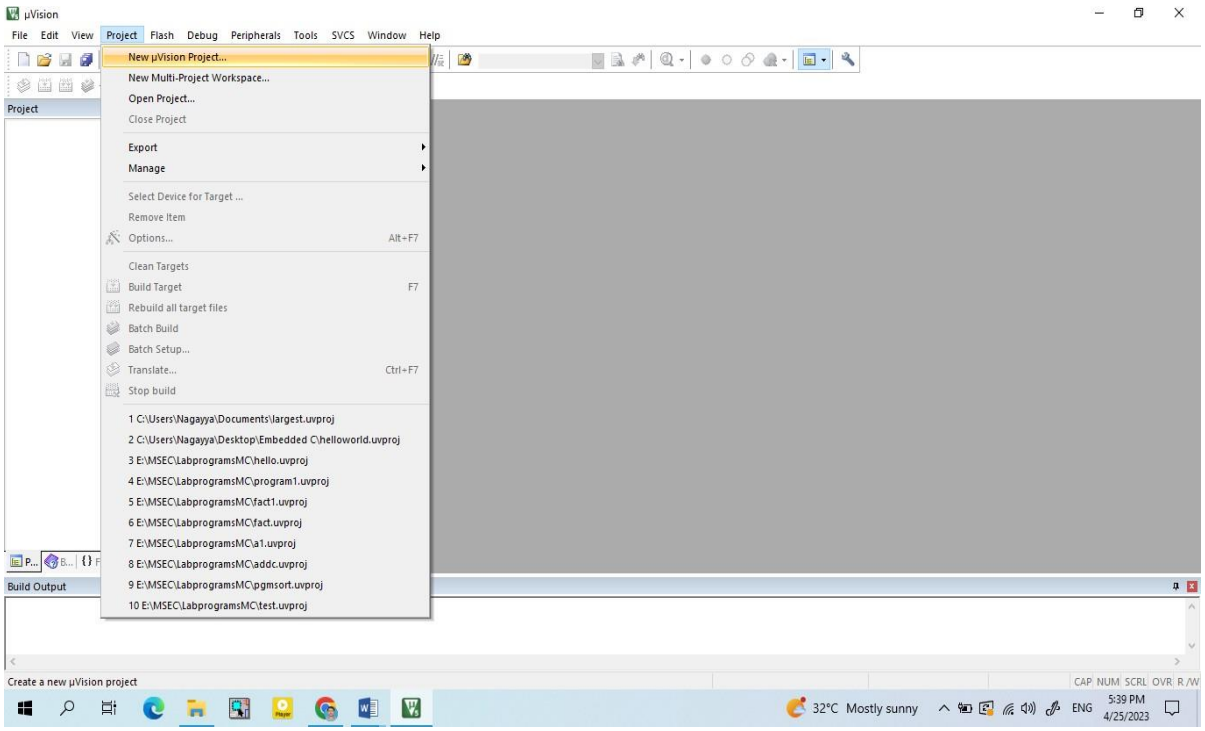
- ✓ Understand the basic programming of Embedded C using 8051.
- ✓ To develop the microcontroller-based programs for various applications.

Execution steps:

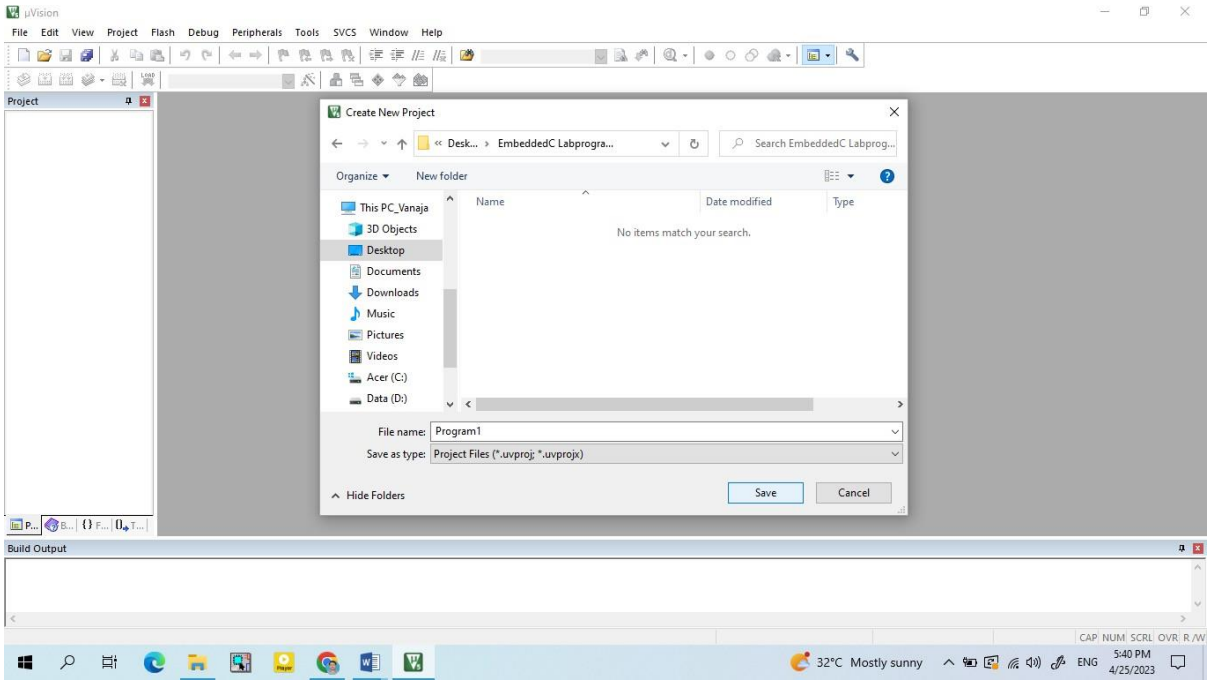
Procedure for Keil project to generate .C File for AT89C51

Figure No. 1: Snap shot of keil setup to generate .C file.

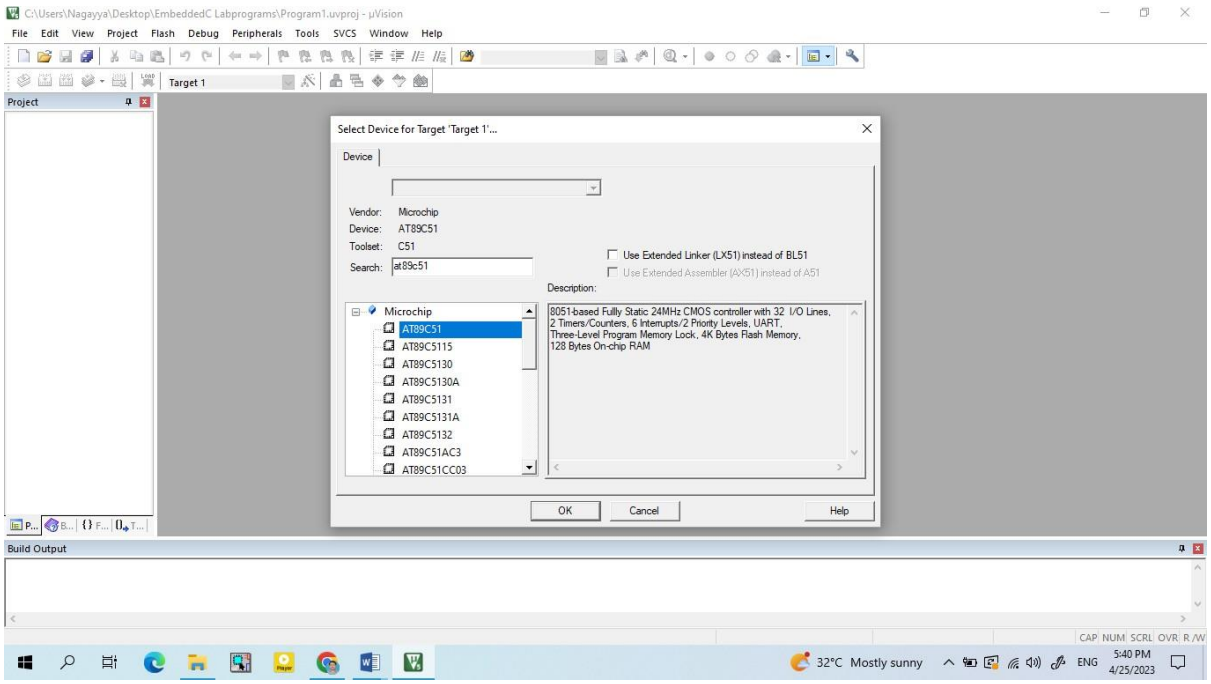
Step1: Open the Keil software and select the New Microvision project from Project Menu as shown below.



Step2: Browse to your project folder and provide the project name and click on save.

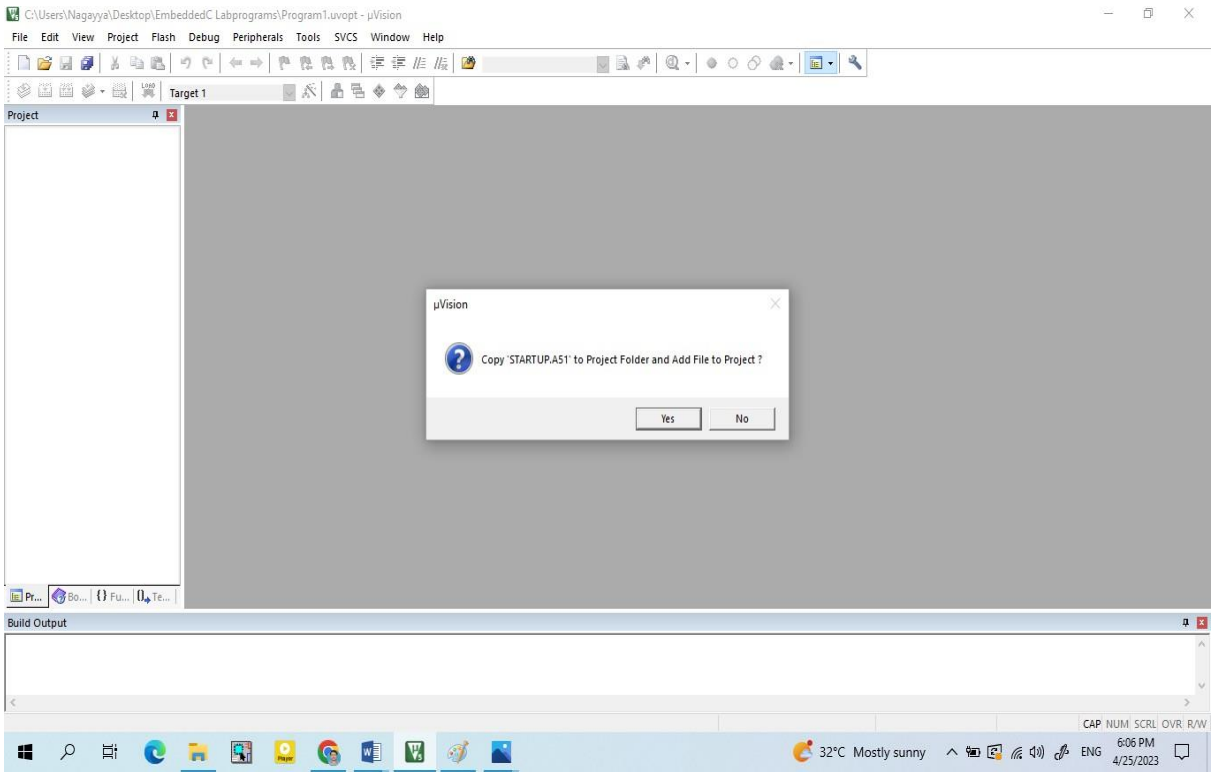


Step3: Once the project is saved a new pop up "Select Device for Target" opens, Select the controller (AT89C51) and click on OK.

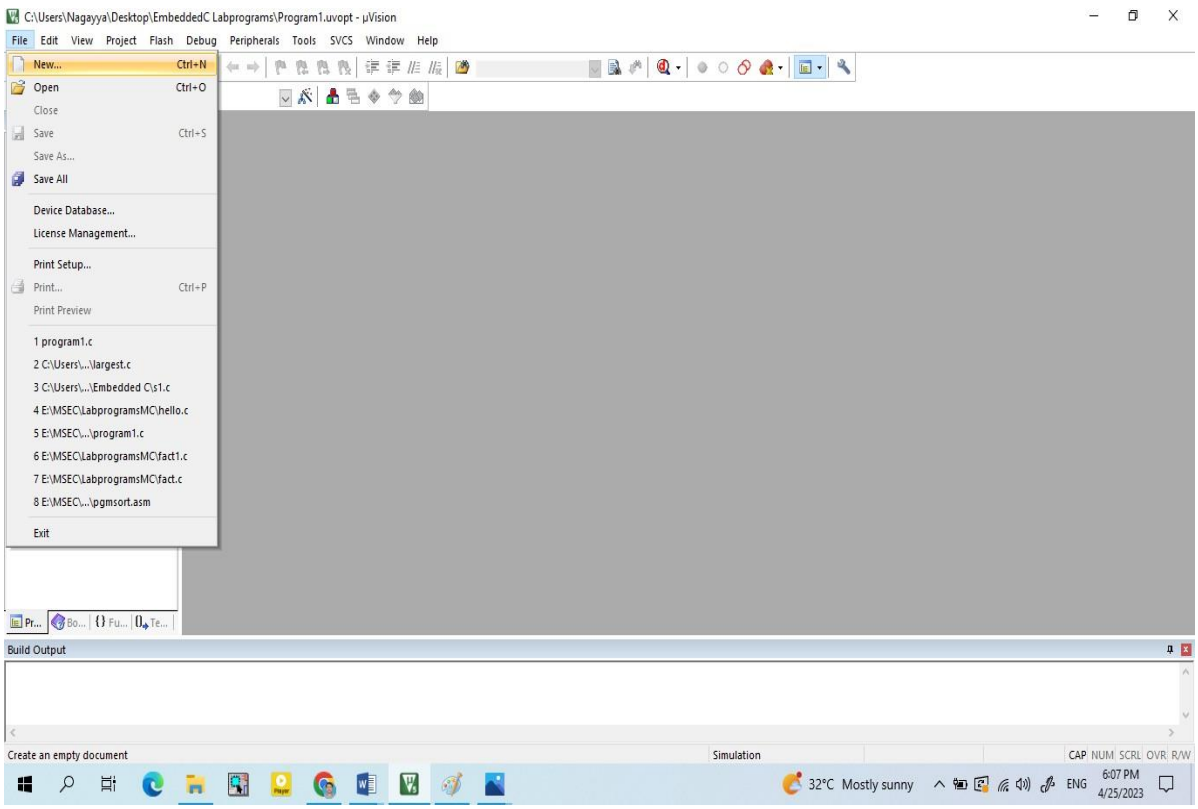


Step4: Select the controller (AT89C51) and click on OK.

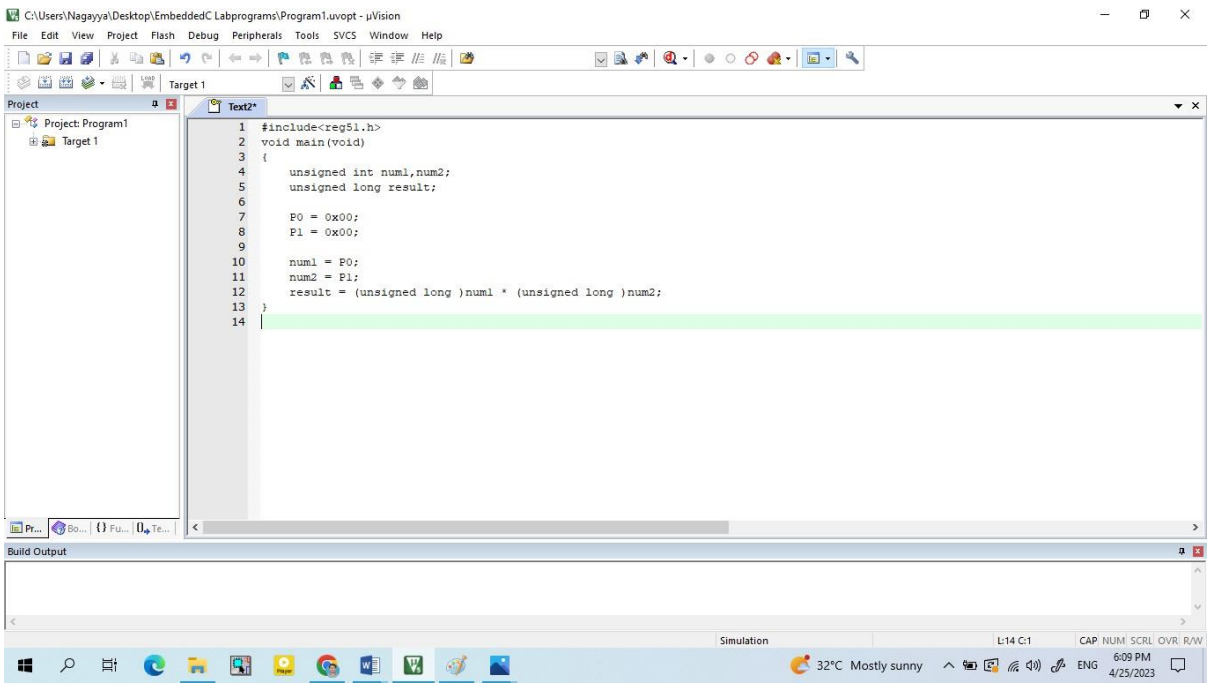
Step5: As AT89C51 needs the startup code click on **No**



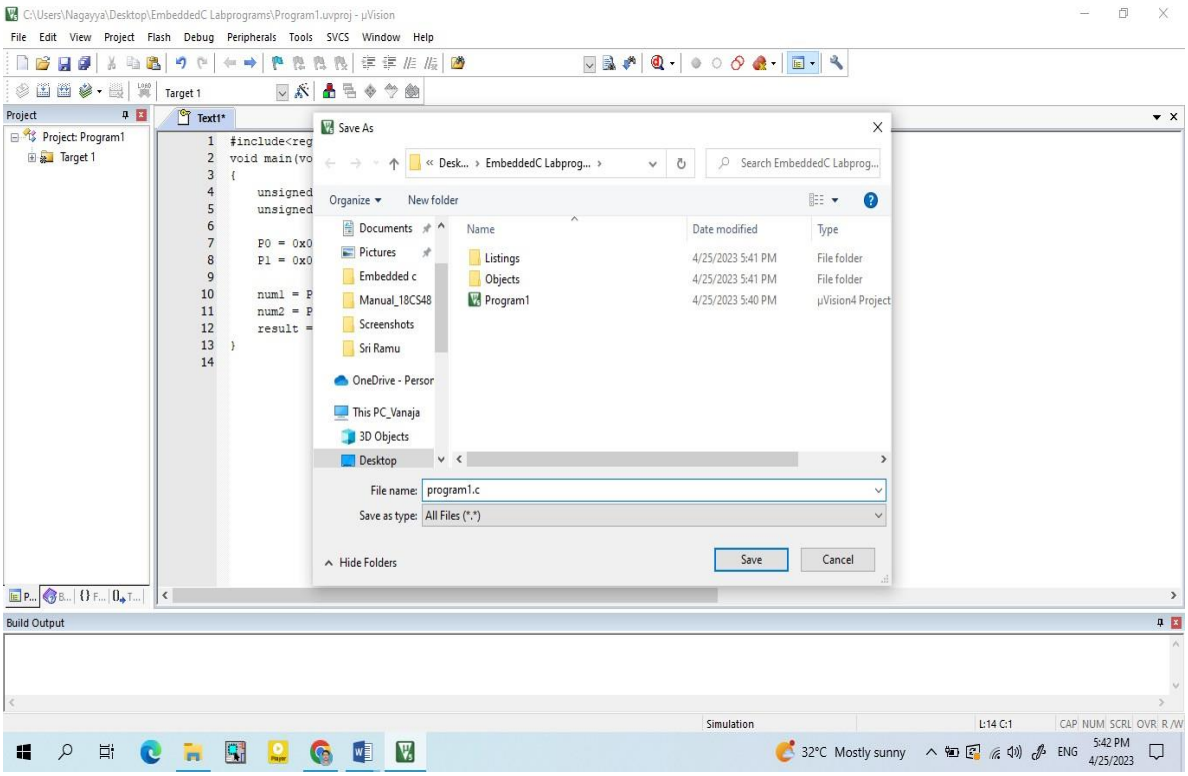
Step6: Create a new file to write the program.



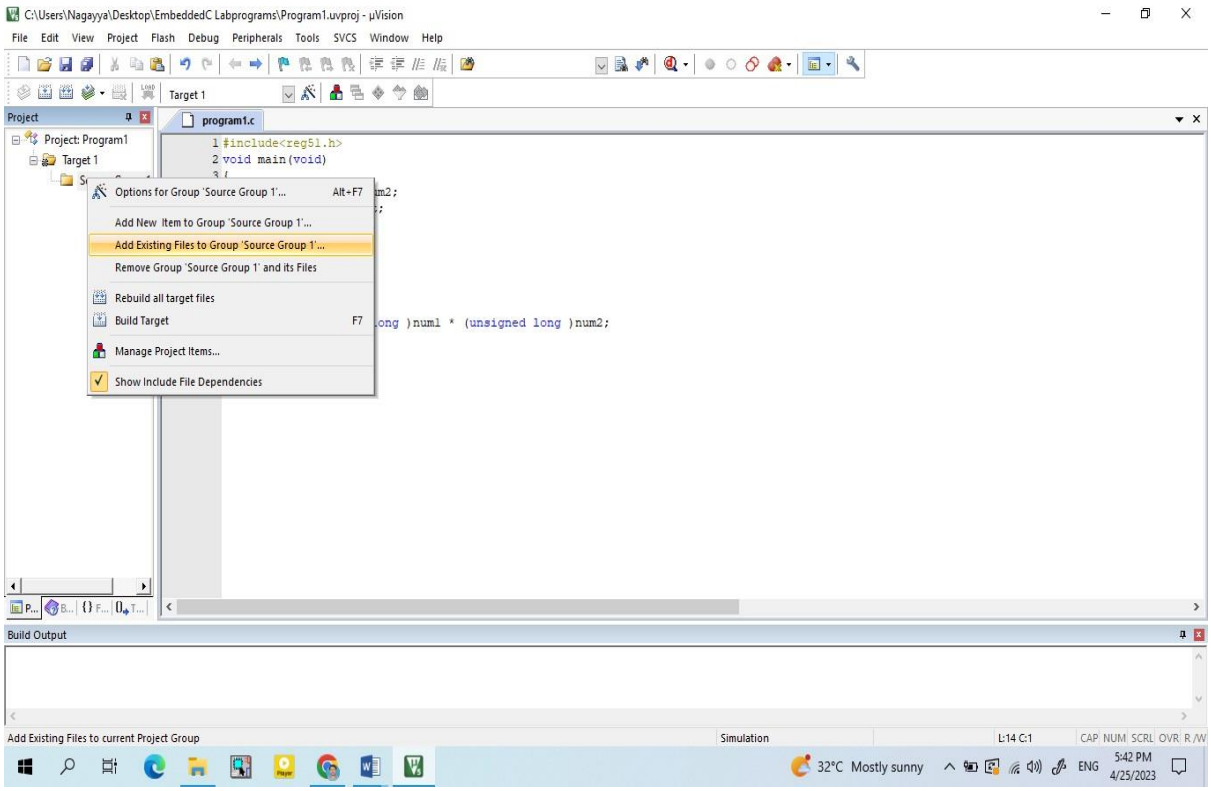
Step7: Type the code.



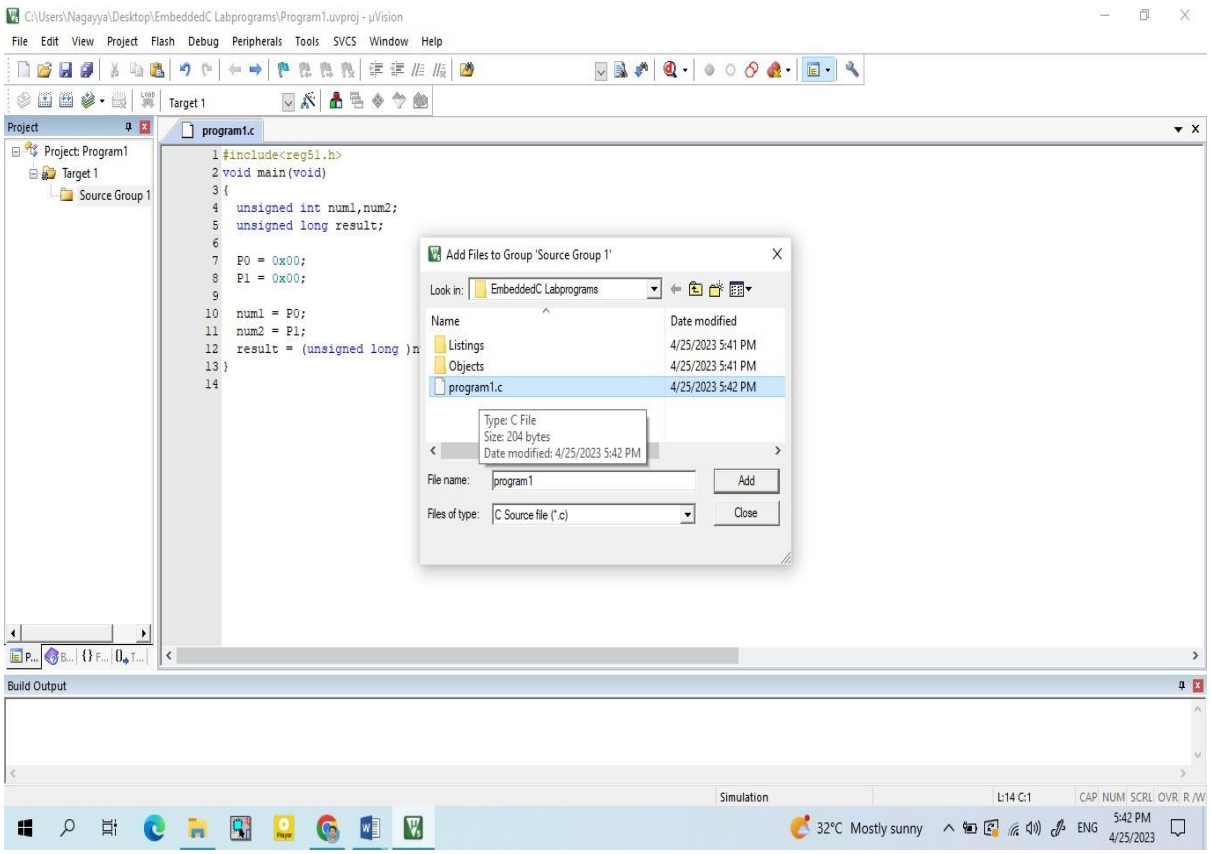
Step8: After typing the code save the file as **FILENAME.C** (user defined filename)



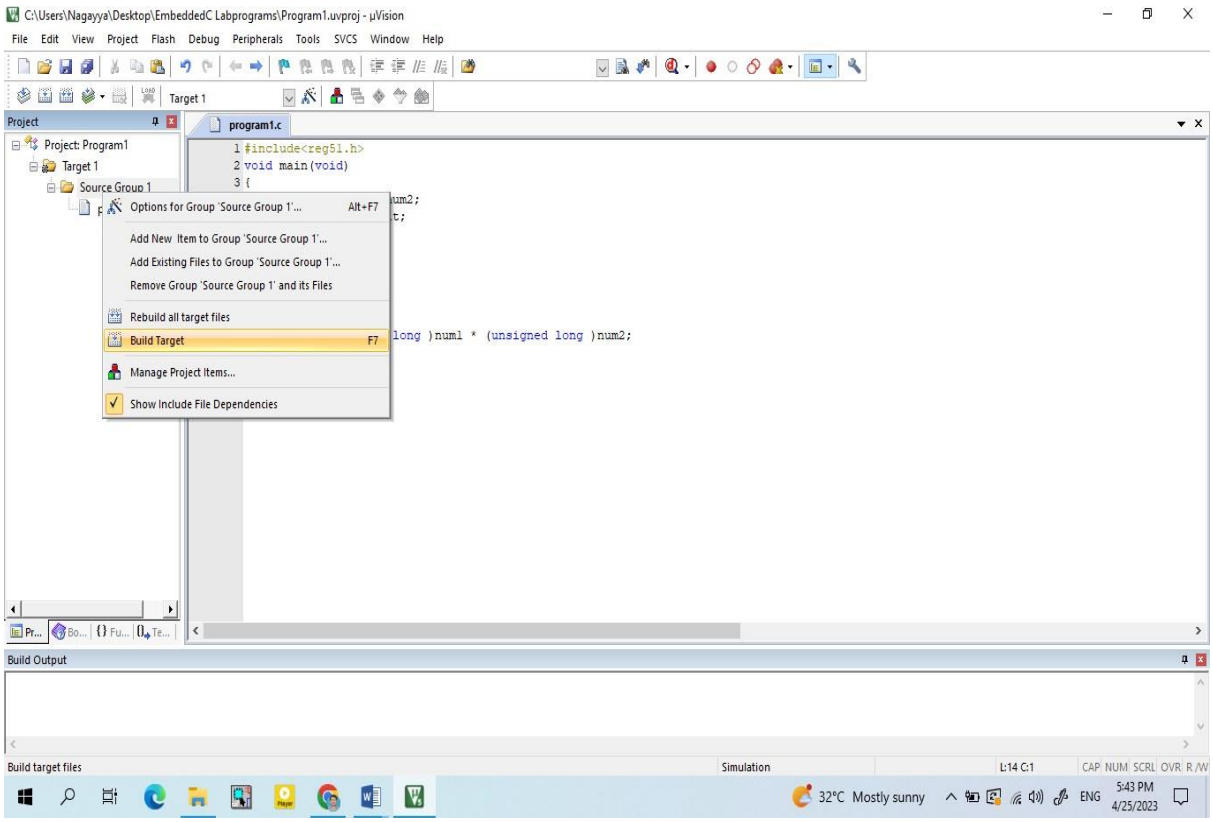
Step9: Add the recently saved file to the project.

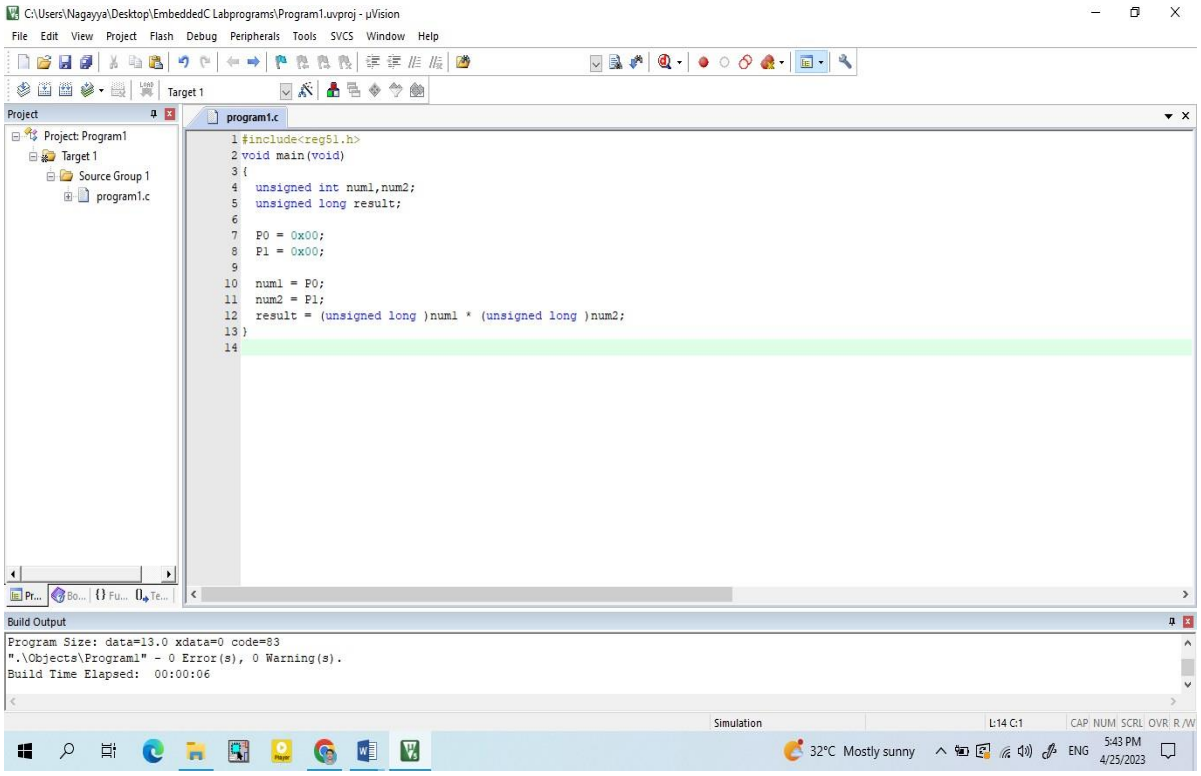


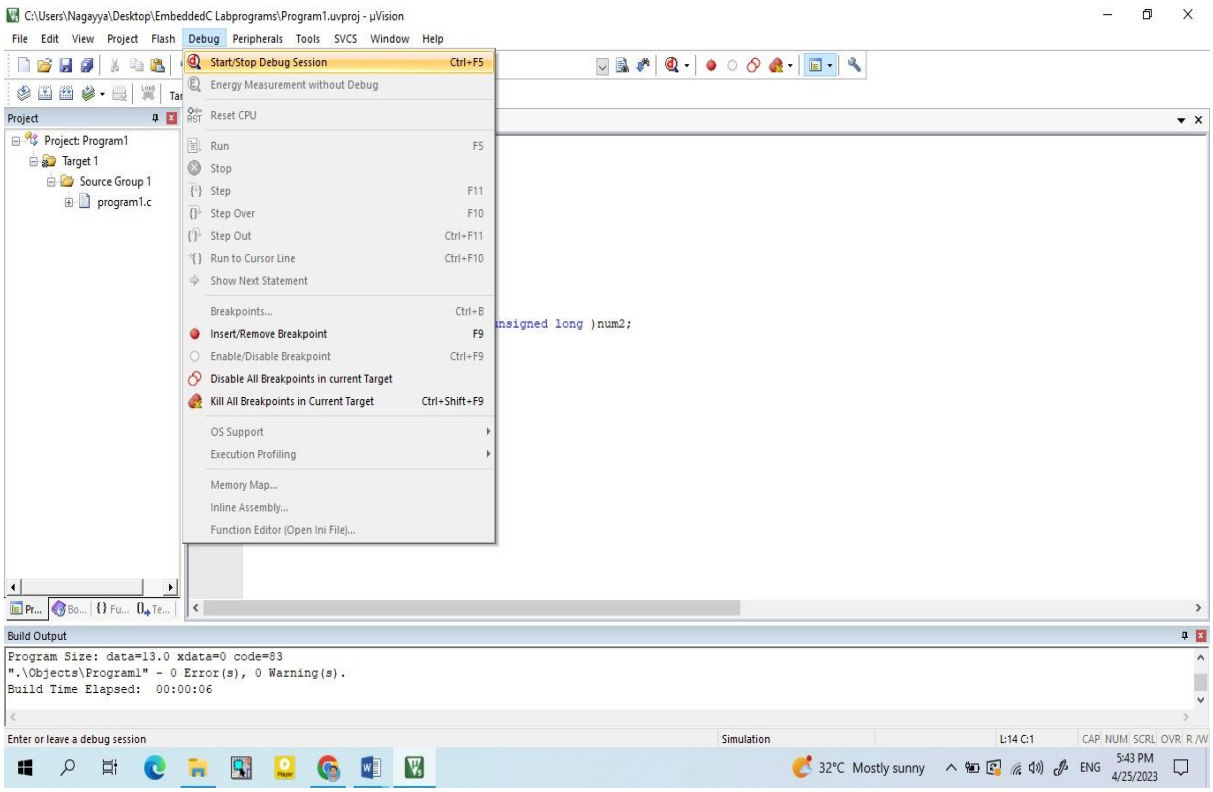
Step10: Add the filename.c



Step11: Build the project and fix the compiler errors/warnings if any.

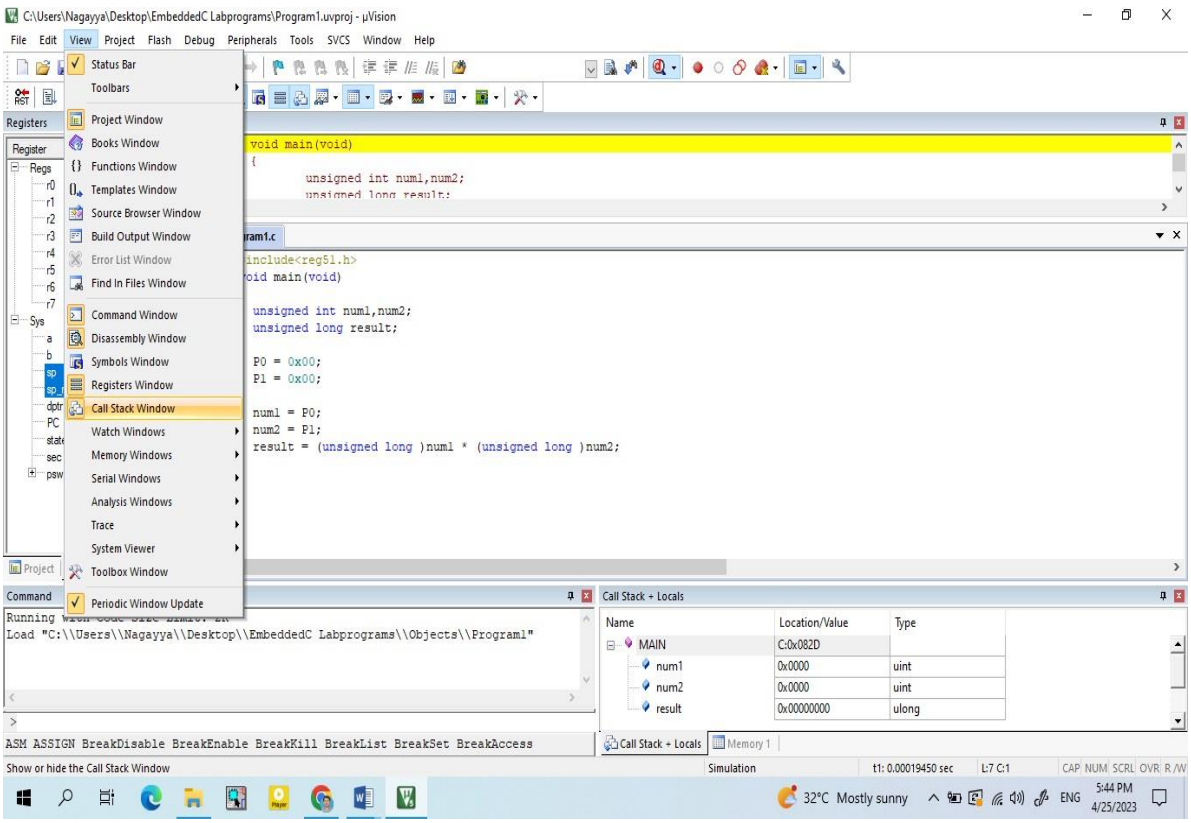


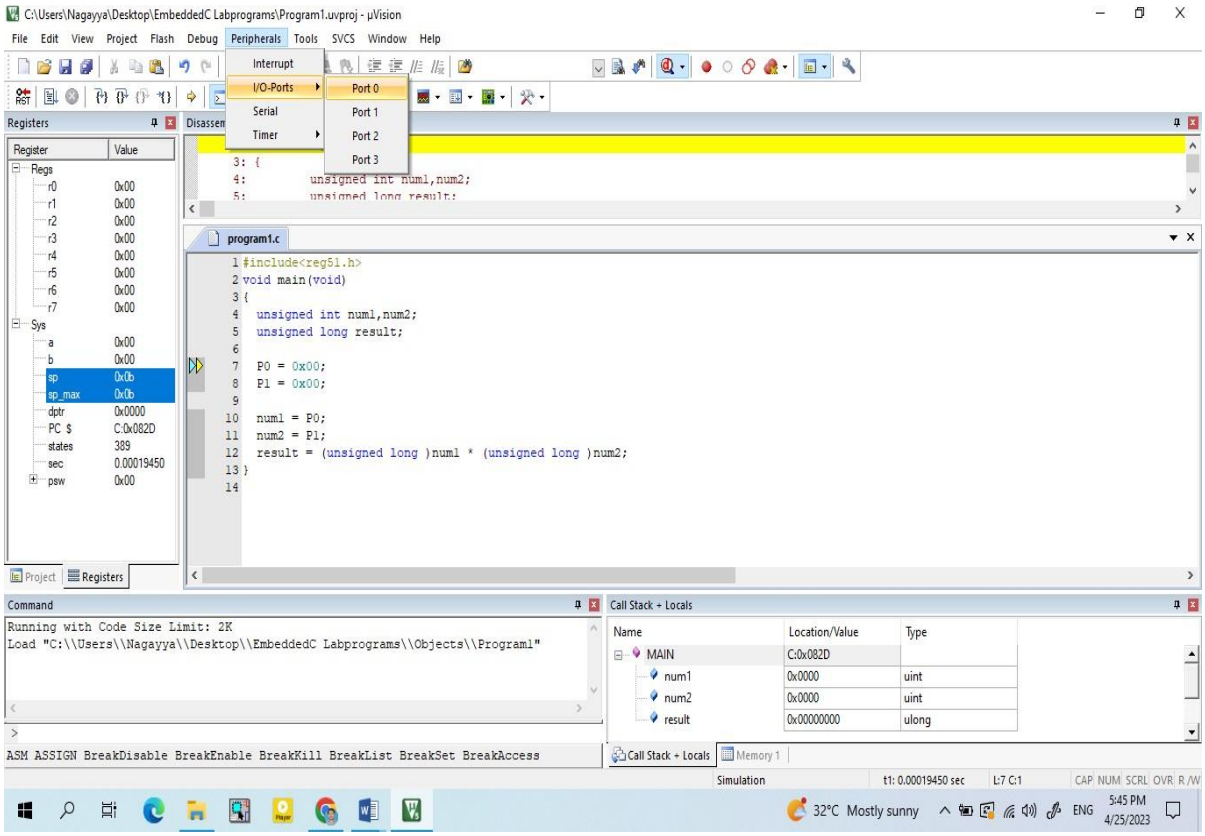


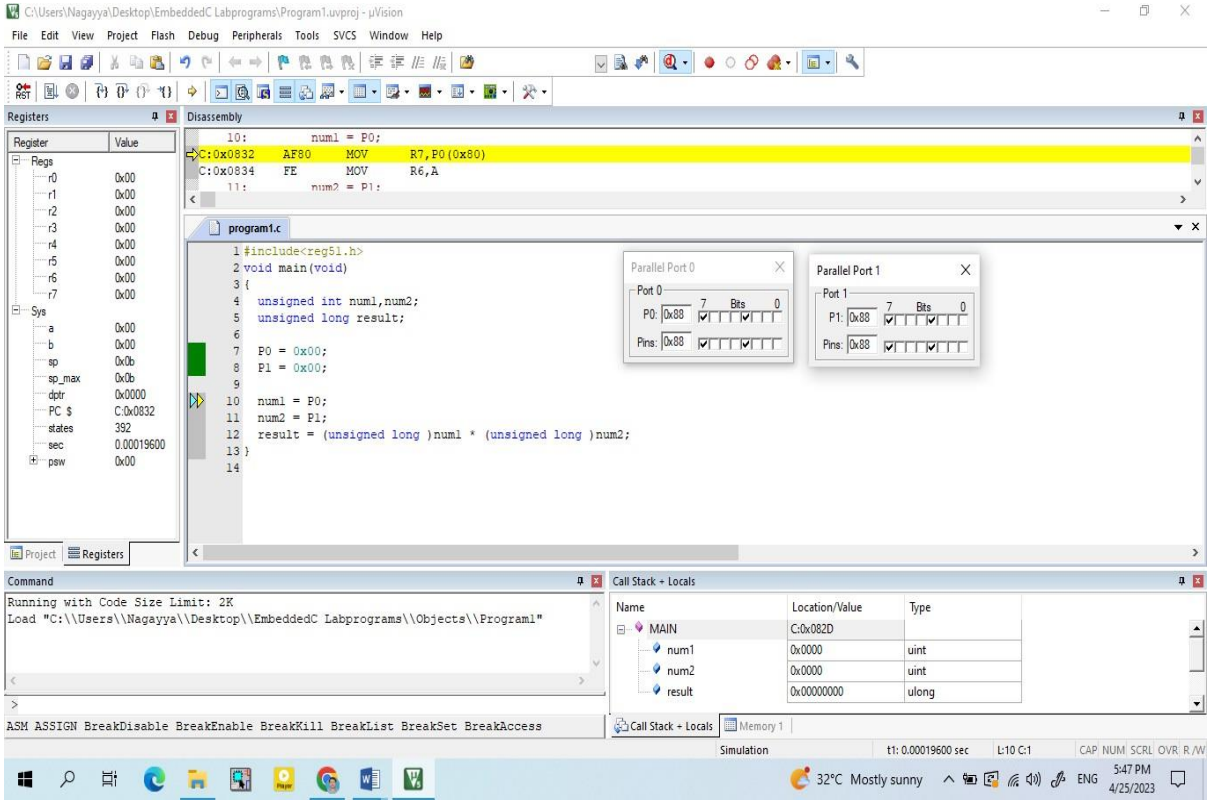


Step12: Code is compiled with no errors.

Note: After debug option, press F11 from keyboard for execution of programs.




C:\Users\Nagayya\Desktop\EmbeddedC Labprograms\Program1.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Interrupt
I/O-Ports
Serial
Timer
Port 0
Port 1
Port 2
Port 3
Registers
Register Value
Regs
r0 0x00
r1 0x00
r2 0x00
r3 0x00
r4 0x00
r5 0x00
r6 0x00
r7 0x00
Sys
a 0x00
b 0x00
sp 0x0b
sp_max 0x0b
dptr 0x0000
PC $ C:0x082D
states 389
sec 0.00019450
psw 0x00
program1.c
1 #include<reg51.h>
2 void main(void)
3 {
4 unsigned int num1,num2;
5 unsigned long result;
6
7 P0 = 0x00;
8 P1 = 0x00;
9
10 num1 = P0;
11 num2 = P1;
12 result = (unsigned long)num1 * (unsigned long)num2;
13 }
14
Command
Running with Code Size Limit: 2K
Load "C:\\Users\\Nagayya\\Desktop\\EmbeddedC Labprograms\\Objects\\Program1"
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess
Call Stack + Locals
Name Location/Value Type
MAIN C:0x082D
num1 0x0000 uint
num2 0x0000 uint
result 0x00000000 ulong
Call Stack + Locals
Memory 1
Simulation
t1: 0.00019450 sec
L:7 C:1
32°C Mostly sunny
ENG
5:45 PM
4/25/2023


C:\Users\Nagayya\Desktop\EmbeddedC Labprograms\Program1.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
10: num1 = P0;
C:0x0832 AF80 MOV R7,P0(0x80)
C:0x0834 FE MOV R6,A
program1.c
Parallel Port 0
Port 0
P0: 0x88
Pins: 0x88
Parallel Port 1
Port 1
P1: 0x88
Pins: 0x88
Command
Running with Code Size Limit: 2K
Load "C:\\Users\\Nagayya\\Desktop\\EmbeddedC Labprograms\\Objects\\Program1"
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess
Call Stack + Locals
Memory 1
Simulation
t1: 0.00019600 sec
L:10 C:1
32°C Mostly sunny
ENG
5:47 PM
4/25/2023