ATME College of Engineering

Department of Electronics and Communication Engineering

LABORATORY MANUAL

(ACADEMIC YEAR 2022-23)

EMBEDDED C BASICS LABORATORY

(AEC-21EC481)

IV Semester

Prepared By:

Mr. GIRISH M.

Assistant Professor

Verified By:

Mr. Abhilash G.
Assistant Professor

Institute Vision & Mission

VISION:

Development of academically excellent, culturally vibrant, socially responsible, and globally competent human resources.

MISSION:

- To keep pace with advancements in knowledge and make the students competitive and capable at the global level.
- To create an environment for the students to acquire the right physical, intellectual, emotional, and moral foundations and shine as torch bearers of tomorrow's society.
- To strive to attain ever-higher benchmarks of educational excellence

Department Vision and Mission

VISION:

To develop highly skilled and globally competent professionals in the field of Electronics and Communication Engineering to meet industrial and social requirements with ethical responsibility.

MISSION:

- To provide State-of-art technical education in Electronics and Communication at undergraduate and post-graduate levels to meet the needs of the profession and society and to adopt the best educational methods and achieve excellence in teaching-learning and research.
- To develop talented and committed human resource, by providing an opportunity for innovation, creativity, and entrepreneurial leadership with high standards of professional ethics, transparency, and accountability.
- To function collaboratively with technical Institutes/Universities/Industries and offer opportunities for long-term interaction with academia and industry.

PROGRAM OUTCOMES (POs)

Engineering Graduates will be able to:

PO1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6. The engineer and society: Apply reasoning informed by contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to professional engineering practice.

PO7. Environment and sustainability: Understand the impact of professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of engineering practice.

PO9. Individual and teamwork: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

PROGRAM SPECIFIC OUTCOMES (PSOs)

- To have the capability to understand and adapt to technological advancements with the usage of modern tool to analyze and design embedded system or processes for variety of applications.
- To work effectively in a group as an independent visionary, team member and leader having the ability to understand the requirement and develop feasible solutions to emerge as potential core or electronic engineer.

PROGRAM EDUCATIONAL OBJECTIVES (PEO's)

- To Produce Graduates to excel in the profession, higher education and pursue research exercises in Electronics and Communication Engineering.
- To create technically able alumni with the capacity to examine, plan, create and execute Electronics and Communication frameworks thereby involving in deep rooted learning

Syllabus

Conduct the following experiments by writing C Program using Keil microvision simulator (any 8051 microcontroller can be chosen as the target).

1. Write a 8051 C program to multiply two 16 bit binary numbers.
2. Write a 8051 C program to find the sum of first 10 integer numbers.
3. Write a 8051 C program to find factorial of a given number.
4. Write a 8051 C program to add an array of 16 bit numbers and store the 32 bit result in internal RAM
5. Write a 8051 C program to find the square of a number (1 to 10) using look-up table.
6. Write a 8051 C program to find the largest/smallest number in an array of 32 numbers
7. Write a 8051 C program to arrange a series of 32 bit numbers in ascending/descending order
8. Write a 8051 C program to count the number of ones and zeros in two consecutive memory locations.
9. Write a 8051 C program to scan a series of 32 bit numbers to find how many are negative.
10. Write a 8051 C program to display "Hello World" message (either in simulation mode on interface an LCD display).
11. 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).

Course Outcomes:

At the end of the course the student will be able to:

1. Write C programs in 8051 for solving simple problems that manipulate input data using different instructions of 8051 C.

2. Develop testing and experimental procedures on 8051 Microcontroller; analyze their operation under different cases.

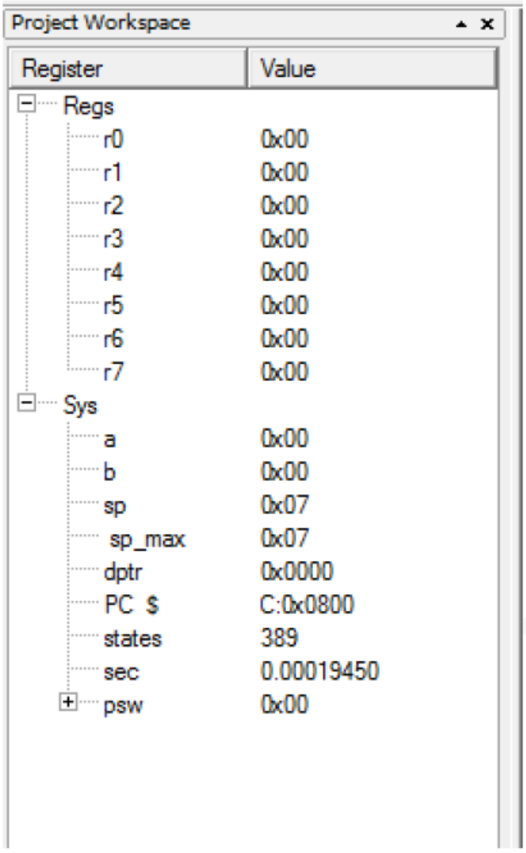
3. Develop programs for 8051 Microcontroller to implement real world problems.

4. Design and Develop Mini projects

Table of Contents

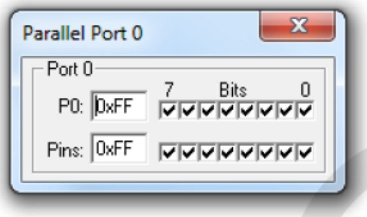
Output of Experiment-1

Before Execution



Project Workspace

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	0x07
sp_max	0x07
dptr	0x0000
PC $	C:0x0800
states	389
sec	0.00019450
⊞ psw	0x00

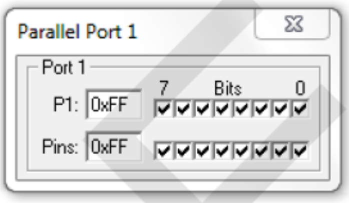


Parallel Port 0

Port 0 — 7 Bits 0

P0: 0xFF ☑☑☑☑☑☑☑☑

Pins: 0xFF ☑☑☑☑☑☑☑☑

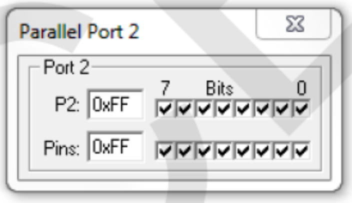


Parallel Port 1

Port 1 — 7 Bits 0

P1: 0xFF ☑☑☑☑☑☑☑☑

Pins: 0xFF ☑☑☑☑☑☑☑☑



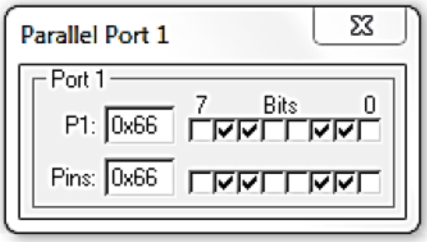
Parallel Port 2

Port 2 — 7 Bits 0

P2: 0xFF ☑☑☑☑☑☑☑☑

Pins: 0xFF ☑☑☑☑☑☑☑☑

After Execution

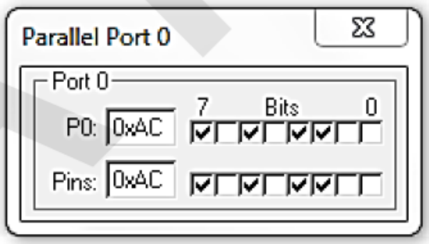


Parallel Port 1

Port 1 — 7 Bits 0

P1: 0x66 ☐☑☑☐☐☑☑☐

Pins: 0x66 ☐☑☑☐☐☑☑☐

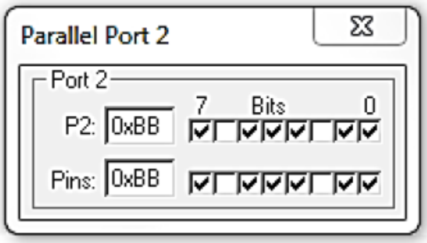


Parallel Port 0

Port 0 — 7 Bits 0

P0: 0xAC ☑☐☑☐☑☑☐☐

Pins: 0xAC ☑☐☑☐☑☑☐☐



Parallel Port 2

Port 2 — 7 Bits 0

P2: 0xBB ☑☐☑☑☑☐☑☑

Pins: 0xBB ☑☐☑☑☑☐☑☑

Experiment- 1

Arithmetic Operations: Multiplication of Numbers

Objective: To write a 8051 C program to multiply two 16 bit binary numbers.

Program:

```
#include<reg51.h>            // to include the 8051 register set
void main()
{
   unsigned int i, j;          // declare the inputs

   i=0x89;                     // feed the input value
   j=0x23;                     // feed the input value

   P0=i+j;    //addition of two inputs & store it in Port 0      or 0x80H
   P1=i-j;    //subtraction of two inputs& store it in Port 0 or 0x90H
   P2=i*j;    //multiplication of two inputs & store it in Port 0 or 0xA0H

}
```

Result:

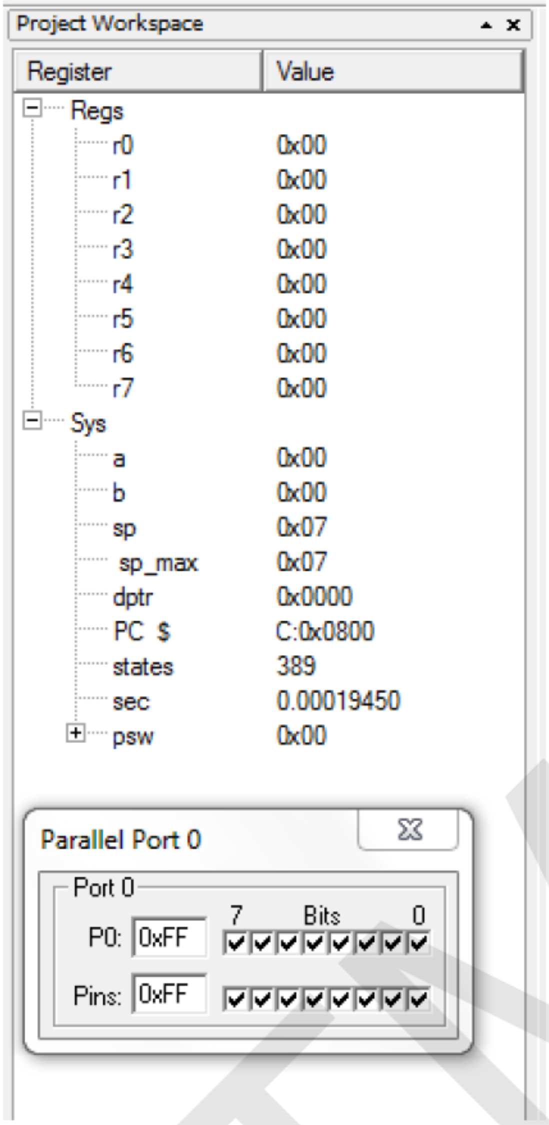
Written the code and verified the multiplication of 8 bit, 16bit numbers.

Self-Learning Task-1:

Write code to add, subtract and multiply ABh and 89h and show the substantial calculations

Output of Experiment-2

Before execution



Project Workspace

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	0x07
sp_max	0x07
dptr	0x0000
PC $	C:0x0800
states	389
sec	0.00019450
psw	0x00

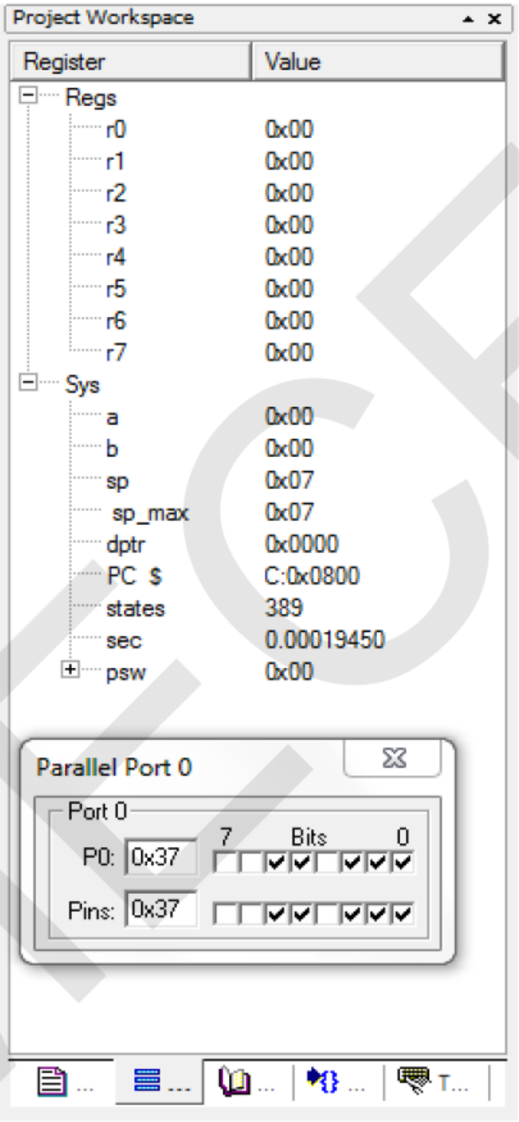
Parallel Port 0

Port 0

P0: 0xFF   7 Bits 0 ☑☑☑☑☑☑☑☑

Pins: 0xFF ☑☑☑☑☑☑☑☑

After Execution



Project Workspace

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	0x07
sp_max	0x07
dptr	0x0000
PC $	C:0x0800
states	389
sec	0.00019450
psw	0x00

Parallel Port 0

Port 0

P0: 0x37   7 Bits 0 □□☑☑□☑☑☑

Pins: 0x37 □□☑☑□☑☑☑

Experiment- 2

Sum of Integers

Objective: To write a 8051 C program to find the sum of first 10 integer numbers.

Program:

```
#include<reg51.h>                // to include the 8051 register set
void main()
{
   unsigned char i;
   unsigned int sum = 0;         // Initialize sum with zero
for (i = 1; i <= 10; i++)        //Loop to 10 iteration
   {
         sum += i;                //Increment and add
   }
P0=sum;                          //Display the sum

while (1);                       //Loop indefinitely
}
```

Result:

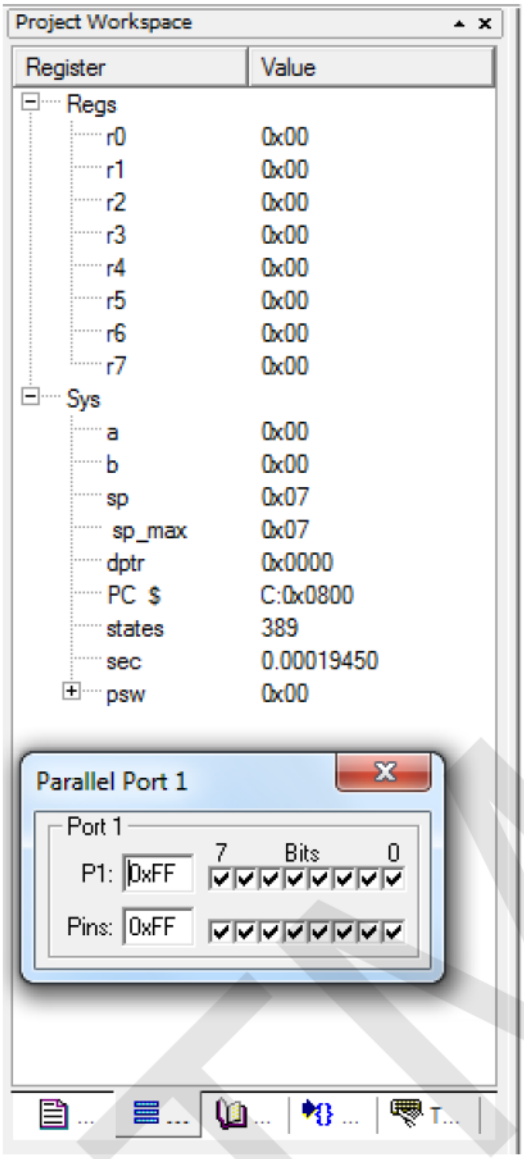
Written the code and verified the sum of first ten integers.

Self-Learning Task-2:

Can you try to get the sum of only odd numbers of first 15 integers?

Output of Experiment-3

Before execution



Project Workspace

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	0x07
sp_max	0x07
dptr	0x0000
PC $	C:0x0800
states	389
sec	0.00019450
psw	0x00

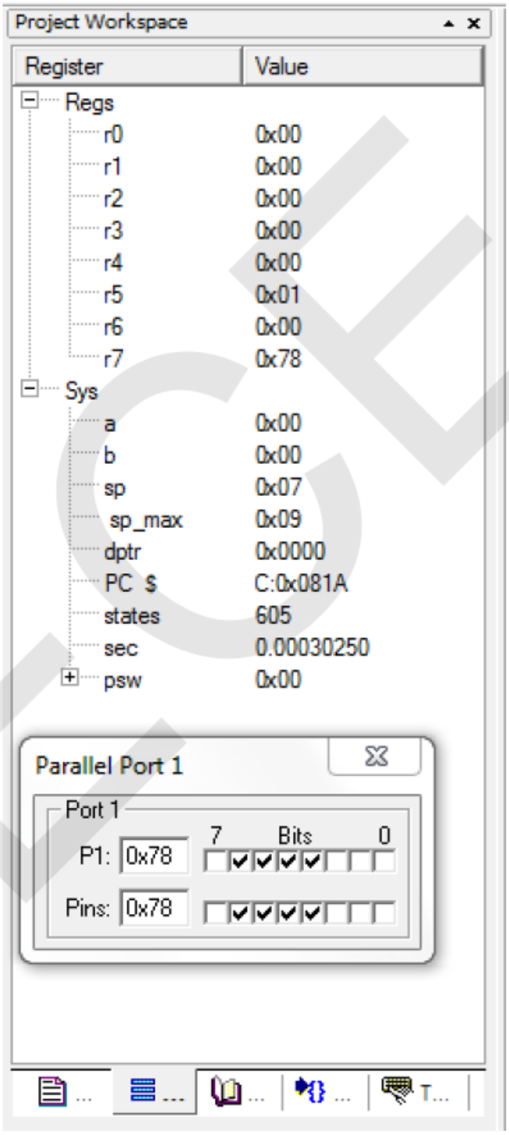
Parallel Port 1

Port 1 — 7 Bits 0

P1: 0xFF ☑☑☑☑☑☑☑☑

Pins: 0xFF ☑☑☑☑☑☑☑☑

After Execution



Project Workspace

Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x01
r6	0x00
r7	0x78
Sys	
a	0x00
b	0x00
sp	0x07
sp_max	0x09
dptr	0x0000
PC $	C:0x081A
states	605
sec	0.00030250
psw	0x00

Parallel Port 1

Port 1 — 7 Bits 0

P1: 0x78 □☑☑☑☑□□□

Pins: 0x78 □☑☑☑☑□□□

Experiment- 3

Factorial of Numbers

Objective: To write a 8051 C program to find the factorial of a number.

Program:

```
#include <reg51.h> //Include the 8051 microcontroller header file
void main()
{
   //Give the number to which you need the factorial of the given number & Initialize with 1
   unsigned int num=5, fact=1;
   while(num>0)
   {
         fact = fact*num;
         num--;
   }
P1 = fact;     //Display the result on the output port
   while(1);
}
```

Result:

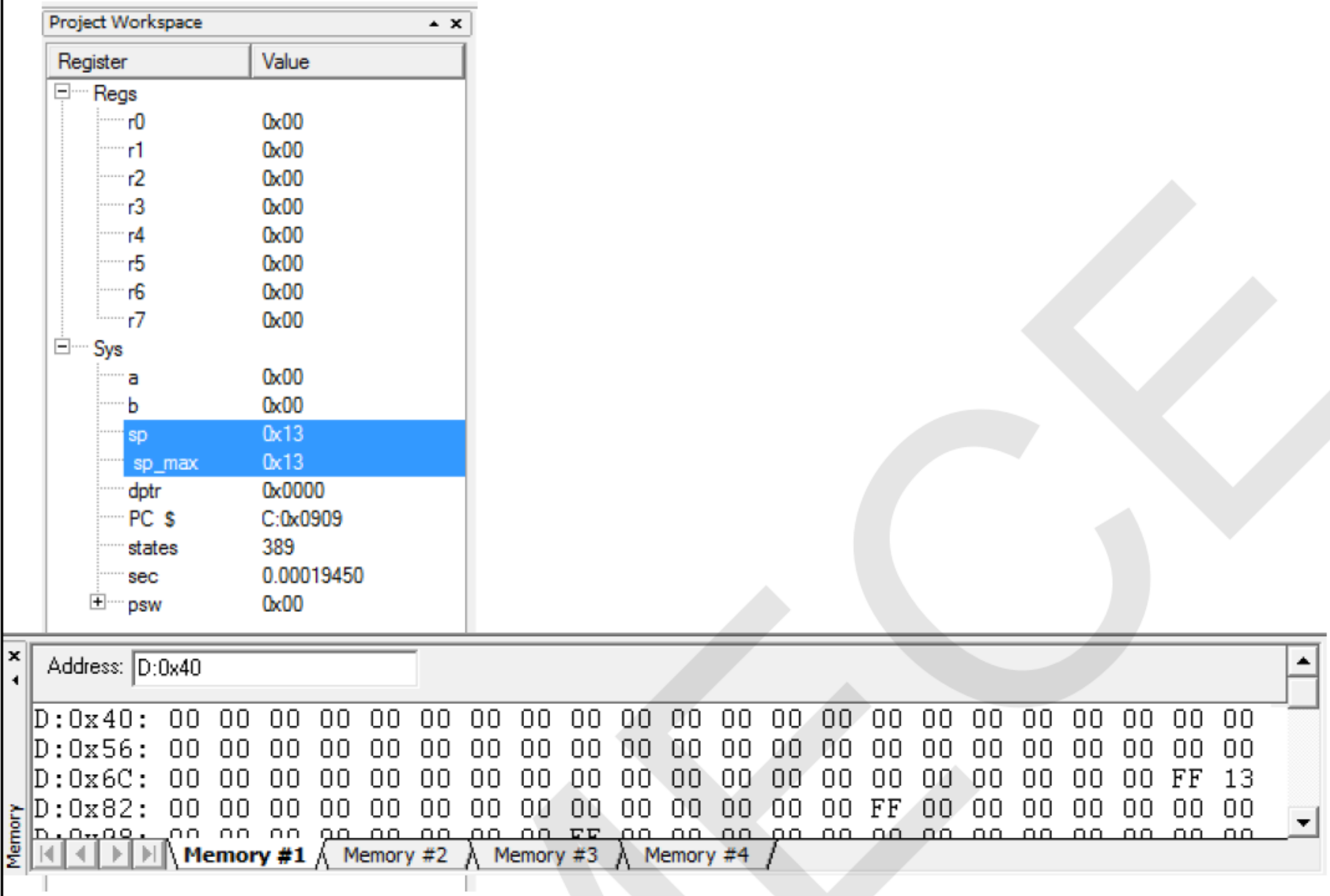
Written the code and verified the factorial of given number.

Self-Learning Task-3:

Determine the factorial of 25, try to use for loop and compute

Output of Experiment-4

Before execution



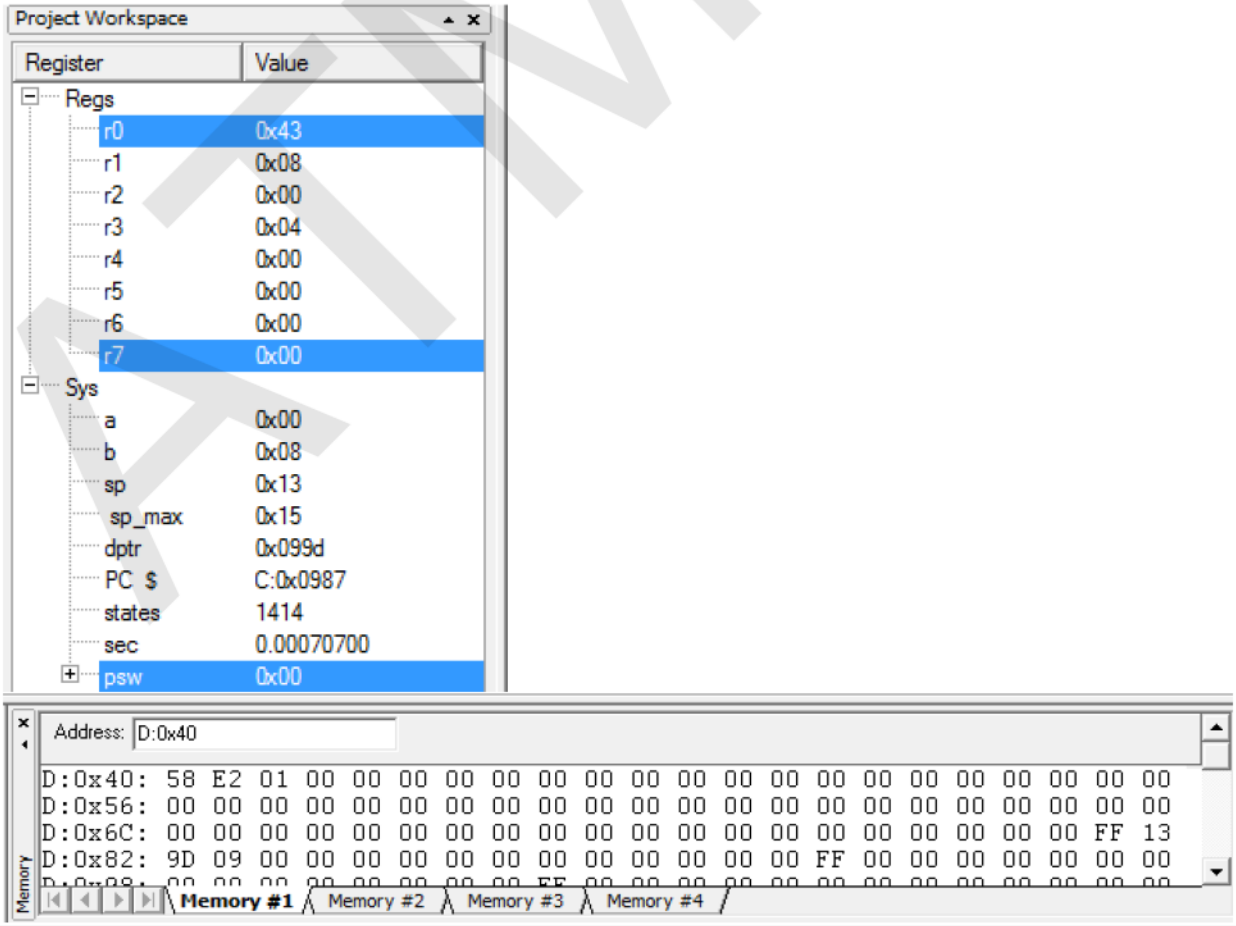
Project Workspace

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	0x13
sp_max	0x13
dptr	0x0000
PC $	C:0x0909
states	389
sec	0.00019450
psw	0x00

Address: D:0x40

```
D:0x40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x56: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x6C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF 13
D:0x82: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF 00 00 00 00 00 00 00
```

After Execution



Project Workspace

Register	Value
Regs	
r0	0x43
r1	0x08
r2	0x00
r3	0x04
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x08
sp	0x13
sp_max	0x15
dptr	0x099d
PC $	C:0x0987
states	1414
sec	0.00070700
psw	0x00

Address: D:0x40

```
D:0x40: 58 E2 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x56: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x6C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF 13
D:0x82: 9D 09 00 00 00 00 00 00 00 00 00 00 00 00 FF 00 00 00 00 00 00 00
```

Experiment- 4

Array addition

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

Program:

```
#include <reg51.h>
#define MEMORY_LOCATION (0x40)  // Starting memory location to store the result
void main()
{
    unsigned int array[4] = {0x1234, 0x5678, 0x9ABC, 0xDEF0}; // Sample array of 16-bit numbers
    unsigned long sum = 0;
    unsigned char i;
for (i = 0; i <4; i++)
    {
    sum += array[i];
    }
// Store the 32-bit result in internal RAM
// assuming the result will fit in 32 bits
// Low byte of the sum
*((unsigned char *)(MEMORY_LOCATION + 0)) = (unsigned char)(sum & 0xFF);
// High byte of the sum
*((unsigned char *)(MEMORY_LOCATION + 1)) = (unsigned char)((sum >> 8) & 0xFF);
// Lower word of the sum
*((unsigned char *)(MEMORY_LOCATION + 2)) = (unsigned char)((sum >> 16) & 0xFF);
// Upper word of the sum
*((unsigned char *)(MEMORY_LOCATION + 3)) = (unsigned char)((sum >> 24) & 0xFF);
while (1); //program execution stops here
}
```

Result:

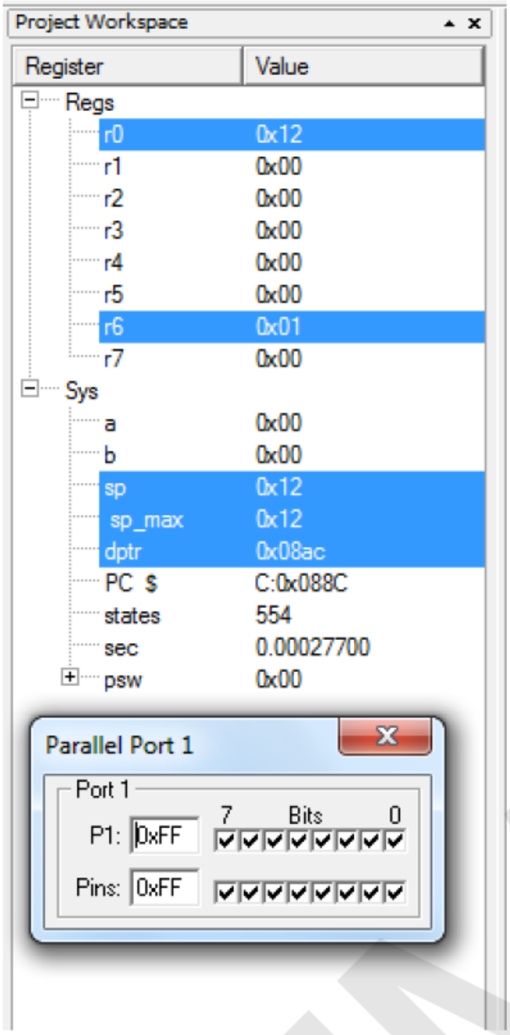
Written the code and verified the addition of array elements.

Self-Learning Task-4:

Determine the addition of array of 8 elements and store it in memory location

Output of Experiment-5

Before execution



Project Workspace

Register	Value
Regs	
r0	0x12
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x01
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x12
sp_max	0x12
dptr	0x08ac
PC $	C:0x088C
states	554
sec	0.00027700
psw	0x00

Parallel Port 1

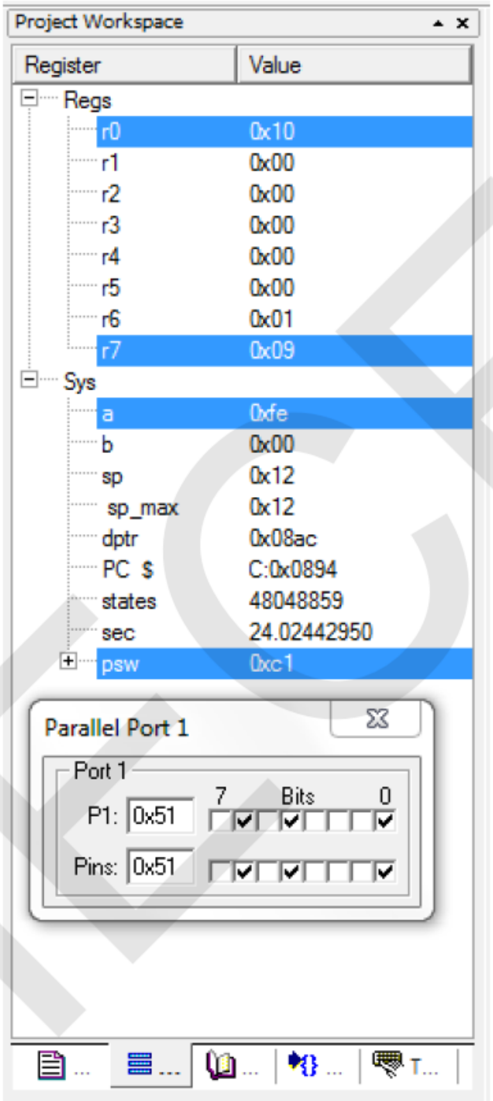
Port 1

7 Bits 0

P1: 0xFF ☑☑☑☑☑☑☑☑

Pins: 0xFF ☑☑☑☑☑☑☑☑

After Execution



Project Workspace

Register	Value
Regs	
r0	0x10
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x01
r7	0x09
Sys	
a	0xfe
b	0x00
sp	0x12
sp_max	0x12
dptr	0x08ac
PC $	C:0x0894
states	48048859
sec	24.02442950
psw	0xc1

Parallel Port 1

Port 1

7 Bits 0

P1: 0x51 □☑□☑□□□☑

Pins: 0x51 □☑□☑□□□☑

Experiment- 5

Square of a Number

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

Program:

```
#include <reg51.h>
// Look-up table of squares
const unsigned char squares[] = {1, 4, 9, 16, 25, 36, 49, 64, 81, 100};
void main()
{
    unsigned char number = 9; // Number for which we want to find the square
    while (number <= 10)
    {
    unsigned char square = squares[number - 1];
    P1 = square; // Display the result on Port 1
    }
}
```

Result:

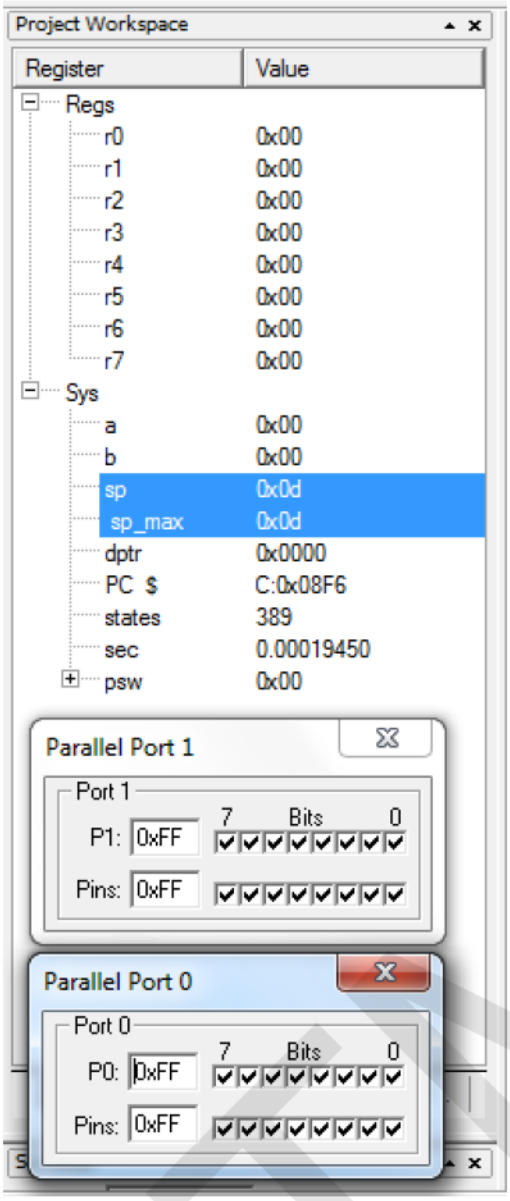
Written the code and verified the square of a number (1 to 10) using lookup table.

Self-Learning Task-5:

Determine the square of 25 using lookup table

Output of Experiment-6

Before execution



Project Workspace

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	0x0d
sp_max	0x0d
dptr	0x0000
PC $	C:0x08F6
states	389
sec	0.00019450
psw	0x00

Parallel Port 1

Port 1

P1: 0xFF 7 Bits 0 ☑☑☑☑☑☑☑☑

Pins: 0xFF ☑☑☑☑☑☑☑☑

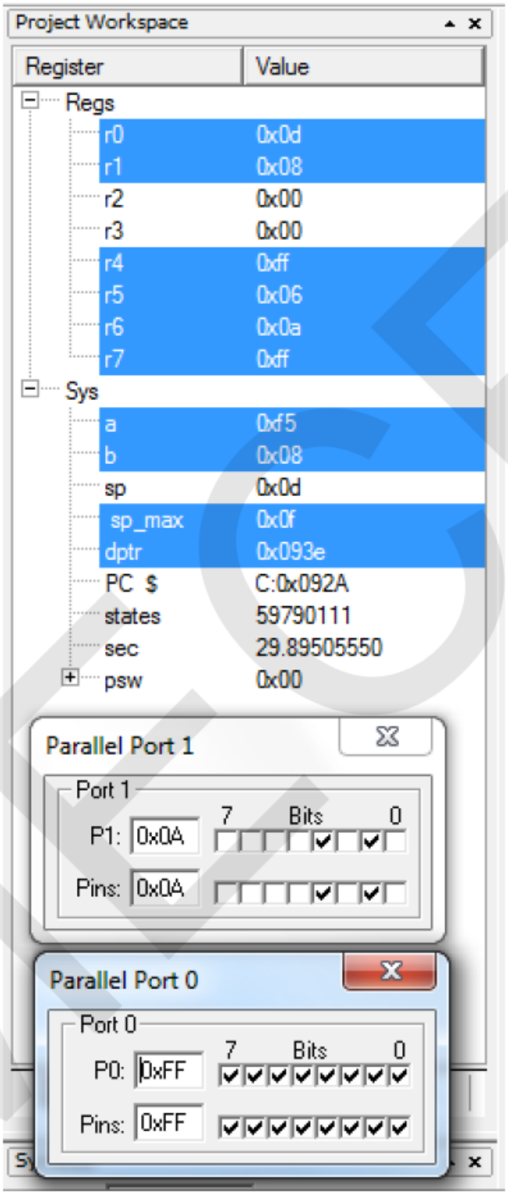
Parallel Port 0

Port 0

P0: 0xFF 7 Bits 0 ☑☑☑☑☑☑☑☑

Pins: 0xFF ☑☑☑☑☑☑☑☑

After Execution



Project Workspace

Register	Value
Regs	
r0	0x0d
r1	0x08
r2	0x00
r3	0x00
r4	0xff
r5	0x06
r6	0x0a
r7	0xff
Sys	
a	0xf5
b	0x08
sp	0x0d
sp_max	0x0f
dptr	0x093e
PC $	C:0x092A
states	59790111
sec	29.89505550
psw	0x00

Parallel Port 1

Port 1

P1: 0x0A 7 Bits 0 □□□□☑□☑□

Pins: 0x0A □□□□☑□☑□

Parallel Port 0

Port 0

P0: 0xFF 7 Bits 0 ☑☑☑☑☑☑☑☑

Pins: 0xFF ☑☑☑☑☑☑☑☑

Experiment- 6

Smallest & Largest Numbers

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

Program:

```
#include <reg51.h>
#define ARRAY_SIZE 6        //declare array size
void main()
{
    // declare array elements
    unsigned char array[ARRAY_SIZE] = {0x25, 0x55,0x13,0x62,0x0a,0xff};
    unsigned char largest = array[0];
    unsigned char smallest = array[0];
    unsigned char i;
for (i = 1; i < ARRAY_SIZE; i++)
    {
            if (array[i] >largest) {
            largest = array[i];
    }
if (array[i] < smallest)
    {
    smallest = array[i];
    }
}
    // Show the Output results at 8051 ports
    P0 = largest;           // Display largest number on P0
    P1 = smallest;          // Display smallest number on P1
while (1)                   // Infinite loop
{
    }
}
```

Result:

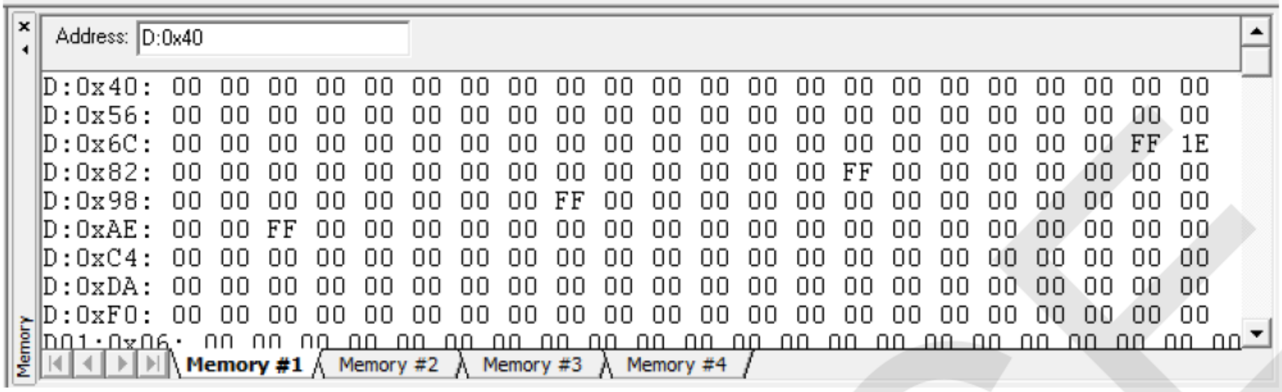
Written the code and found the smallest and largest numbers in the array.

Self-Learning Task-5:

Determine the largest number in an array of 32bit numbers with 8 elements

Output of Experiment-7

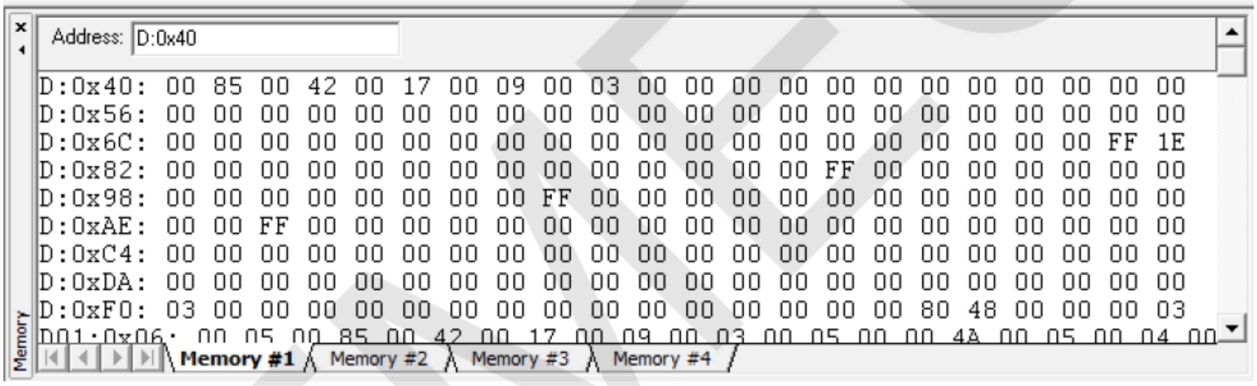
Before execution



Address: D:0x40

```
D:0x40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x56: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x6C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF 1E
D:0x82: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF 00 00 00 00 00 00 00
D:0x98: 00 00 00 00 00 00 00 00 FF 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0xAE: 00 00 FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0xC4: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0xDA: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0xF0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D01:0x06: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

After Execution



Address: D:0x40

```
D:0x40: 00 85 00 42 00 17 00 09 00 03 00 00 00 00 00 00 00 00 00 00 00 00
D:0x56: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x6C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF 1E
D:0x82: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF 00 00 00 00 00 00 00
D:0x98: 00 00 00 00 00 00 00 00 FF 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0xAE: 00 00 FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0xC4: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0xDA: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0xF0: 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80 48 00 00 00 03
D01:0x06: 00 05 00 85 00 42 00 17 00 09 00 03 00 05 00 00 4A 00 05 00 04 00
```

Experiment- 7

Sorting of numbers

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

Program:

```
#include <reg51.h>
void bubbleSort(unsigned int arr[], int size)
{
    int i, j;
    unsigned int temp;
for (i = 0; i < size - 1; i++)
    {
    for (j = 0; j < size - i - 1; j++)
            {
            if (arr[j] < arr[j + 1])
            {
                    temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
            }
            }
    }
}
void main()
{
    unsigned int numbers[] = {0x42, 0x9, 0x17, 0x85, 0x3}; // Replace with your own numbers
    int size = sizeof(numbers) / sizeof(numbers[0]);
    int i;
    // Starting memory address to store the sorted numbers
    unsigned int *memoryLocation = (unsigned int *)0x40;
    bubbleSort(numbers, size);

for (i = 0; i < size; i++)         // Save the sorted numbers in memory locations
{
*memoryLocation = numbers[i];
memoryLocation++;
}
while(1) {
}
}
```

Result:

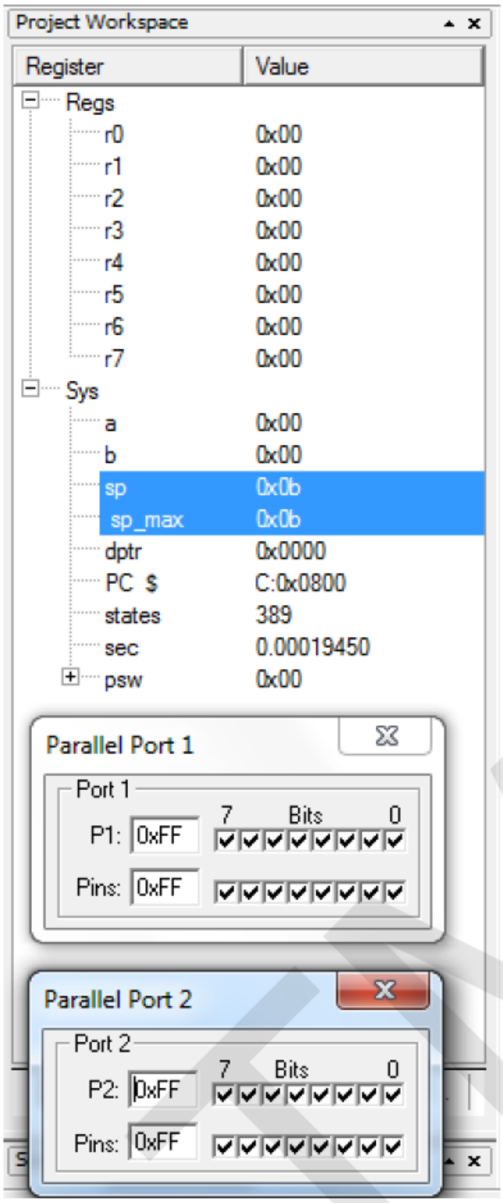
Written the code and sorted the numbers in ascending and descending orders.

Self-Learning Task-7:

Can you make a try to sort only odd numbers in a series of 32bit numbers?

Output of Experiment-8

Before execution



Project Workspace

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:0x0800
states	389
sec	0.00019450
psw	0x00

Parallel Port 1

Port 1 — 7 Bits 0

P1: 0xFF ☑☑☑☑☑☑☑☑

Pins: 0xFF ☑☑☑☑☑☑☑☑

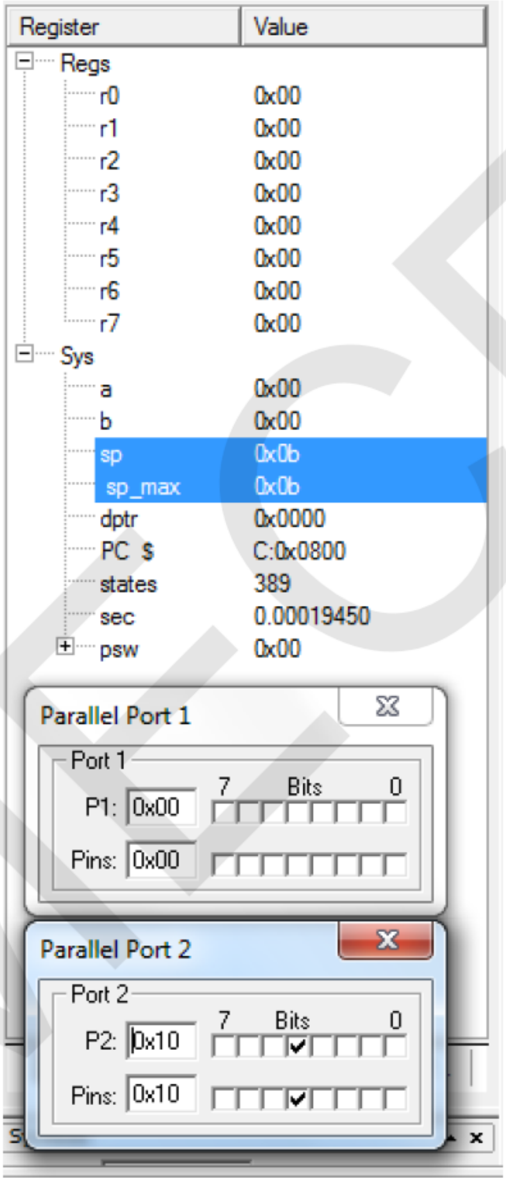
Parallel Port 2

Port 2 — 7 Bits 0

P2: 0xFF ☑☑☑☑☑☑☑☑

Pins: 0xFF ☑☑☑☑☑☑☑☑

After Execution



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:0x0800
states	389
sec	0.00019450
psw	0x00

Parallel Port 1

Port 1 — 7 Bits 0

P1: 0x00 □□□□□□□□

Pins: 0x00 □□□□□□□□

Parallel Port 2

Port 2 — 7 Bits 0

P2: 0x10 □□□☑□□□□

Pins: 0x10 □□□☑□□□□

Experiment- 8

Count the number of 1’s & 0’s

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

Program:

```
#include <reg51.h>
void main()
{
unsigned char data1, data2;
unsigned char onesCount1 = 0, onesCount2 = 0;
unsigned char zerosCount1 = 0, zerosCount2 = 0;
unsigned char i;
// Read the data from two consecutive memory locations
data1 = *(unsigned char*)0x40;
data2 = *(unsigned char*)0x41;
// Count the number of ones and zeros in the first memory location
for (i = 0; i < 8; i++)
    {
    if ((data1 & (1 << i)) != 0)
    onesCount1++;
    else
    zerosCount1++;
    }
// Count the number of ones and zeros in the second memory location
for (i = 0; i < 8; i++)
    {
    if ((data2 & (1 <<i)) != 0)
    onesCount2++;
    else
    zerosCount2++;
    }
// Output the results
P1 = onesCount1 + onesCount2; // Output ones count for the first + second memory
location
P2 = zerosCount1 + zerosCount2; // Output zeros count for the first + second memory
location
while (1) { // Endless loop
}
}
```

Result:

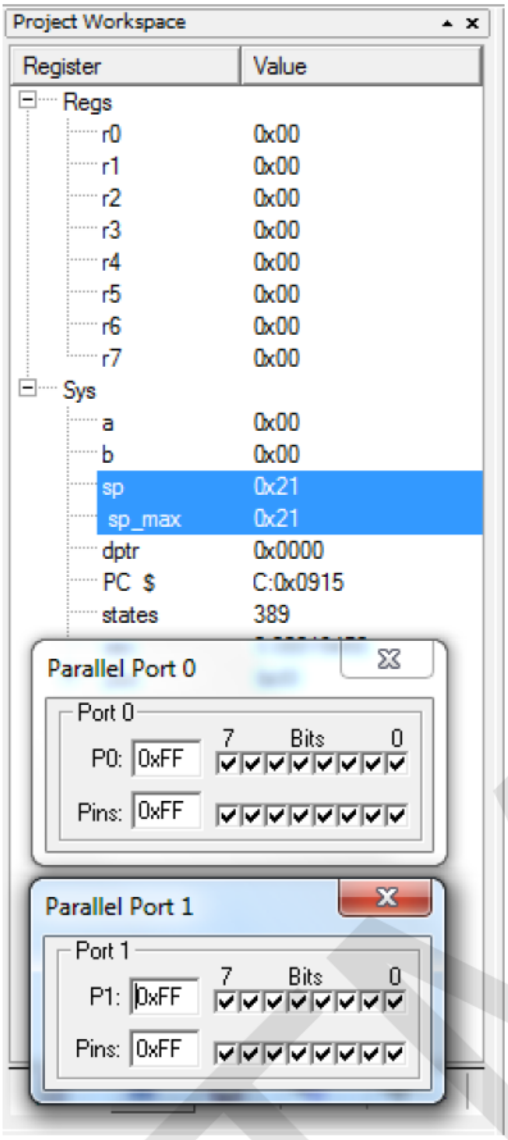
Written the code and sorted the numbers in ascending and descending orders.

Self-Learning Task-8:

Can you try to find the number of 1’s and 0’s in a 16-bit data?

Output of Experiment-9

Before execution



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	0x21
sp_max	0x21
dptr	0x0000
PC $	C:0x0915
states	389

Port 0

P0: 0xFF   7 Bits 0 ☑☑☑☑☑☑☑☑

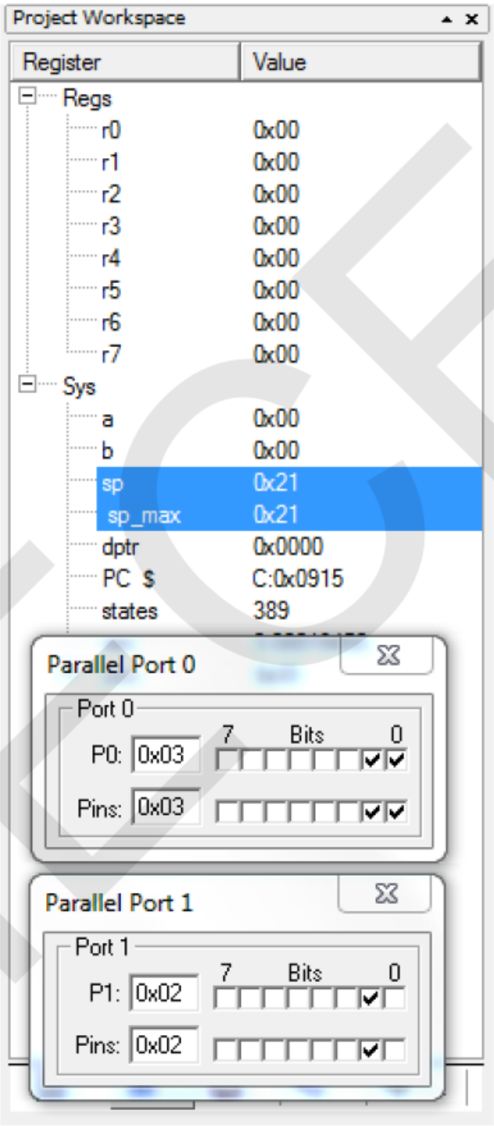
Pins: 0xFF   ☑☑☑☑☑☑☑☑

Port 1

P1: 0xFF   7 Bits 0 ☑☑☑☑☑☑☑☑

Pins: 0xFF   ☑☑☑☑☑☑☑☑

After Execution



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	0x21
sp_max	0x21
dptr	0x0000
PC $	C:0x0915
states	389

Port 0

P0: 0x03   7 Bits 0 □□□□□□☑☑

Pins: 0x03   □□□□□□☑☑

Port 1

P1: 0x02   7 Bits 0 □□□□□□☑□

Pins: 0x02   □□□□□□☑□

Experiment- 9

Count Positive and Negative Numbers

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

Program:

```
#include<reg51.h>  //To include the register set of 8051
void main()
{
unsigned long temp, array[]={0xff223344, 0xAA336699, 0x11223344, 0x33445566,
0x88AA3311};
unsigned char i, pos,neg;
CY=0;
for (i=0;i<5;i++)
    {
    temp=array[i]<<1;               //left shift to check MSB is 0 or 1 through CY
    if (CY==1) neg++;        `      //if CY=1 then increment negative number count
    else pos++;                     //if CY=0 then increment postive number count
    CY=0;                           //reset CY to ZERO
    }
  P0=neg;        //store the negative count at Port 0
  P1=pos;        //store the positive count at Port 1
 while(1);
}
```

Result:

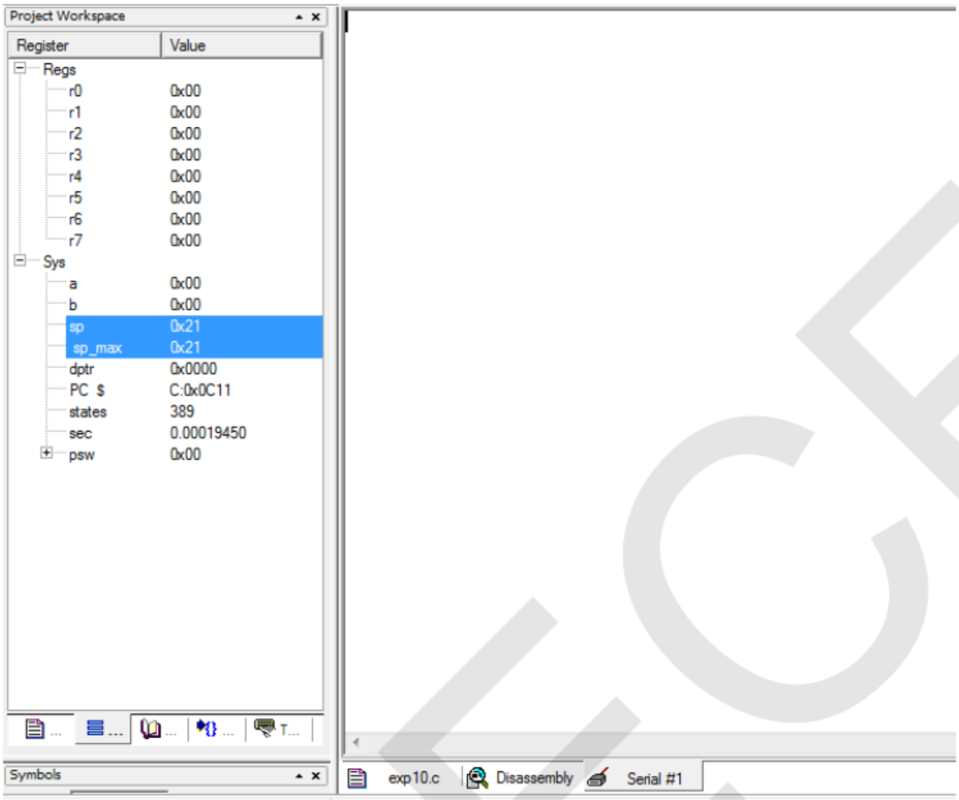
Written the code and found the number of negative and positive numbers in the data loaded in the array

Self-Learning Task-9:

Can you try to find the positive and negative numbers in an array of 8 32-bit elements?

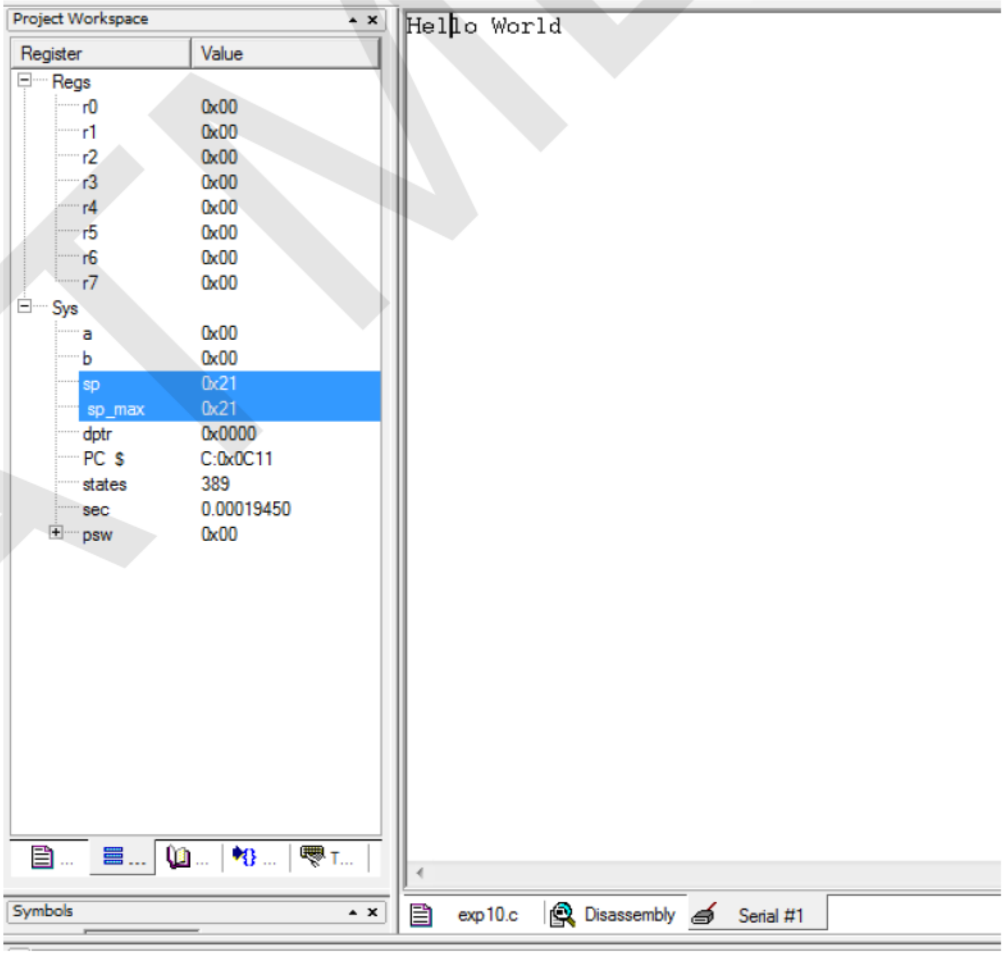
Output of Experiment-10

Before execution



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	0x21
sp_max	0x21
dptr	0x0000
PC $	C:0x0C11
states	389
sec	0.00019450
psw	0x00

After Execution



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	0x21
sp_max	0x21
dptr	0x0000
PC $	C:0x0C11
states	389
sec	0.00019450
psw	0x00

```
Hello World
```

Experiment- 10

Display Message in Serial monitor

Objective: Write a 8051 C program to display "Hello World" message (either in simulation mode or interface an LCD display).

Program:

```
#include <reg51.h>
#include <stdio.h>

void main(void)
 {
 SCON=0x52; // SET SERIAL PORT TO MODE 1
 TMOD=0x20; // TIMER1 MODE 2
 TH1=-13; // RELOAD COUNTER FOR 2400 BAUD RATE
 TR1=1; // START TIMER1
 // Write your code here

 printf("Hello World\n");
 }
```

Result:

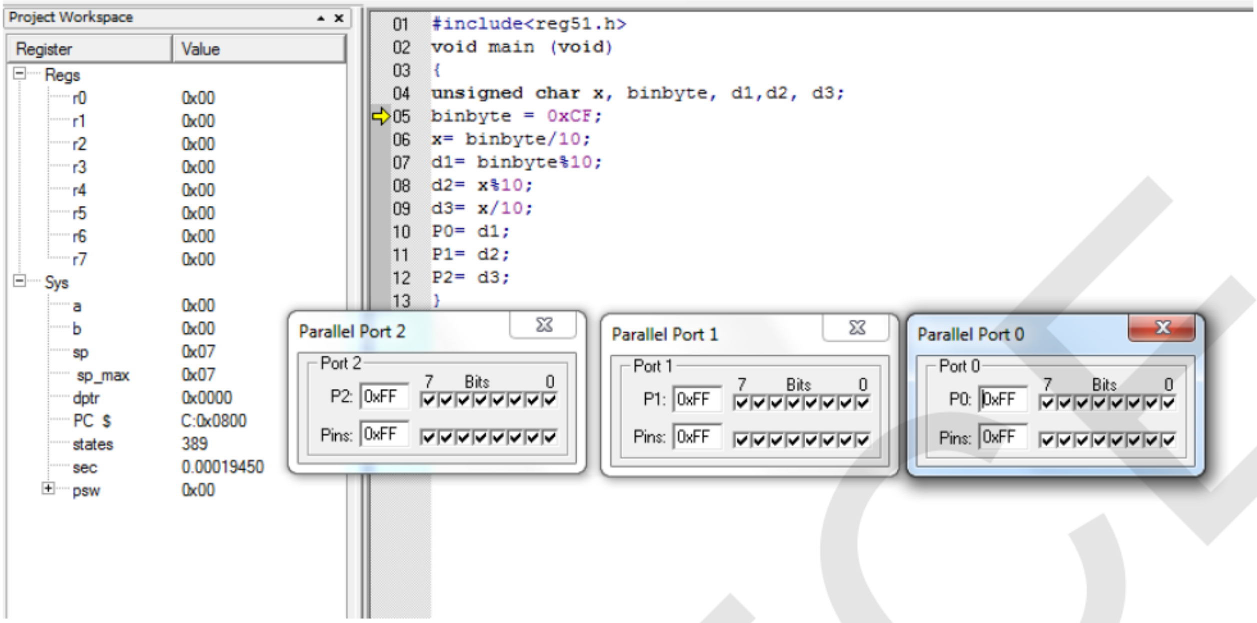
Written the code and successfully displayed the hello world message in serial monitor

Self-Learning Task-10:

Try to display Electronics and communication engineering in the serial monitor with interfacing LCD/Seven Segment Display

Output of Experiment-11

Before execution



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	0x07
sp_max	0x07
dptr	0x0000
PC $	C:0x0800
states	389
sec	0.00019450
psw	0x00

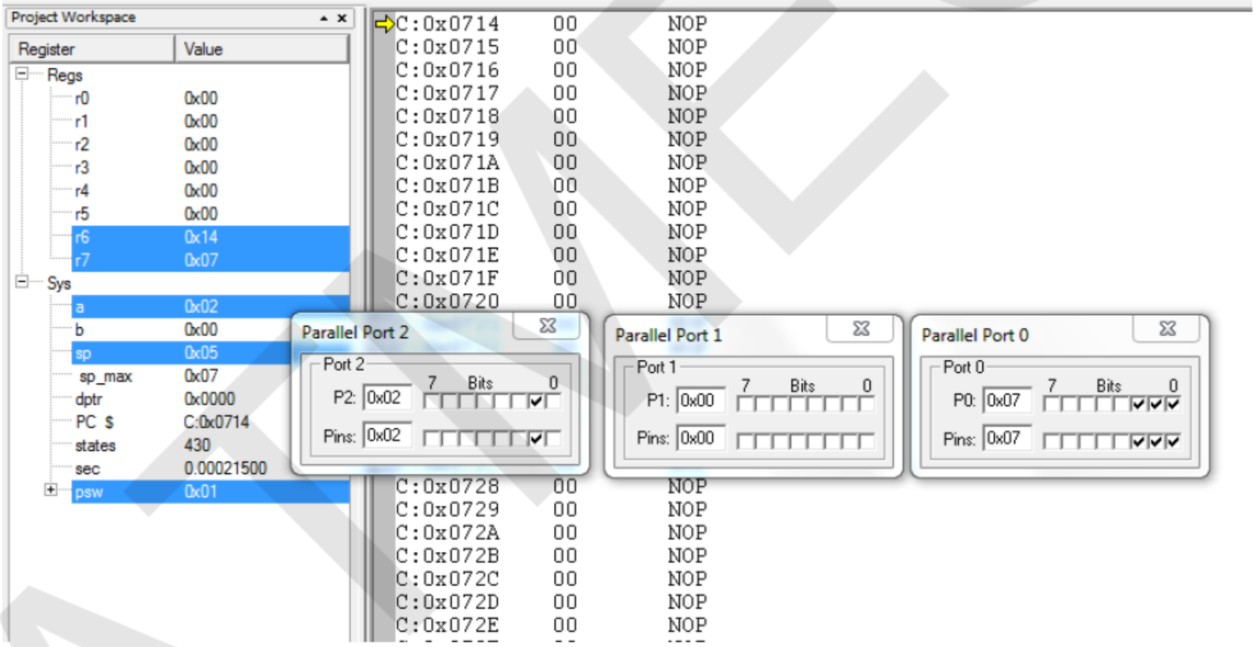
```
#include<reg51.h>
void main (void)
{
unsigned char x, binbyte, d1,d2, d3;
binbyte = 0xCF;
x= binbyte/10;
d1= binbyte%10;
d2= x%10;
d3= x/10;
P0= d1;
P1= d2;
P2= d3;
}
```

Parallel Port 2 — Port 2 — P2: 0xFF ☑☑☑☑☑☑☑☑ — Pins: 0xFF ☑☑☑☑☑☑☑☑

Parallel Port 1 — Port 1 — P1: 0xFF ☑☑☑☑☑☑☑☑ — Pins: 0xFF ☑☑☑☑☑☑☑☑

Parallel Port 0 — Port 0 — P0: 0xFF ☑☑☑☑☑☑☑☑ — Pins: 0xFF ☑☑☑☑☑☑☑☑

After Execution



Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x14
r7	0x07
Sys	
a	0x02
b	0x00
sp	0x05
sp_max	0x07
dptr	0x0000
PC $	C:0x0714
states	430
sec	0.00021500
psw	0x01

```
C:0x0714    00      NOP
C:0x0715    00      NOP
C:0x0716    00      NOP
C:0x0717    00      NOP
C:0x0718    00      NOP
C:0x0719    00      NOP
C:0x071A    00      NOP
C:0x071B    00      NOP
C:0x071C    00      NOP
C:0x071D    00      NOP
C:0x071E    00      NOP
C:0x071F    00      NOP
C:0x0720    00      NOP
C:0x0728    00      NOP
C:0x0729    00      NOP
C:0x072A    00      NOP
C:0x072B    00      NOP
C:0x072C    00      NOP
C:0x072D    00      NOP
C:0x072E    00      NOP
```

Parallel Port 2 — Port 2 — P2: 0x02 ☐☐☐☐☐☐☑☐ — Pins: 0x02 ☐☐☐☐☐☐☑☐

Parallel Port 1 — Port 1 — P1: 0x00 ☐☐☐☐☐☐☐☐ — Pins: 0x00 ☐☐☐☐☐☐☐☐

Parallel Port 0 — Port 0 — P0: 0x07 ☐☐☐☐☐☑☑☑ — Pins: 0x07 ☐☐☐☐☐☑☑☑

Experiment- 11

Conversion of Hex Number to Decimal Number

Objective: 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).

Program:

```
#include<reg51.h>
void main (void)
{
unsigned char x, binbyte, d1,d2, d3;
binbyte = 0xCF;
x= binbyte/10;
d1= binbyte%10;
d2= x%10;
d3= x/10;
P0= d1;
P1= d2;
P2= d3;
}
```

Result:

Written the code and successfully converted the hex number into decimal and displayed the data in Ports

Self-Learning Task-11:

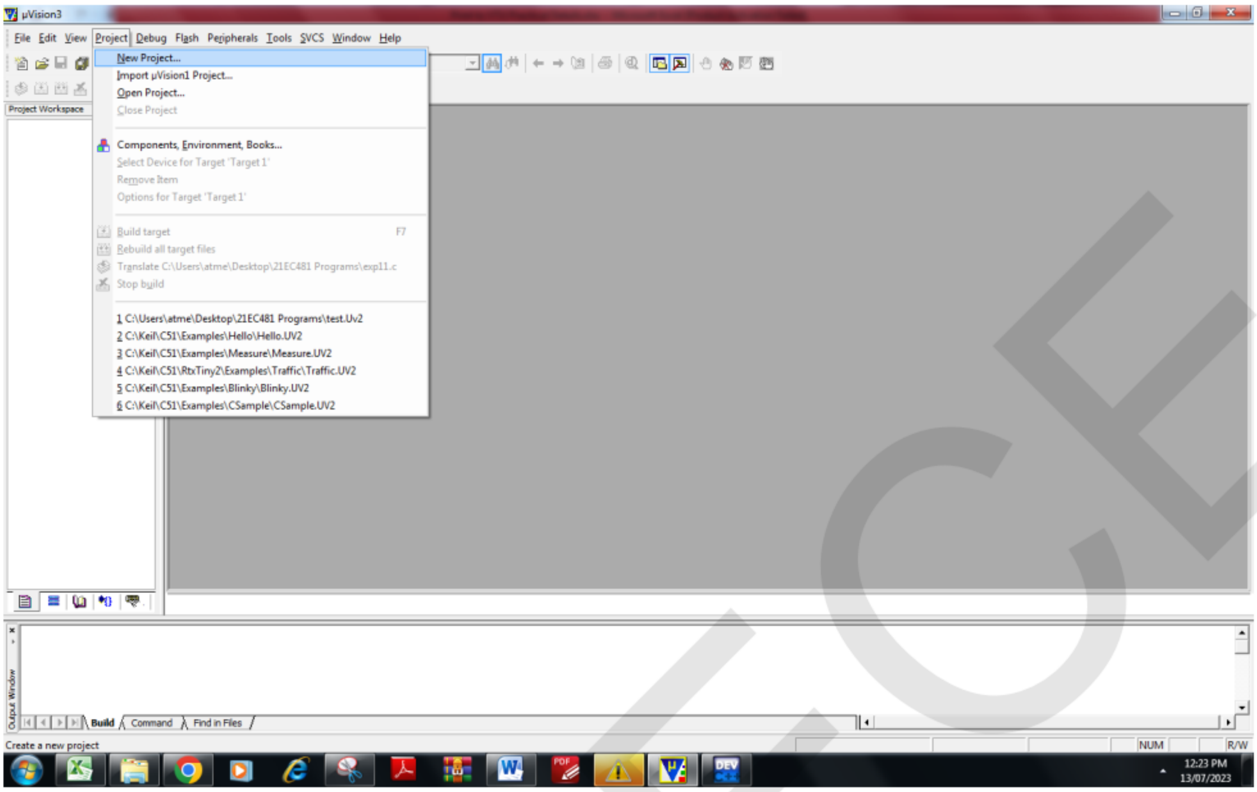
Try to write the code to convert decimal to hex number.

Viva Questions

- **What is Embedded C Programming? How is Embedded C different from C language?**
- **What do you understand by startup code?**
- **What is ISR?**
- **What is Void Pointer in Embedded C and why is it used?**
- **What are the differences between the const and volatile qualifiers in embedded C?**
- **What Is Concatenation Operator in Embedded C?**
- **What are the differences between Inline and Macro Function?**
- **What is the difference between Harvard Architecture and von Neumann Architecture?**
- **Why 8051 is called an 8-bit microcontroller?**
- **List the features of the 8051 microcontrollers?**
- **How Much on-chip RAM is available?**
- **With 12 MHz clock frequency how many instructions (of 1 machine cycle and 2 machine cycle) can execute per second?**
- **What are Special Function Registers (SFR)?**
- **What are the difference between bit addressable and byte address in microcontroller 8051?**
- **What are the types of interrupts in 8051?**
- **Are all the bits of flag register used in 8051?**

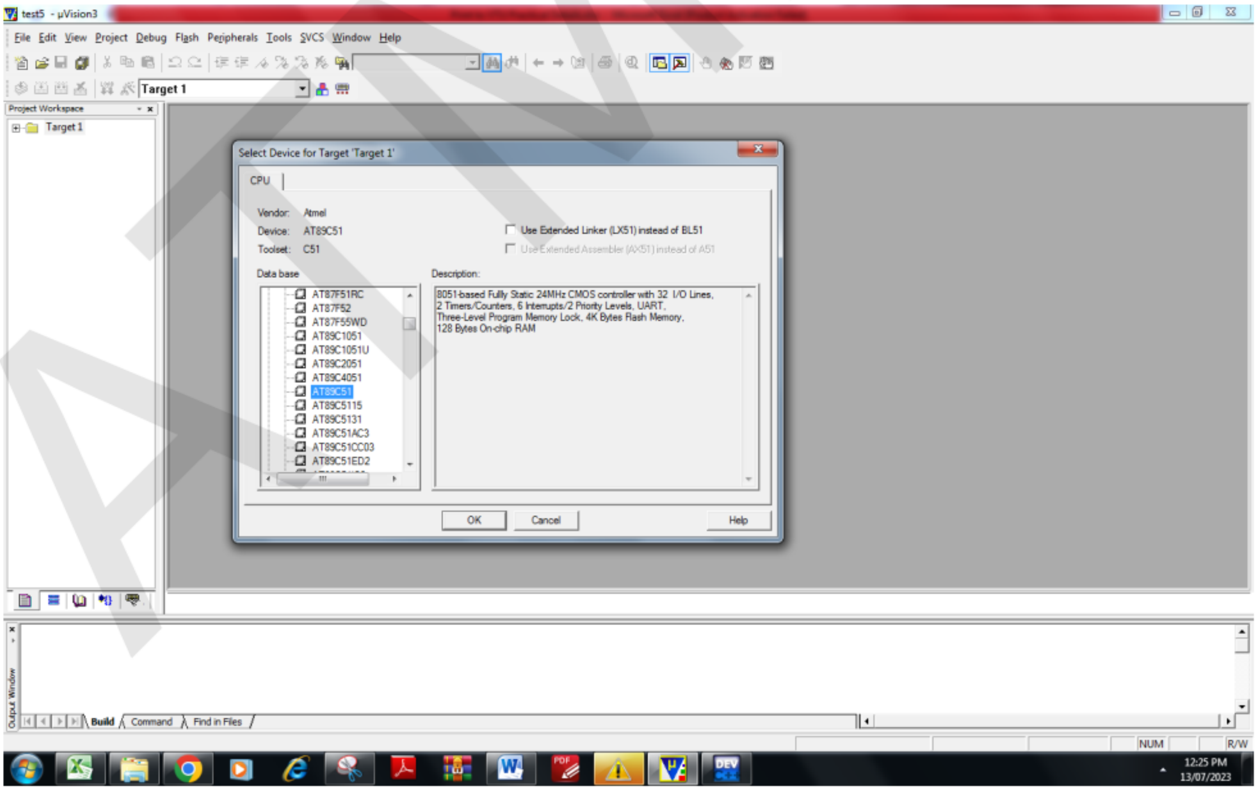
Project Work flow in Keil uVision 3

Create new project->


µVision3
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
New Project...
Import µVision1 Project...
Open Project...
Close Project
Components, Environment, Books...
Select Device for Target 'Target 1'
Remove Item
Options for Target 'Target 1'
Build target F7
Rebuild all target files
Translate C:\Users\atme\Desktop\21EC481 Programs\exp11.c
Stop build
1 C:\Users\atme\Desktop\21EC481 Programs\test.Uv2
2 C:\Keil\C51\Examples\Hello\Hello.UV2
3 C:\Keil\C51\Examples\Measure\Measure.UV2
4 C:\Keil\C51\RtxTiny2\Examples\Traffic\Traffic.UV2
5 C:\Keil\C51\Examples\Blinky\Blinky.UV2
6 C:\Keil\C51\Examples\CSample\CSample.UV2
Project Workspace
Output Window
Build Command Find in Files
Create a new project
NUM R/W
12:23 PM
13/07/2023

Give the project name, do not use extra characters while providing project name.

Select the target as AT89C51

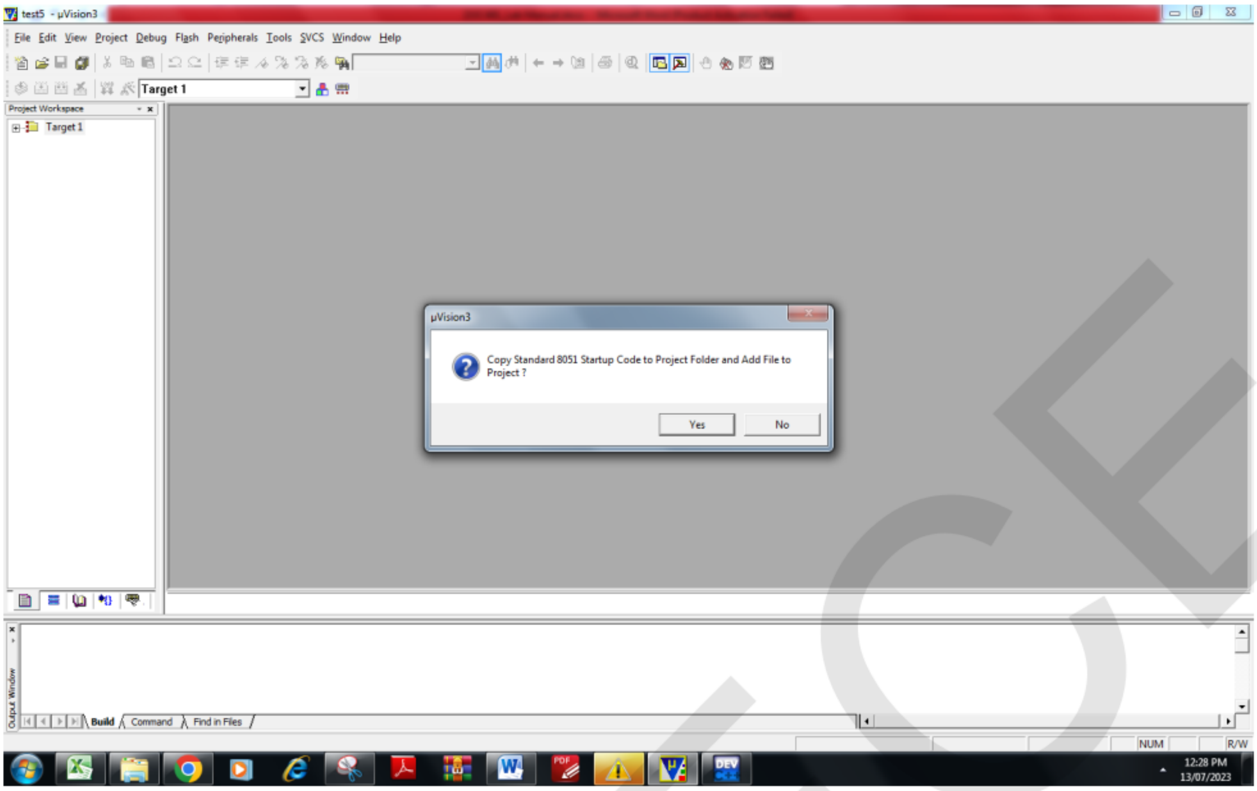


Vendor: Atmel
Device: AT89C51
Toolset: C51

□ Use Extended Linker (LX51) instead of BL51
□ Use Extended Assembler (AX51) instead of A51

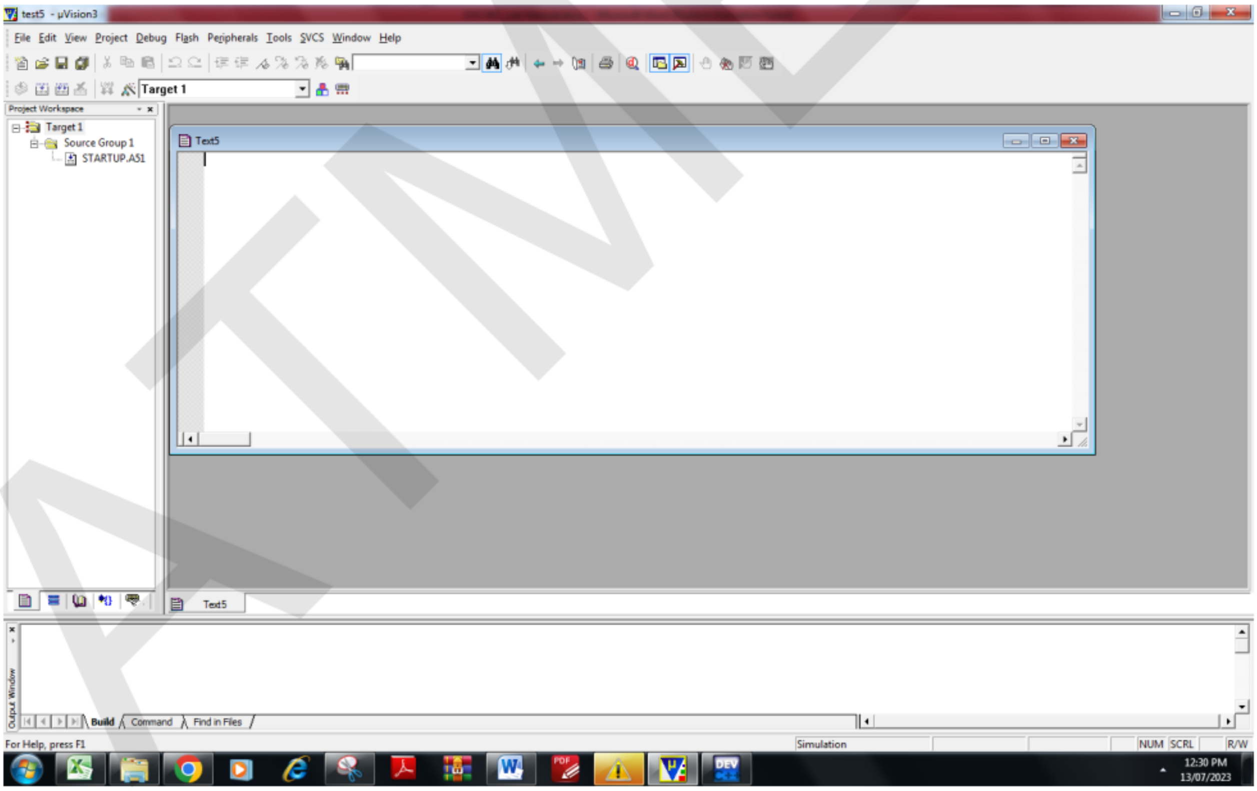
Description:
8051-based Fully Static 24MHz CMOS controller with 32 I/O Lines, 2 Timers/Counters, 6 Interrupts/2 Priority Levels, UART, Three-Level Program Memory Lock, 4K Bytes Flash Memory, 128 Bytes On-chip RAM

Copy the startup code into the source group

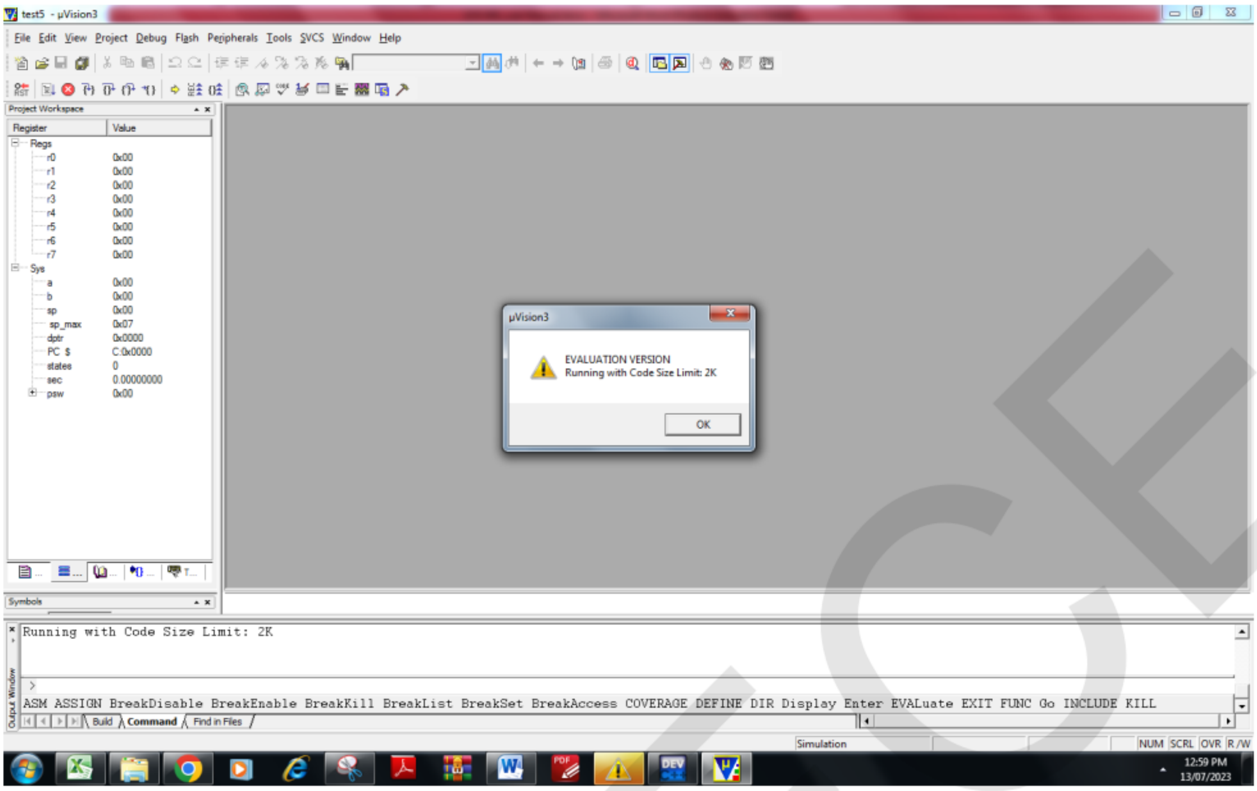


Copy Standard 8051 Startup Code to Project Folder and Add File to Project ?

Go to file new then, write the code and save with extension.c



Debug -> start debug session -> Click ok on Popup



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	0x00
sp_max	0x07
dptr	0x0000
PC $	C:0x0000
states	0
sec	0.00000000
psw	0x00

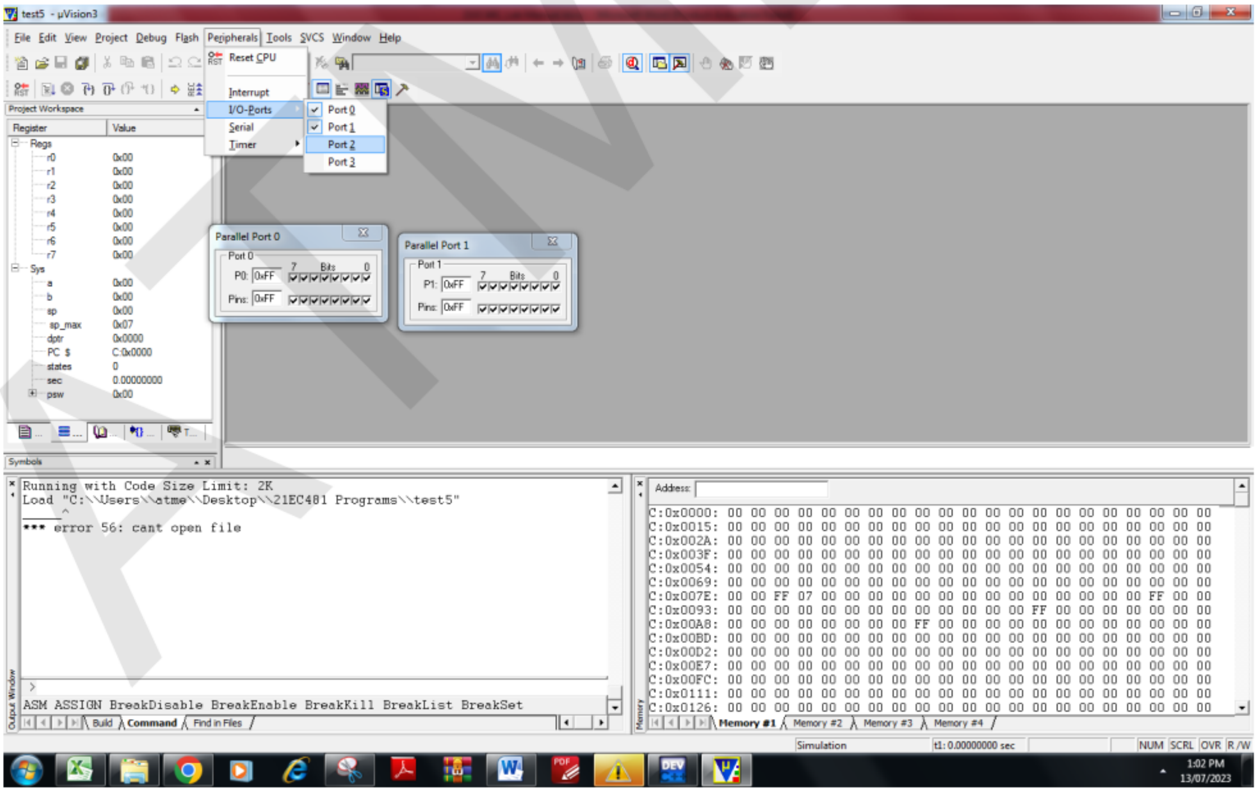
µVision3

EVALUATION VERSION
Running with Code Size Limit: 2K

OK

```
Running with Code Size Limit: 2K
>
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR Display Enter EVALuate EXIT FUNC Go INCLUDE KILL
```

Use F11 for step run, F5 for full run and see the results in the register set, Peripherals or memory window as per your requirement.



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	0x00
sp_max	0x07
dptr	0x0000
PC $	C:0x0000
states	0
sec	0.00000000
psw	0x00

Parallel Port 0 — Port 0 — P0: 0xFF — Pins: 0xFF

Parallel Port 1 — Port 1 — P1: 0xFF — Pins: 0xFF

```
Running with Code Size Limit: 2K
Load "C:\\Users\\atme\\Desktop\\21EC481 Programs\\test5"
     ^
*** error 56: cant open file
>
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet
```

```
C:0x0000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C:0x0015: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C:0x002A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C:0x003F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C:0x0054: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C:0x0069: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C:0x007E: 00 00 FF 07 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF 00 00
C:0x0093: 00 00 00 00 00 00 00 00 00 00 00 00 00 FF 00 00 00 00 00 00 00
C:0x00A8: 00 00 00 00 00 00 00 00 FF 00 00 00 00 00 00 00 00 00 00 00 00
C:0x00BD: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C:0x00D2: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C:0x00E7: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C:0x00FC: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C:0x0111: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C:0x0126: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```