QMP 7.1 D/F

Channabasaveshwara Institute of Technology
(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)
(**ISO 9001:2015 Certified Institution)**
NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka.

Department of Electronics & Communication Engineering

EMBEDDED C BASICS

Sub Code: 21EC481

B.E - IV Semester

Lab Manual 2022-'23

Name : _____

USN : _____

Batch :_____Section : _____

Channabasaveshwara Institute of Technology
(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)
(**ISO 9001:2015 Certified Institution)**
NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka.

Department of Electronics & Communication Engineering

EMBEDDED C BASICS

Version 1.0

JUNE 2023

Prepared by:

Ms. Anushree T
Assistant Professor
Dept. of ECE

Reviewed by:

Dr. Pradeep N.R
Associate Professor
Dept. of ECE

Approved by:

Dr. Sekar R
Professor & Head,
Dept. of ECE

VISION OF THE INSTITUTE

"To create centres of excellence in education and to serve the society by enhancing the quality of life through value based professional leadership"

MISSION STATEMENT OF THE INSTITUTE

- **To provide high quality technical and professionally relevant education in a diverse learning environment.**
- **To provide the values that prepare students to lead their lives with personal integrity, professional ethics and civic responsibility in a global society.**
- **To prepare the next generation of skilled professionals to successfully compete in the diverse global market.**
- **To promote a campus environment that welcomes and honors women and men of all races, creeds and cultures, values and intellectual curiosity, pursuit of knowledge and academic integrity and freedom.**
- **To offer a wide variety of off-campus education and training programmes to individuals and groups.**
- **To stimulate collaborative efforts with industry, universities, government and professional societies.**
- **To facilitate public understanding of technical issues and achieve excellence in the operations of the institute.**

QUALITY POLICY OF THE INSTITUTE

Our organization delights customers (students, parents and society) by providing value added quality education to meet the national and international requirements. We also provide necessary steps to train the students for placement and continue to improve our methods of education to the students through effective quality management system, quality policy and quality objectives.

VISION OF THE DEPARTMENT

"To Nurture and Develop Competent Electronics and Communication Engineering Skilled Professionals with values for the betterment of Society"

MISSION OF THE DEPARTMENT

- **To nurture the technical/professional/engineering and entrepreneurial skills for overall self and societal upliftment through co-curricular and extra-curricular events.**
- **To orient the Faculty/Student community towards the higher education, research and development activities.**
- **To create the Centres of Excellence in the field of electronics and communication in collaboration with industries/Universities by training the faculty through latest technologies.**
- **To impart quality technical education in the field of electronics and communication engineering to meet over the current/future global industry requirements.**

PROGRAM EDUCATIONAL OBJECTIVES

PEO1 : Provide technical solutions to real world problems in the areas of electronics and communication by developing suitable systems.

PEO2 : Pursue engineering career in Industry and/or pursue higher education and Research.

PEO3: Acquire and follow best professional, ethical practices in Industry relevant to the Society.

PEO4 : Communicate effectively and Develop an ability to work in team/Individually

PROGRAM SPECIFIC OBJECTIVES

PSO1: Build Analog and Digital Electronic systems for Multimedia Applications, VLSI and Embedded Systems in Interdisciplinary Research / Development.

PSO2: Design and Develop Communication Systems as per Real Time Applications and Current Trends.

Channabasaveshwara Institute of Technology
(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)
(**ISO 9001:2015 Certified Institution**)
NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka

Department of Electronics and Communication Engineering

Embedded C LABORATORY: Course Objectives/Outcomes/Mapping CO-PO/PSO's

COURSE OBJECTIVES:

The main objectives of this lab are:

1. Understand the basic programming of Microprocessor and microcontroller.
2. To develop the microcontroller-based programs for various applications.

COURSE OUTCOMES:

After completing this course the student could 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

MAPPING OF COURSE OUTCOMES WITH PROGRAM OUTCOMES:

		PO												PSO	
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**1**	**2**
CO	**1**	2	1	2	2	3	-	-	-	-	-	-	1	2	2
	2	3	2	2	2	3	-	-	-	-	-	-	1	2	3
	3	3	2	2	2	3	-	-	-	-	-	-	1	2	2
	4	3	2	3	3	3	-	-	-	-	-	-	1	3	3

3: high correlation, 2: medium correlation, 1: low correlation

Channabasaveshwara Institute of Technology
(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)
(**ISO 9001:2015 Certified Institution)**
NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka

PROGRAM OUTCOMES

PROGRAM OUTCOMES

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialisation for the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs.
4. **Conduct investigations of complex problems:** An ability to design and conduct scientific and engineering experiments, as well as to analyze and interpret data to provide valid conclusions
5. **Modern tool usage:** Ability to apply appropriate techniques, modern engineering and IT tools, to engineering problems.
6. **The engineer and society:** An ability to apply reasoning to assess societal, safety, health and cultural issues and the consequent responsibilities relevant to the professional engineering practice
7. **Environment and sustainability:** An ability to understand the impact of professional engineering solutions in societal and environmental contexts
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Ability to function effectively as an individual, and as a member or leader in a team, and in multidisciplinary tasks.
10. **Communication:** Ability to communicate effectively on engineering activities with the engineering community such as, being able to comprehend and write effective reports and design documentation, make effective presentations.
11. **Project management and finance:** An ability to apply knowledge, skills, tools, and techniques to project activities to meet the project requirements with the aim of managing project resources properly and achieving the project's objectives.
12. **Life-long learning:** Recognise the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Channabasaveshwara Institute of Technology

(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)
(**ISO 9001:2015 Certified Institution)**
NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka.

DEPARTMENT OF ELECTRONICS & COMMUNICATIONENGINEERING

SYLLABUS

B.E: Electronics & Communication Engineering / B.E: Electronics & Telecommunication Engineering NEP, Outcome Based Education (OBE) and Choice Based Credit System (CBCS)
(Effective from the academic year 2021 – 22)
SEMESTER – IV

Embedded C Basics LABORATORY

Laboratory Code	21EC481	**CIE Marks**	**50**
Number of Lecture Hours/Week	0:0:2:0	**SEE Marks**	**50**
RBT Level	**L2, L3**	**Exam Hours**	**03**

CREDITS – 01

Laboratory Experiments

Sl.No	Experiments
	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 or 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).

Conduct of Practical Examination:

1. All laboratory experiments are to be included for practical examination.
2. Students are allowed to pick one experiment from the lot.
3. Strictly follow the instructions as printed on the cover page of answer script for breakup of marks.
4. Change of experiment is allowed only once and Marks allotted to the procedure part to be made zero.

Instructions to the Candidates

General Lab Guidelines:

- Conduct yourself in a responsible manner at all times in the laboratory. Intentional misconduct will lead to the exclusion from the lab.
- Do not wander around, or distract other students, or interfere with the laboratory experiments of other students.
- Read the handout and procedures before starting the experiments. Follow all written and verbal instructions carefully. If you do not understand the procedures, ask the instructor or teaching assistant.
- Attendance in all the labs is mandatory, absence permitted only with prior permission from Class teacher.
- The workplace has to be tidy before, during and after the experiment.
- Do not eat food, drink beverages or chew gum in the laboratory.
- Every student should know the location and operating procedures of all Safety equipment including First Aid Kit and Fire extinguisher.

DO'S:-

- Uniform and ID card are must.
- Strictly follow the procedures for conduction of experiments.
- Records have to be submitted every week for evaluation.
- Chairs and stools should be kept under the workbenches when not in use.
- After the lab session, switch off every supply, disconnect and disintegrate the experiments and return the components.
- Keep your belongings in designated area.
- Never use damaged instruments, wires or connectors. Hand these parts to the instructor/ teaching assistant.
- Sign the log book when you enter/leave the laboratory.

DONT'S:-

- Don't touch open wires unless you are sure that there is no voltage. Always disconnect the plug by pulling on the connector body not by the cable. Switch off the supply while you make changes to the experiment.
- Don't leave the experiment table unattended when the experimental setup supply is on.
- Students are not allowed to work in laboratory alone or without presence of the teaching staff/ instructor.
- No additional material should be carried by the students during regular labs.
- Avoid stepping on electrical wires or any other computer cables.

CONTENTS

Sl.No.	Name of the Experiment	Page No.
1.	Introduction to Embedded C Basics	**1-16**
	Programme's	
2.	8051 C program to multiply two 16 bit binary numbers.	17
3.	8051 C program to find the sum of first 10 integer numbers.	18
4.	8051 C program to find factorial of a given number.	19
5.	8051 C program to add an array of 16 bit numbers and store the 32 bit result in internal RAM	20
6.	8051 C program to find the square of a number (1 to 10) using look-up table.	21
7.	8051 C program to find the largest/smallest number in an array of 32 numbers	22
8.	8051 C program to arrange a series of 32 bit numbers in ascending/descending order	23
9.	8051 C program to count the number of ones and zeros in two consecutive memory locations.	25
10.	8051 C program to scan a series of 32 bit numbers to find how many are negative.	27
11.	8051 C program to display "Hello World" message (either in simulation mode or interface an LCD display).	29
12.	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).	31
13.	**ADDITIONAL PROGRAMS FOR PRACTICE**	32-34
14.	**VIVA Questions**	35-38
15.	**Question Bank**	39
16.	**REFERENCES**	40
16.	**APPENDIX**	41-47

INDEX PAGE

Sl. No	Name of the Experiment	Date	Manual Marks (Max . 20)	Record Marks (Max. 10)	Signature (Student)	Signature (Faculty)
1.	8051 C program to multiply two 16 bit binary numbers.					
2.	8051 C program to find the sum of first 10 integer numbers.					
3.	8051 C program to find factorial of a given number.					
4.	8051 C program to add an array of 16 bit numbers and store the 32 bit result in internal RAM					
5.	8051 C program to find the square of a number (1 to 10) using look-up table.					
6.	8051 C program to find the largest/smallest number in an array of 32 numbers					
7.	8051 C program to arrange a series of 32 bit numbers in ascending/descending order					
8.	8051 C program to count the number of ones and zeros in two consecutive memory locations.					
9.	8051 C program to scan a series of 32 bit numbers to find how many are negative.					
10.	8051 C program to display "Hello World" message (either in simulation mode or interface an LCD display).					
11.	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).					
Average						

Note: If the student fails to attend the regular lab, the experiment has to be completed in the same week. Then the manual/observation and record will be evaluated for 50% of maximum marks.

Channabasaveshwara Institute of Technology
(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)
(**ISO 9001:2015 Certified Institution)**
NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka.

DEPARTMENT OF ELECTRONICS & COMMUNICATIONENGINEERING

Scheme of evaluation for Embedded C Basics Laboratory:

Lab Internal:

I. **Observation – 5M**
(Successful Wording/Algorithm/flowchart-1M, Successful Program verification – 1M, Successful Program Execution – 1M, Record Initial and Indexing – 2M)

II. **Record – 10M**
(Aim & Apparatus – 1M, Theory – 3M, Algorithm/flowchart – 2M(each experiment should have at least one flowchart, Calculations, Input/Output observations & Result – 1M, Daily Performance 3M)

III. **Program Execution – 10M**
(It is compulsory for each student to create their own Microcontroller project for personal use based on 8051 and show the results for different cases)

IV. **Internal Assesment No.: 01 – 25M**
(Aim, Apparatus – 2M, Program – 15M (C Program/code – 10M, Relevant Comments – 2M, Algorithm/flow chart – 3M), Calculations, Input/Output observations & Result – 5M, Performance – 3M)

V. **Internal Assesment No.: 02 – 25M**
(Aim, Apparatus – 2M, Program – 15M (C Program/code – 10M, Relevant Comments – 2M, Algorithm/flow chart – 3M), Calculations, Input/Output observations & Result – 5M, Performance – 3M)

VI. **Internal Assesment No.: 03 – 25M**
(Aim, Apparatus – 2M, Program – 15M (C Program/code – 10M, Relevant Comments – 2M, Algorithm/flow chart – 3M), Calculations, Input/Output observations & Result – 5M, Performance – 3M)

Internal Assessment (I.A) = [(I.A-I) + (I.A-II) + (I.A-III)]/ 03
Lab I.A = I+II+III+I.A

Lab External:

I. **Writeup – 10M**
(Aim– 2M, Theory – 3M, Algorithm/flowchart – 5M)

II. **Program Execution – 70M**
(Program/Code Output – 65M, Comments – 3M, Optimization– 2M, (Partial output – 35M, No Output – 00M)

III. **Results – 05M**
(Identifying & Showing the inputs and outputs – 2M and/or theoretical calculations – 2M, Output Verification – 1M

IV. **Viva – 15M**

INTRODUCTION

Introduction to Keil:-

Embedded system means some combination of computer hardware and programmable software which is specially designed for a particular task like displaying message on LCD. It involves hardware (8051 microcontroller) andsoftware (the code written in assembly language). Some real life examples of embedded systems may involve ticketing machines, vending machines, temperature controlling unit in air conditioners etc. Microcontrollers are nothing without a Program in it.

One of the important part in making an embedded system is loading the software/program we develop into the microcontroller. Usually it is called "burning software" into the controller. Before "burning a program" into a controller, we must do certain prerequisite operations with the program. This includes writing the program in assembly language or C language in a text editor like notepad, compiling the program in a compiler and finally generating the hex code from the compiled program. Earlier people used different software"s /applications for all these 3 tasks. Writing was done in a text editor like notepad/ WordPad, compiling was done using a separate software (probably a dedicated compiler for a particular controller like 8051),converting the assembly code to hex code was done using another software etc. It takes lot of time and work to do all these separately, especially when the task involves lots of error debugging and reworking on the source code.

The µVision IDE is the easiest way for most developers to create embedded applications using the Keil development tools. The new Keil **µVision4** IDE has been designed to enhance developer's productivity, enabling faster, more efficient program development.

Keil MicroVision is a free software which solves many of the main points for an embedded program developer. This software is an integrated development environment (IDE), which integrated a text editor to write programs, a compiler and it will convert your source code to hex files too. µVision4 introduces a flexible window management system, enabling us to drag and drop individual windows anywhere on the visual surface including support for **Multiple Monitors**.

Embedded Systems Vs General Computing Systems

General Purpose System	Embedded System
A system which is a combination of generic hardware and General Purpose Operating System for executing a variety of applications	A system which is a combination of special purpose hardware and embedded OS for executing a specific set of applications
Contain a General Purpose Operating System (GPOS)	May or may not contain an operating system for functioning
Applications are alterable (programmable) by user (It is possible for the end user to re-install the Operating System, and add or remove user applications)	The firmware of the embedded system is pre-programmed and it is non-alterable by end-user (There may be exceptions for systems supporting OS kernel image flashing through special hardware settings)
Performance is the key deciding factor on the selection of the system. Always 'Faster is Better'	Application specific requirements (like performance, power requirements, memory usage etc) are the key deciding factors
Less/not at all tailored towards reduced operating power requirements, options for different levels of power management.	Highly tailored to take advantage of the power saving modes supported by hardware and Operating System
Response requirements are not time critical	For certain category of embedded systems like mission critical systems, the response time requirement is highly critical
Need not be deterministic in execution behavior	Execution behavior is deterministic for certain type of embedded systems like 'Hard Real Time' systems

source fr...

C51 Development Tools

Keil development tools for the 8051 microcontroller family support every level of developer from the professional applications engineer to the student just learning about embedded software development. The industry-standard Keil C Compilers, Macro Assemblers, Debuggers, Real-time Kernels, and Single-board Computers support ALL 8051-compatible derivatives and help you get your projects completed on schedule. The C51 development tool chains are designed for the professional software developer, but any level of programmer can use them to get the most out of the 8051 microcontroller architecture. With the C51 tools, embedded applications can be generated for virtually every 8051 variant. Refer to the µVision Device Database for a list of currently supported microcontrollers.

This introduction includes a brief explanation of the:

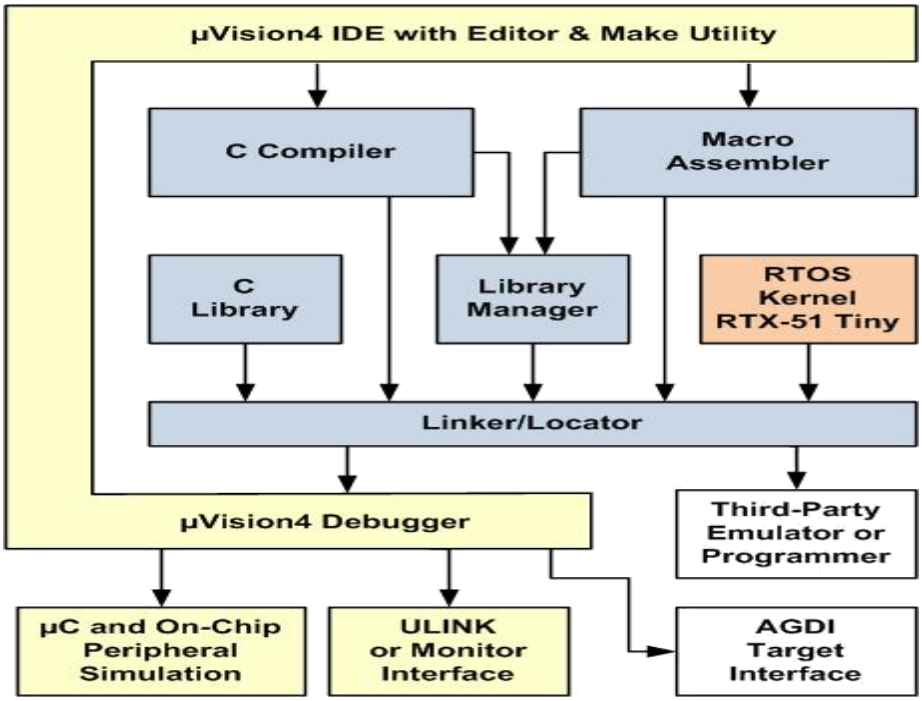
- Software Development Cycle that describes the steps and tools involved tocreate a project.
- Development Tools that describes the major features of the Keil C51development tools including the µVision IDE and Debugger.
- Folder Structure that describes the default location of µVision and the C51tool chain installation

Development Tools

The Keil C51 development tools offer numerous features and advantages that help you to develop embedded applications quickly and successfully. Find out more about the supported devices and the possible tool combinations available for the different 8051 variants.

The following block diagram shows the components involved in the build process.

The µVision IDE is a window-based software development tool that combines project management and a rich-featured editor with interactive error correction, option setup, make facility, and on-line help. Use µVision to create source files andorganize them into a project that defines your target application.



µVision Integrated Development Environment (IDE)

C COMPILER: The Keil Cx51 Compiler is a full ANSI implementation of the C programming language and supports all standard features of the C language. In addition, numerous extensions have been included to directly support the 8051 and extended 8051 architecture.

Macro Assembler: The Keil Ax51 Macro Assembler supports the complete instruction set of the 8051 and all 8051 derivatives.

Library Manager: The LIBx51 Library Manager allows you to create the object library from object files created by the compiler and assembler. Libraries are specially formatted, ordered program collections of object modules that may be used by the linker at a later time. When the linker processes a library, only those object modules necessary to create the program are used.

Linker/Locater: The Lx51 Linker/Locater creates the final executable 8051 program and combines the object files created by the compiler or assembler, resolves external and public references, and assigns absolute addresses. In addition, it selects and includes the appropriate run-time library modules.

µVision Debugger:

The µVision Debugger is ideally suited for fast and reliable program debugging. The debugger includes a high-speed simulator capable of simulating an entire 8051 system including on-chip peripherals and external hardware.

The µVision Debugger provides several ways to test programs on target hardware:

Use the Keil ULINK USB-JTAG adapter for downloading and testing your program.

- Install a target monitor on your target system and download your program using the built-in monitor interface of the µVision Debugger.
- Use the Advanced GDI interface to attach and use the µVision Debugger front end with your target system.

RTOS Kernel:

The RTOS Kernel, describes the advantages of using a real-time kernel like the Keil RTX51 Tiny in embedded systems.

Start Debugging:

µVision provides several ways to invoke debugging commands:

- Commands used from the menu Debug or the **Debug Toolbar**
- Commands entered manually in the Command Window
- Commands available from the **Context Menu** of the **Editor** or **Disassembly** window
- Debug Functions executed from an initialization file

Start the Debugger:

Use the **Start/Stop Debug Session** button from the **Debug Toolbar** to start or stop a debugging session. The current instruction or high-level statement (the one about to execute) is marked with a yellow arrow. For each step-command, the arrow moves to reflect the new current line or instruction. Depending on the **Options for Target — Debug** configuration, µVision loads the application program and runs the startup code (**Run to main ()**).

µVision saves the editor screen layout and restores the screen layout of the last debug session. When program execution stops, µVision opens an Editor window with the source text or shows MCU instructions in the Disassembly Window.

Execute Commands:

- Run the program to the next break point, or type GO in the Command Line
- Halt the program, or press **Esc** while in the **Command Line**
- Click Reset from the Debug Toolbar or from the Debug — Reset CPU Menu or type RESET in the Command Line to reset the CPU

Single-Stepping Commands:

- To step through the program and into function calls. Alternatively, you can enter TSTEP in the Command Line, or press F11
- To step over the program and over function calls. Alternatively, you can enter **PSTEP** in the **Command Line**, or press **F10**
- To step out of the current function. Alternatively, you can enter **OSTEP** in the **Command Line**, or press **Ctrl+F11**

On-Chip Peripherals:

There are a number of techniques you must know to create programs that can use the various on-chip peripherals and features of the 8051 family. Use the code examples provided here to get started working with the 8051.

There is no single standard set of on-chip peripherals for the 8051 family. Instead, 8051 chip vendors use a wide variety of on-chip peripherals to distinguish their parts from each other. The code examples demonstrate how to use the peripherals of a particular chip or family. Be aware that there are more configuration options available than are presented in this text.

Follow the links to the on-chip peripherals:

Header Files - use the include files to define peripheral registers of the device in use

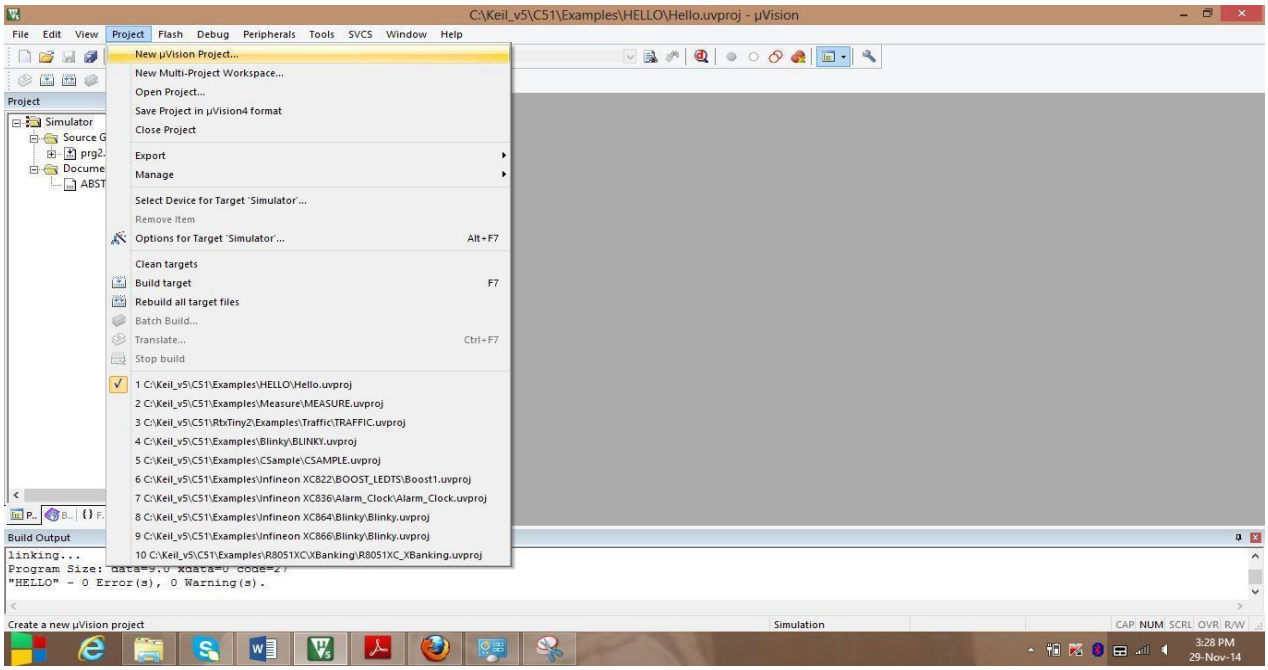
- **Startup Code** - initializes the microcontroller and transfers control to the **main** function
- **Special Function Registers** - explains how to use Special Function Registers (SFRs)
- **Register Bank**s **-** explains how to use Register Banks
- **Interrupt Service Routines** - lists the different interrupt variants on 8051 devices
- **Interrupt Enable Registers** - shows how to enable the interrupts
- **Parallel Port I/O** - explains how to use standard I/O ports
- **Timers/Counters** - explains standard timers and counters
- **Serial Interface** - explains the implementation of serial UART communication
- **Watchdog Timer -** use a watchdog timer to recover from hardware or software failures
- **D/A Converter** - convert a digital output voltage to an analog output value
- **A/D Converter** - convert an analog input voltage to a digital value
- **Power Reduction Modes** - put the device into IDLE or POWER DOWN mode

Startup Code

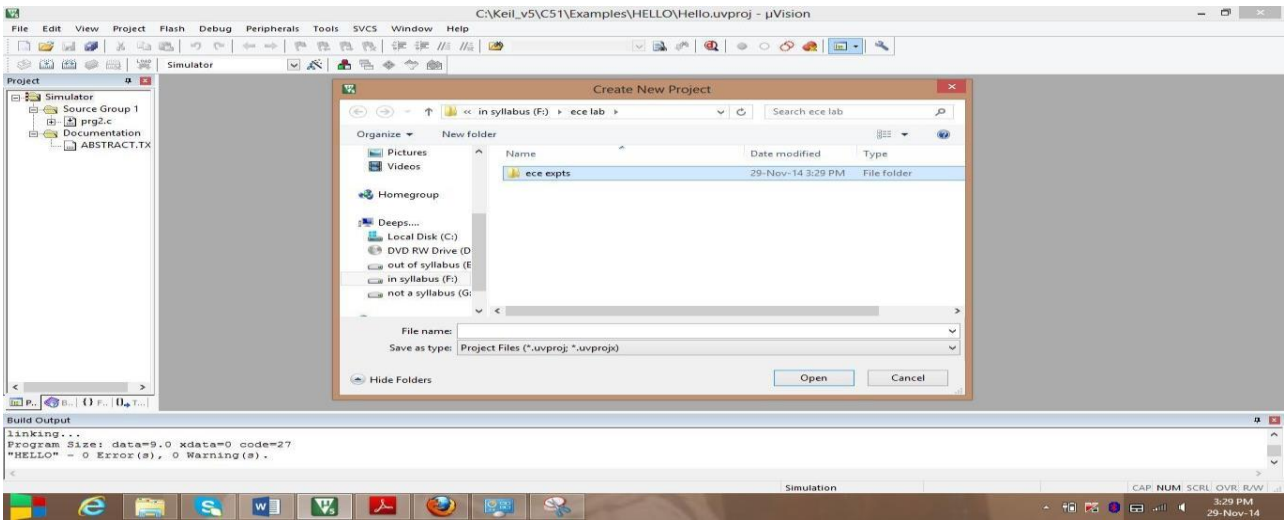
Startup Code is executed immediately upon RESET of the target system and performs the following operations:

- Depending on the device variant, device specific features are configured.
- Clears data memory (optionally).
- Initializes the reentrant stack and re-entrant stack pointer (optionally).
- Initializes the 8051 hardware stack pointer.
- Transfers control to the variable initialization code or to the main C function

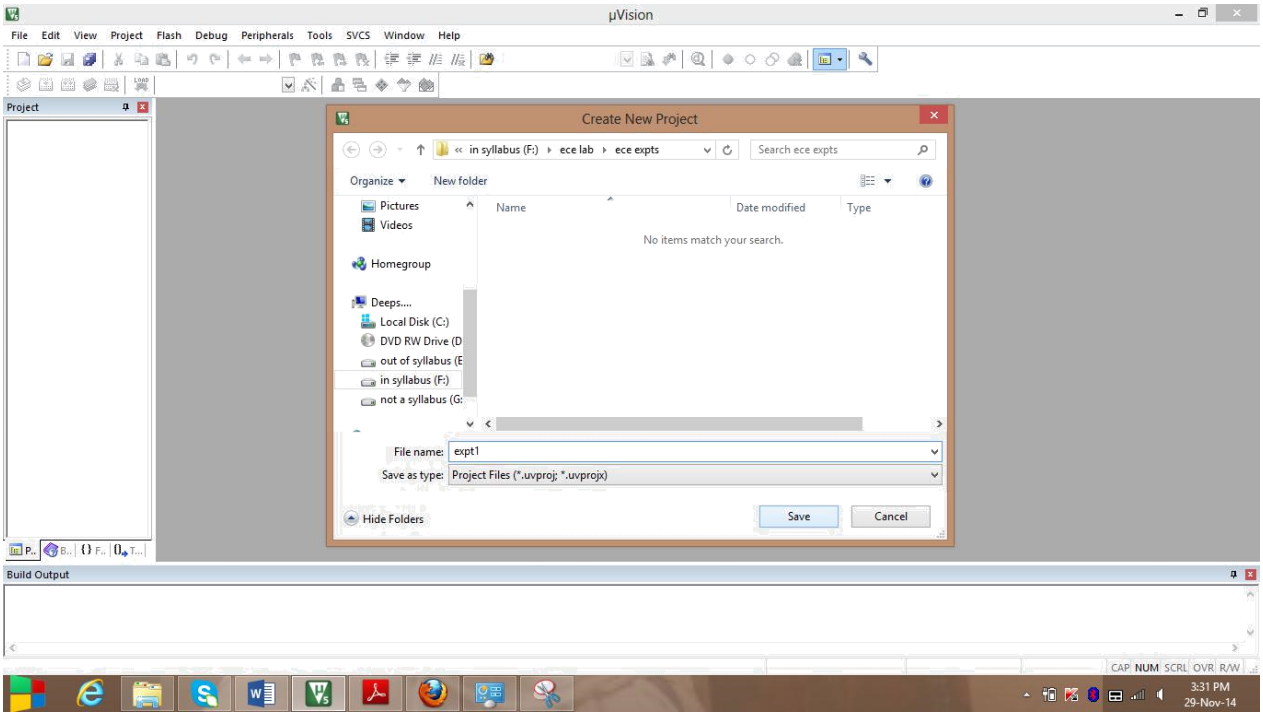
Procedure:-



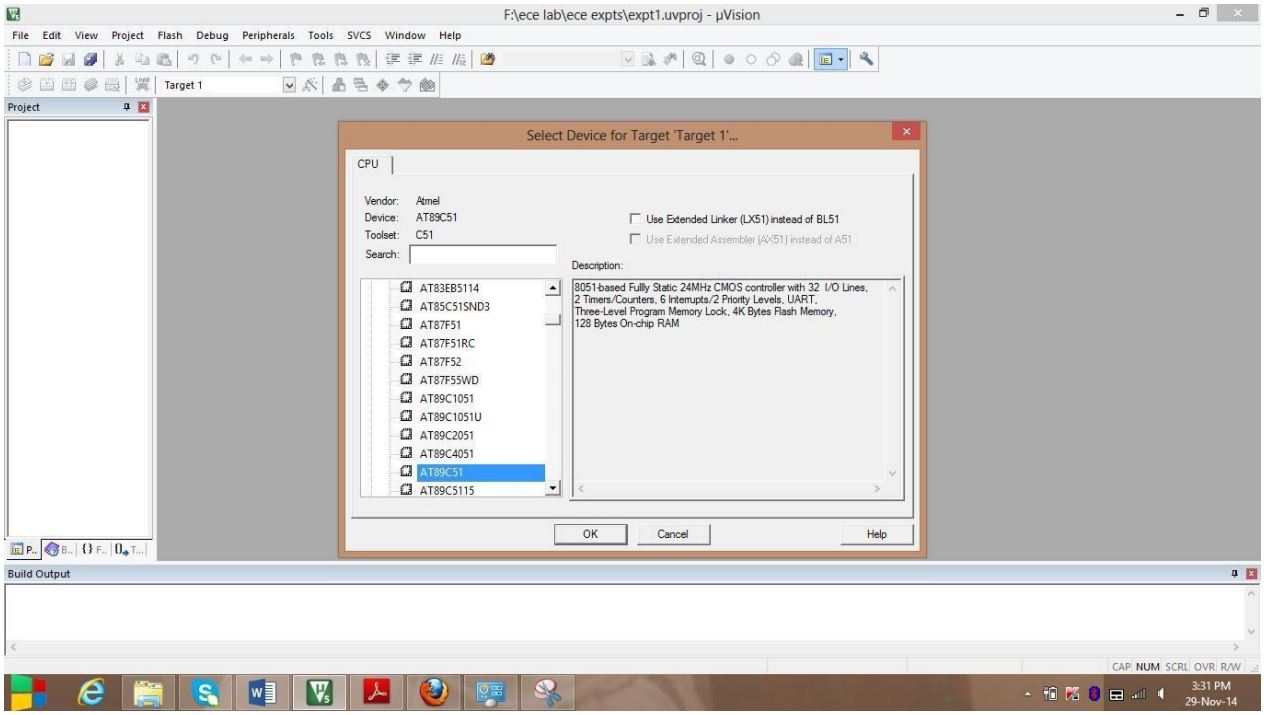
Create new u vision project



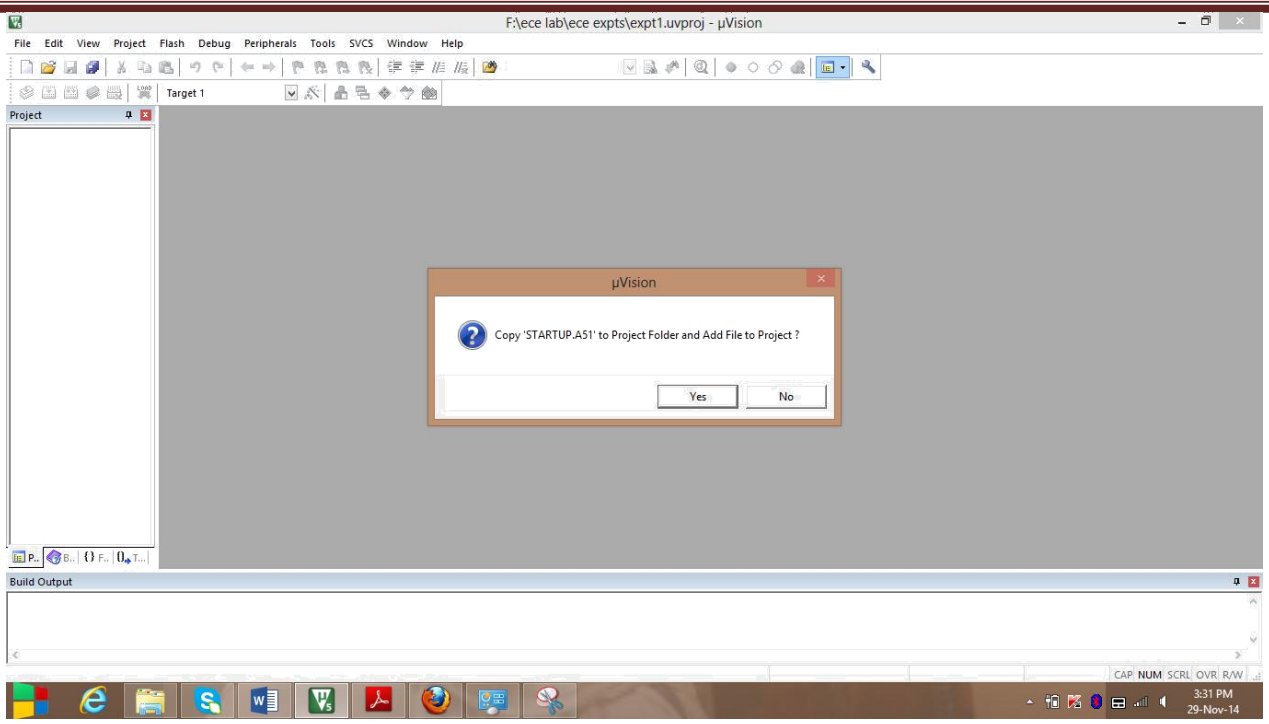
Select the folder (newly created) to save the project



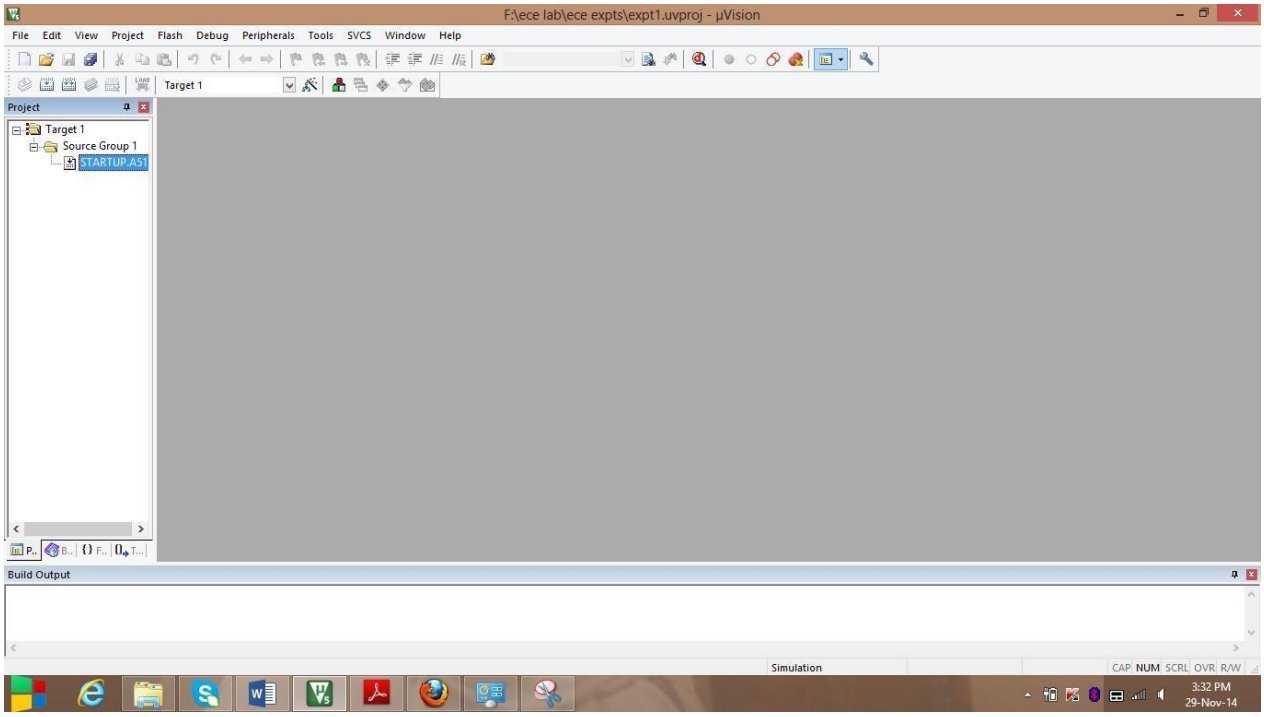
Save the project



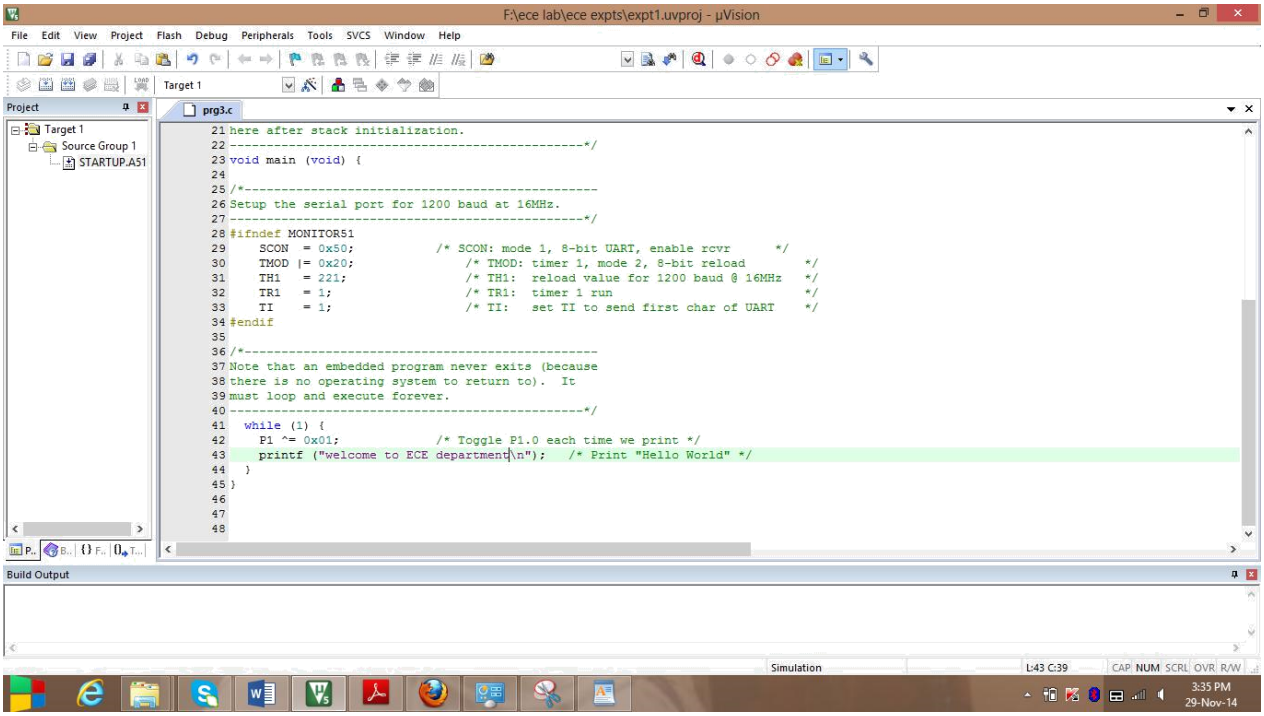
Select the vendor "Atmel" and device "AT89C51"



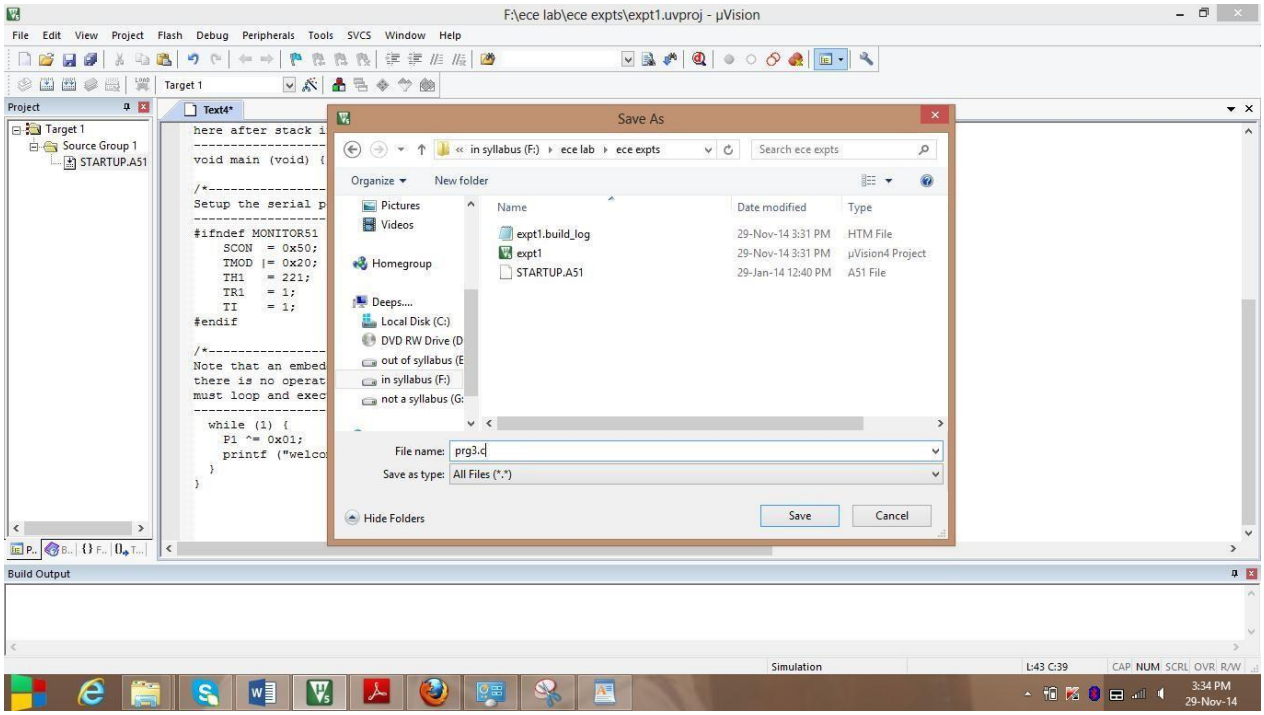
Addition of STARTUP.A51 to project folder



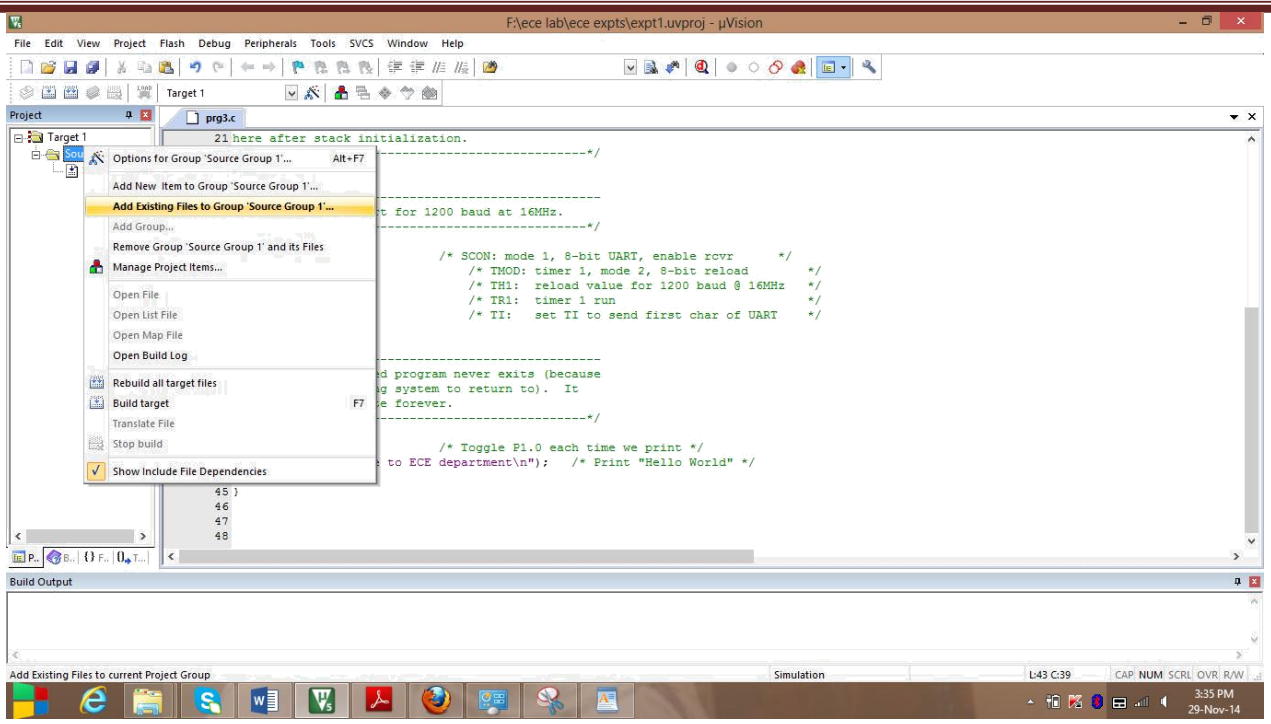
STARTUP.A51 is added



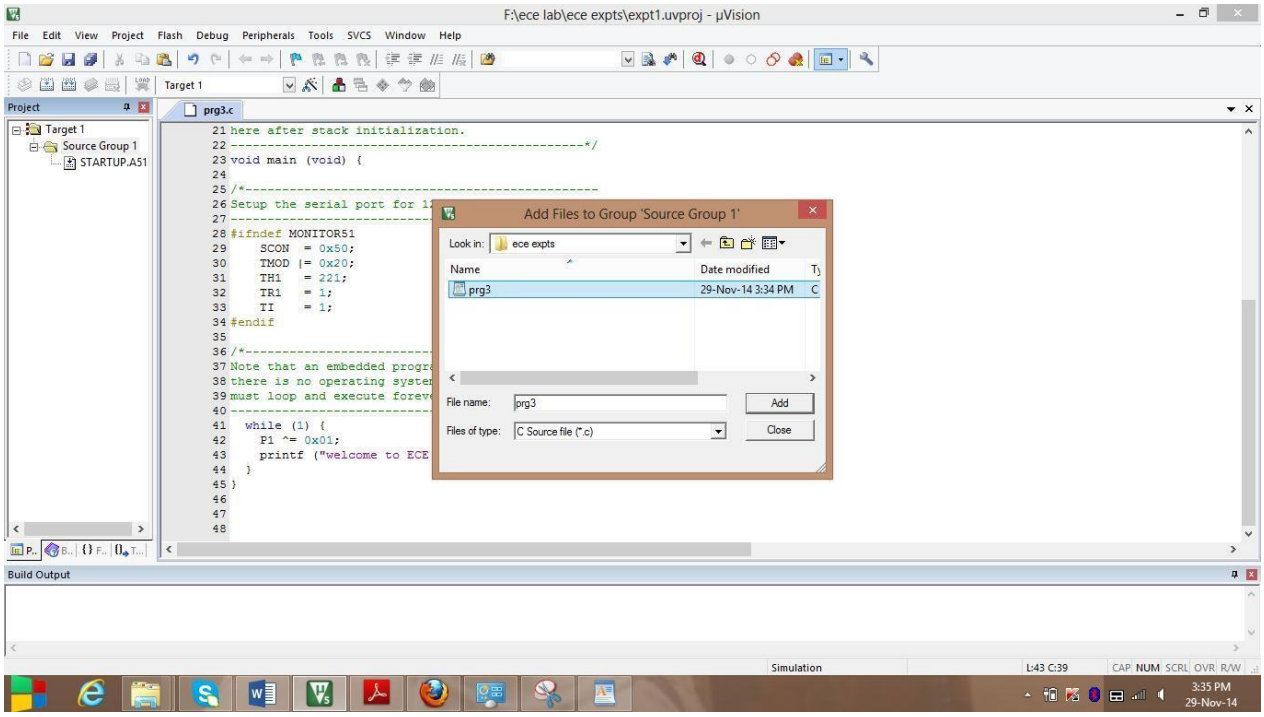
The program to print "welcome to ECE department" is written



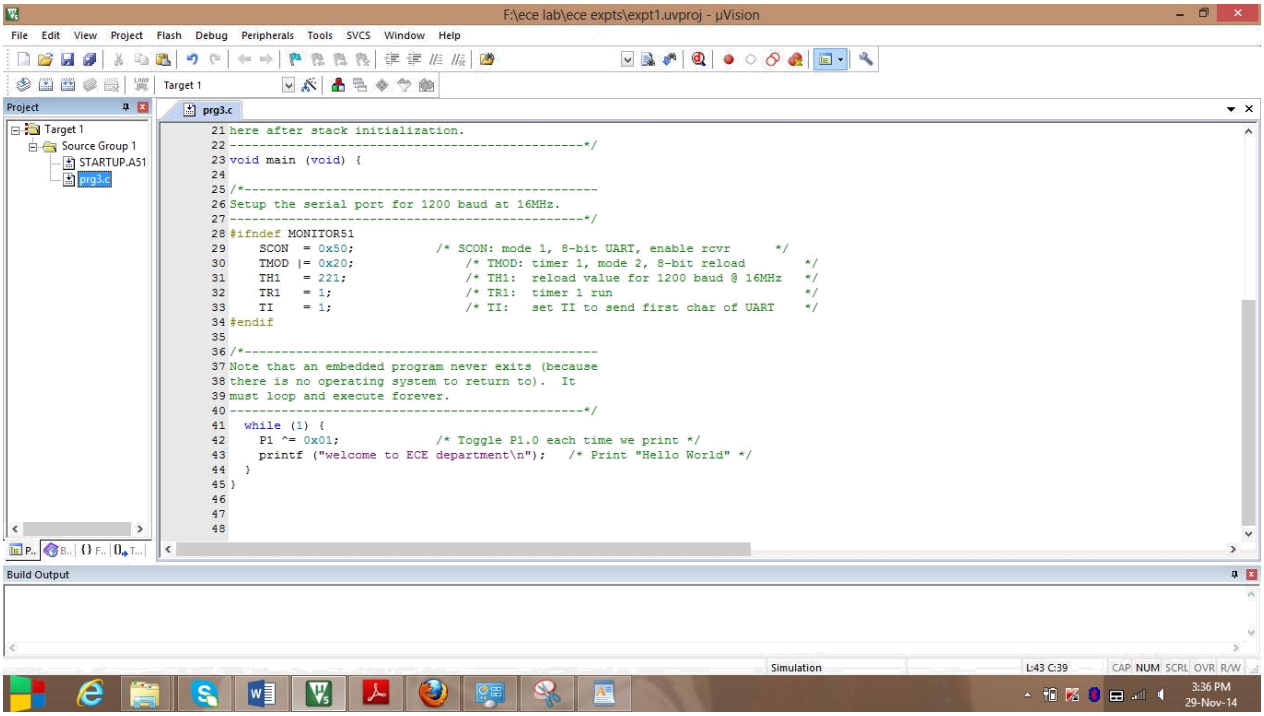
The program is saved as prg3.c



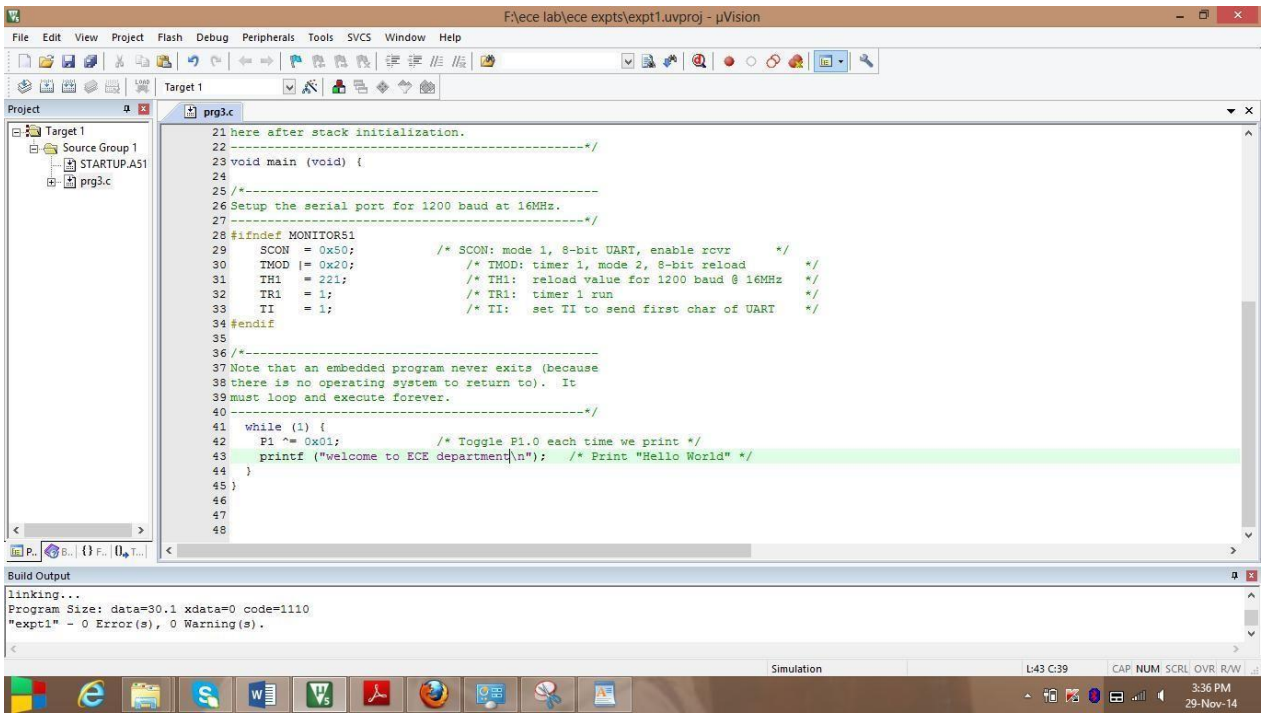
Prg3.c is to be added to Source Group1



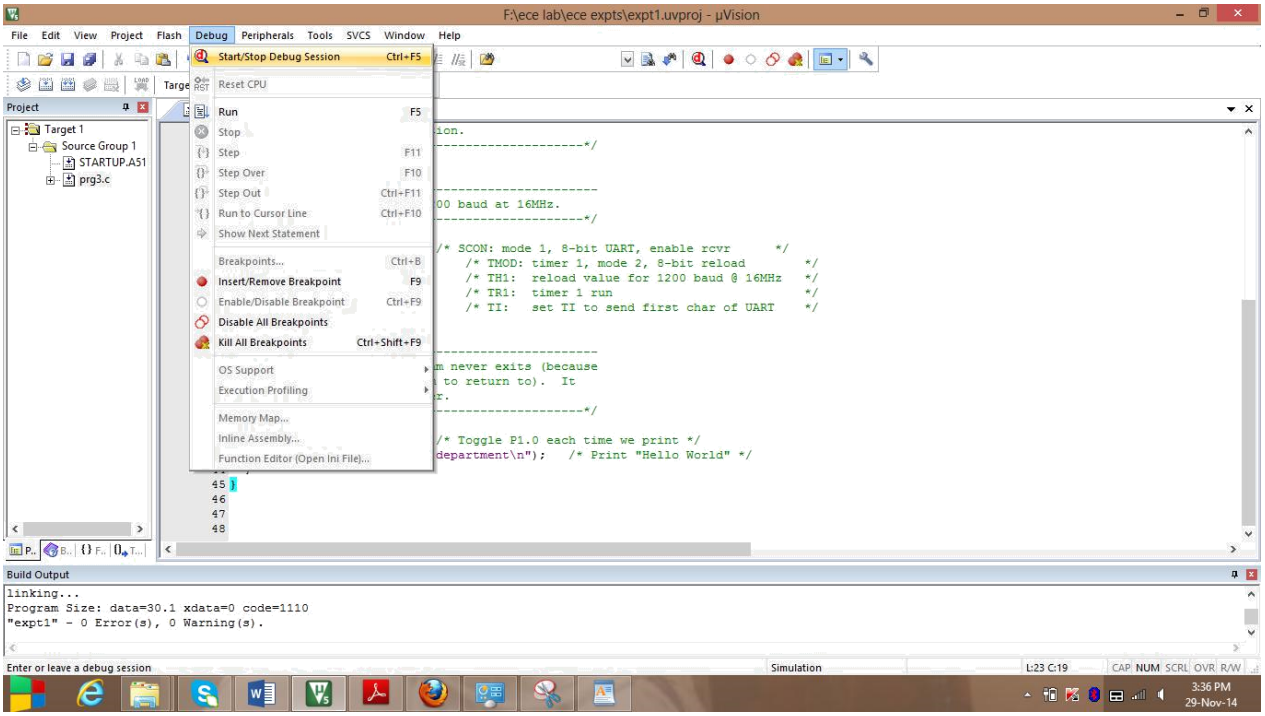
Select the program prg3



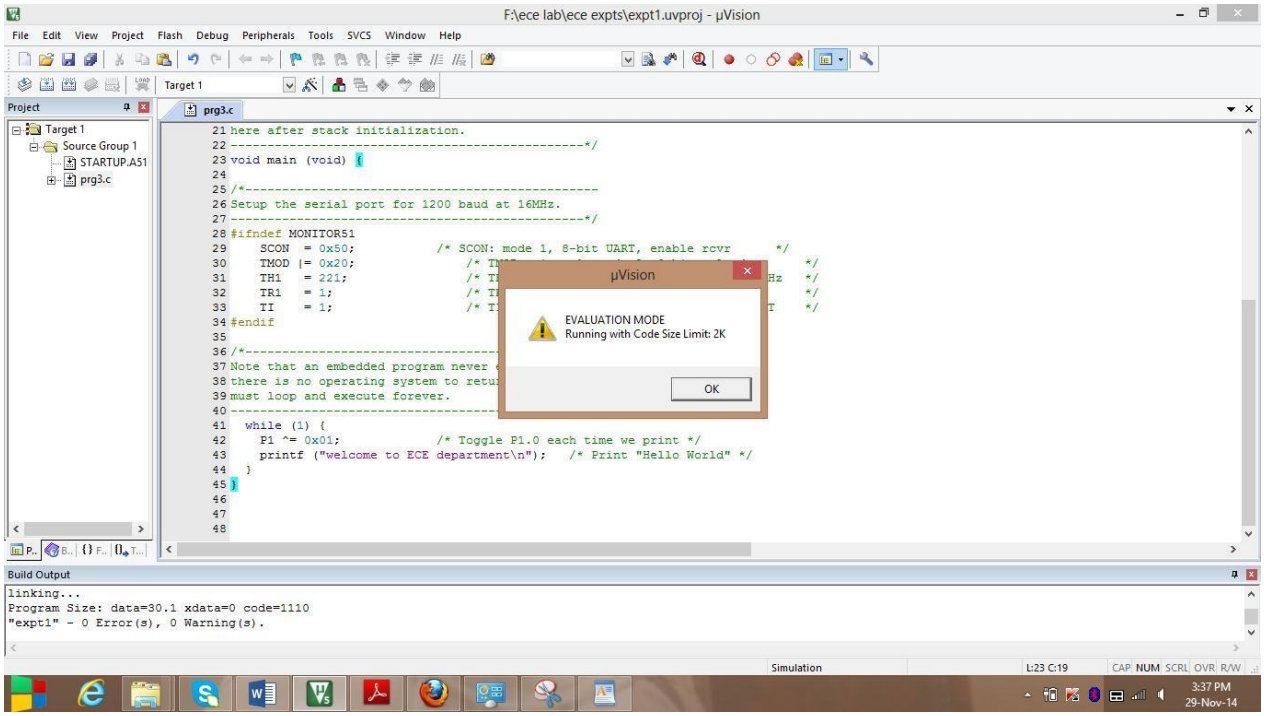
Now prg3.c is added to Source Group 1



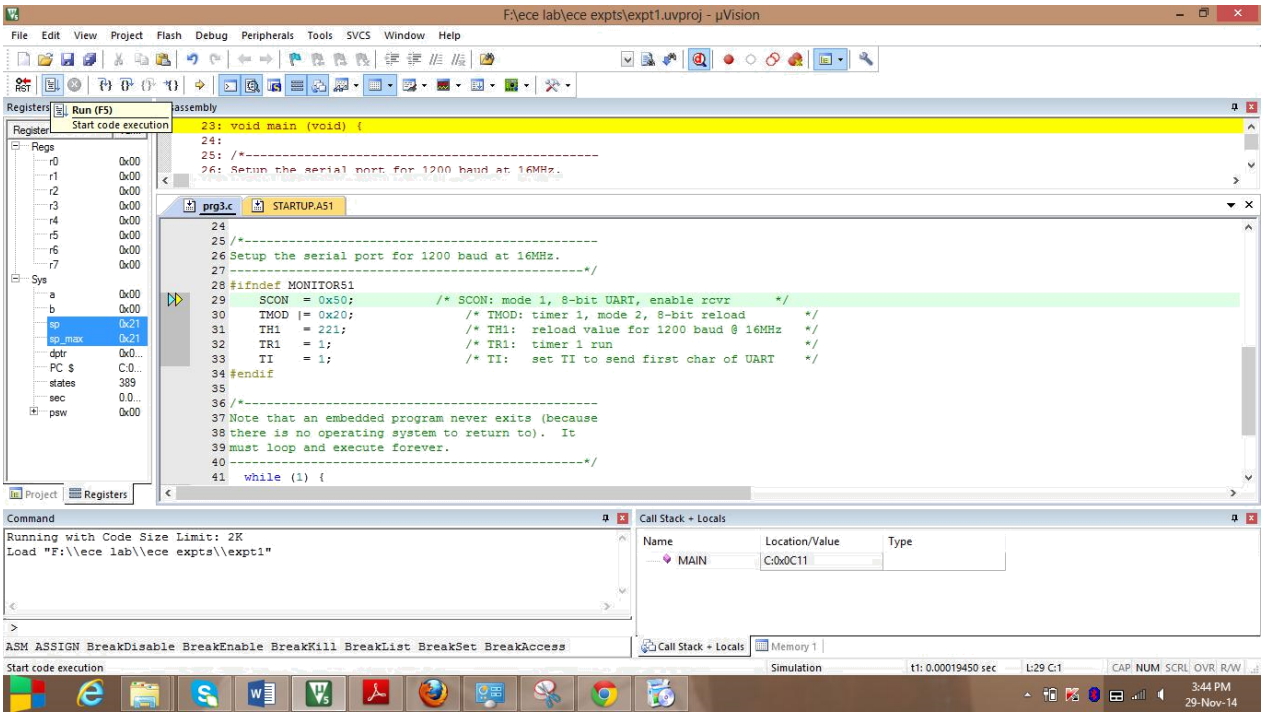
Build the target



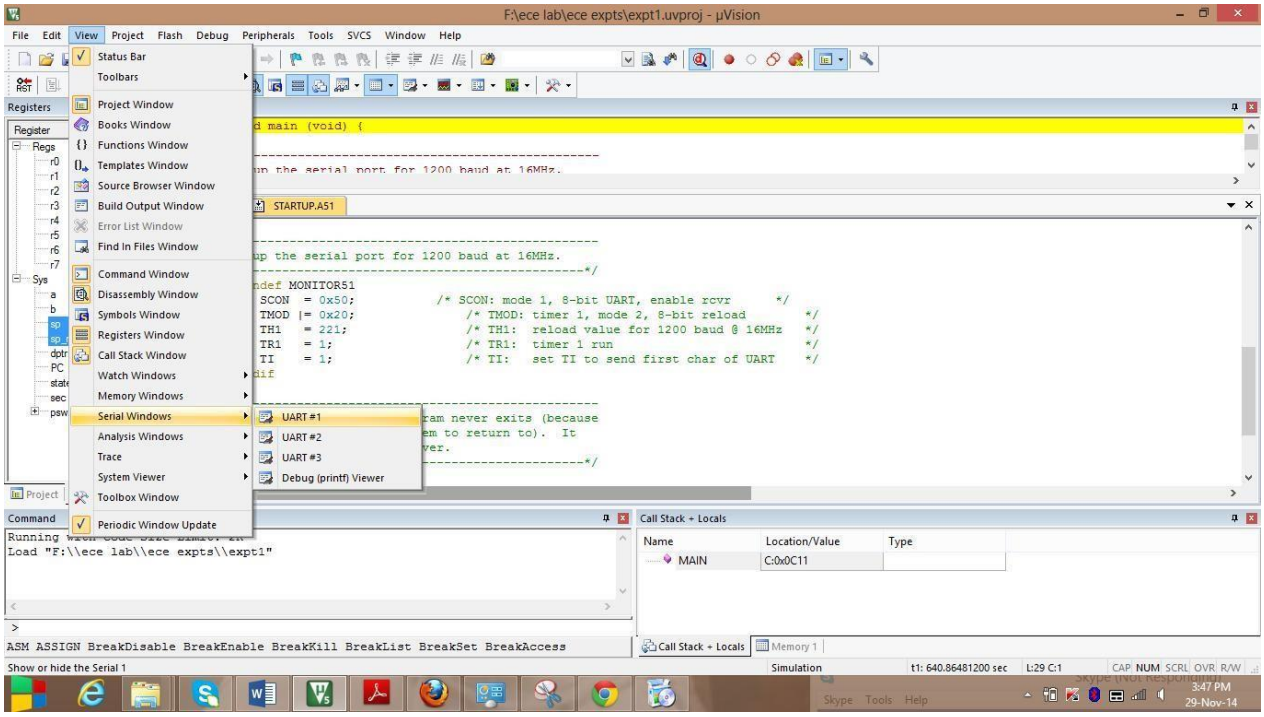
Debugging the target



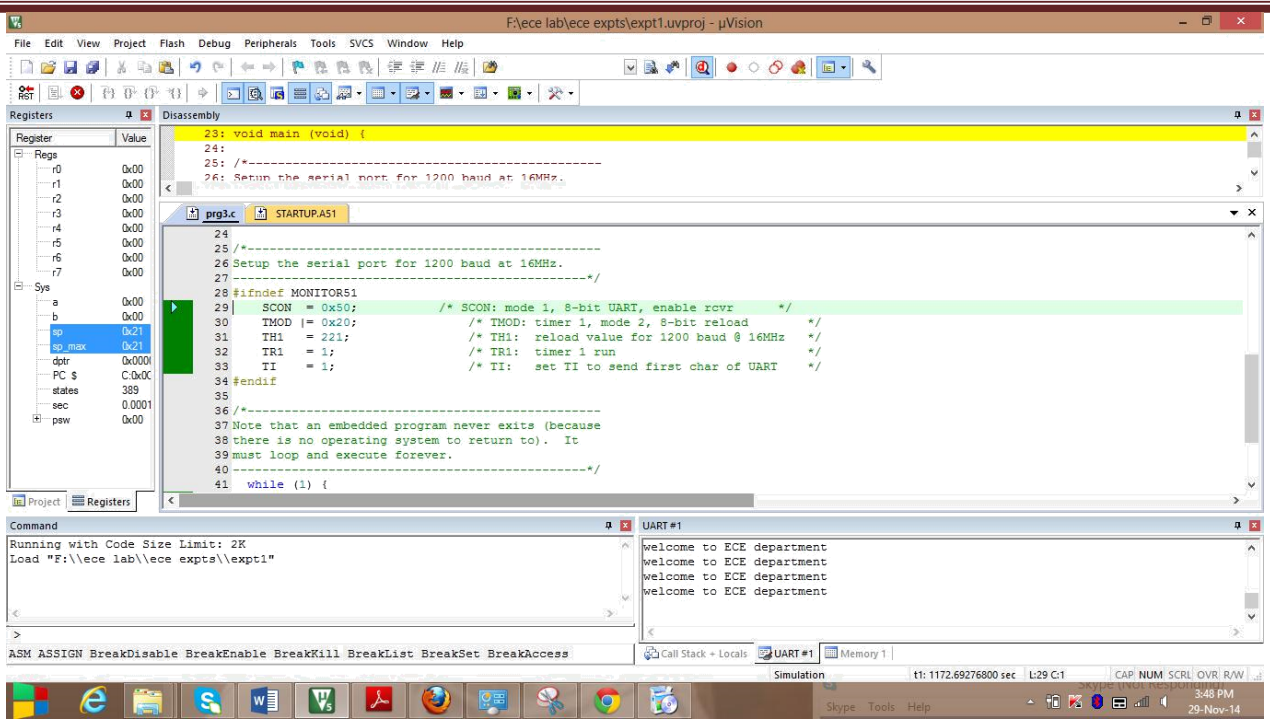
New window evaluation mode appeared. Press ok



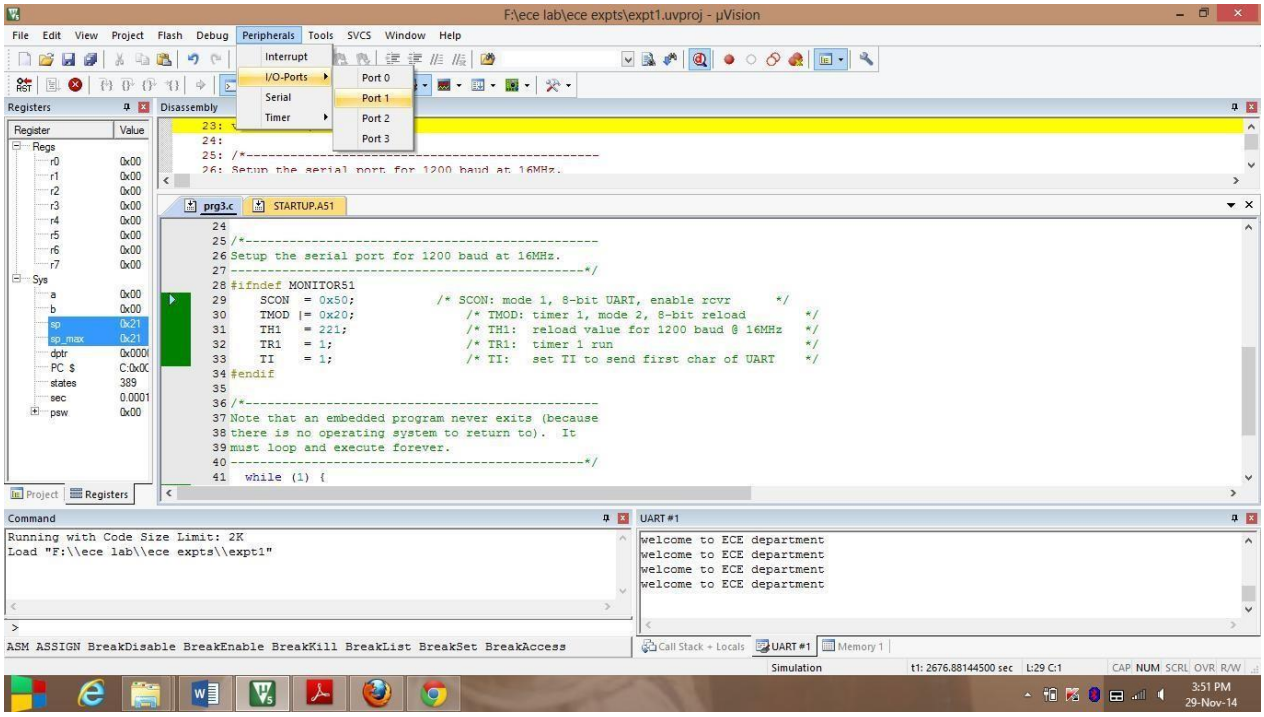
Run the program



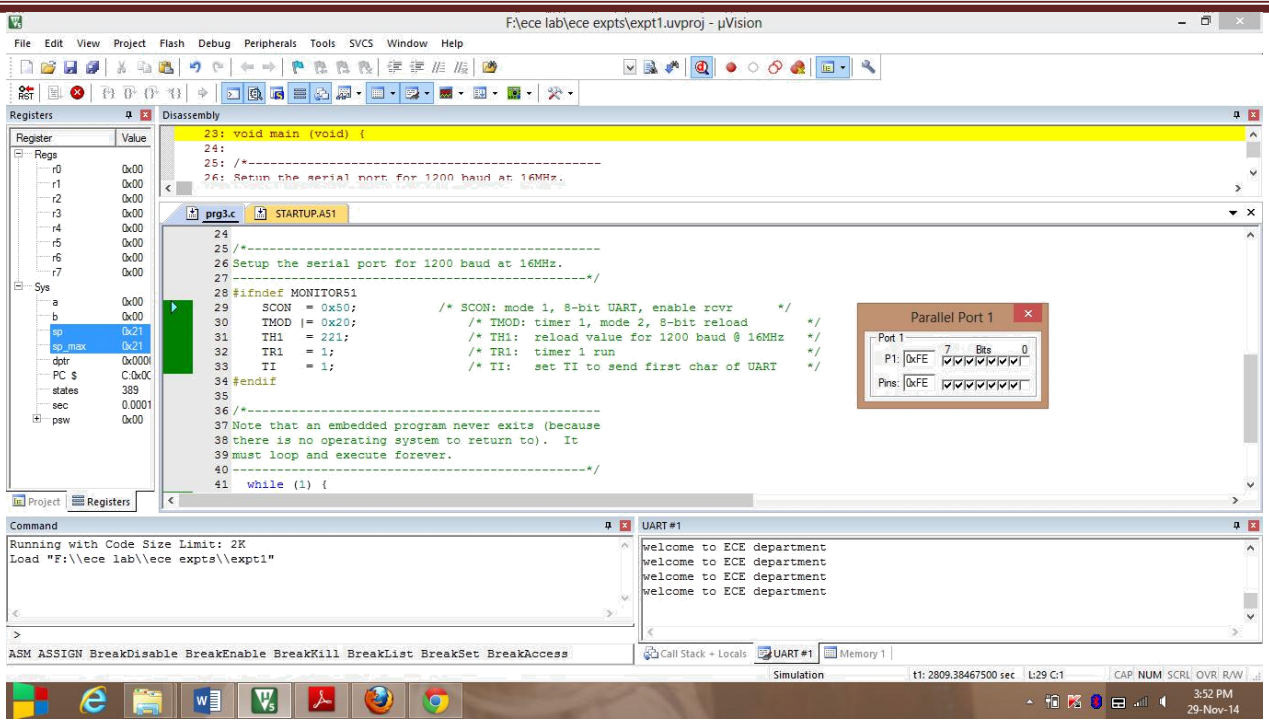
Selecting for UART#1 from serial windows



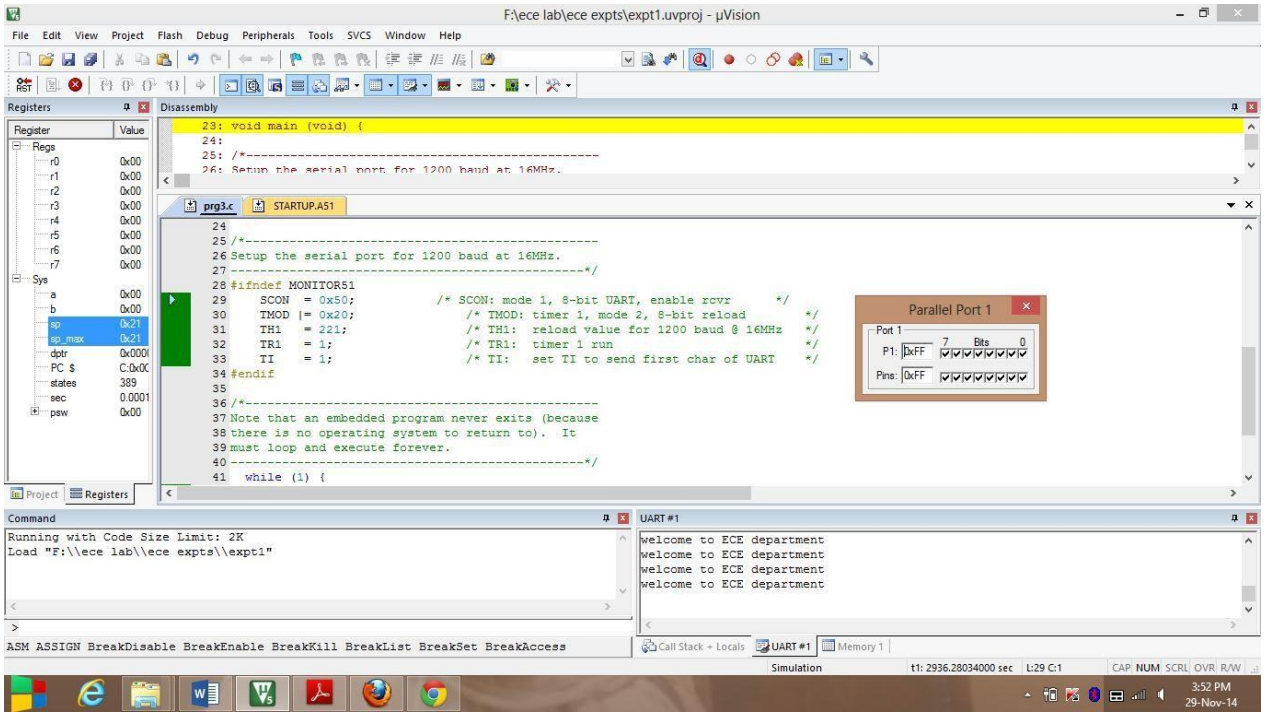
Check the output at UART#1 window



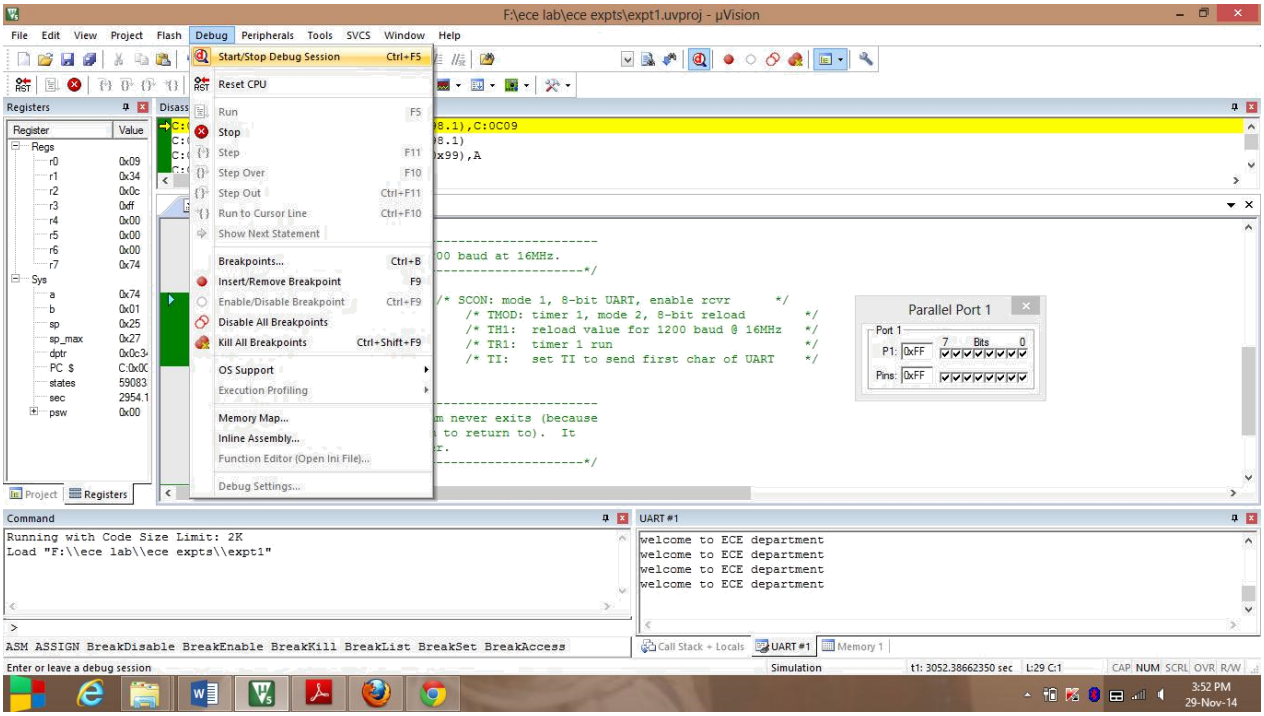
Select Port1 from i/o ports in peripherals



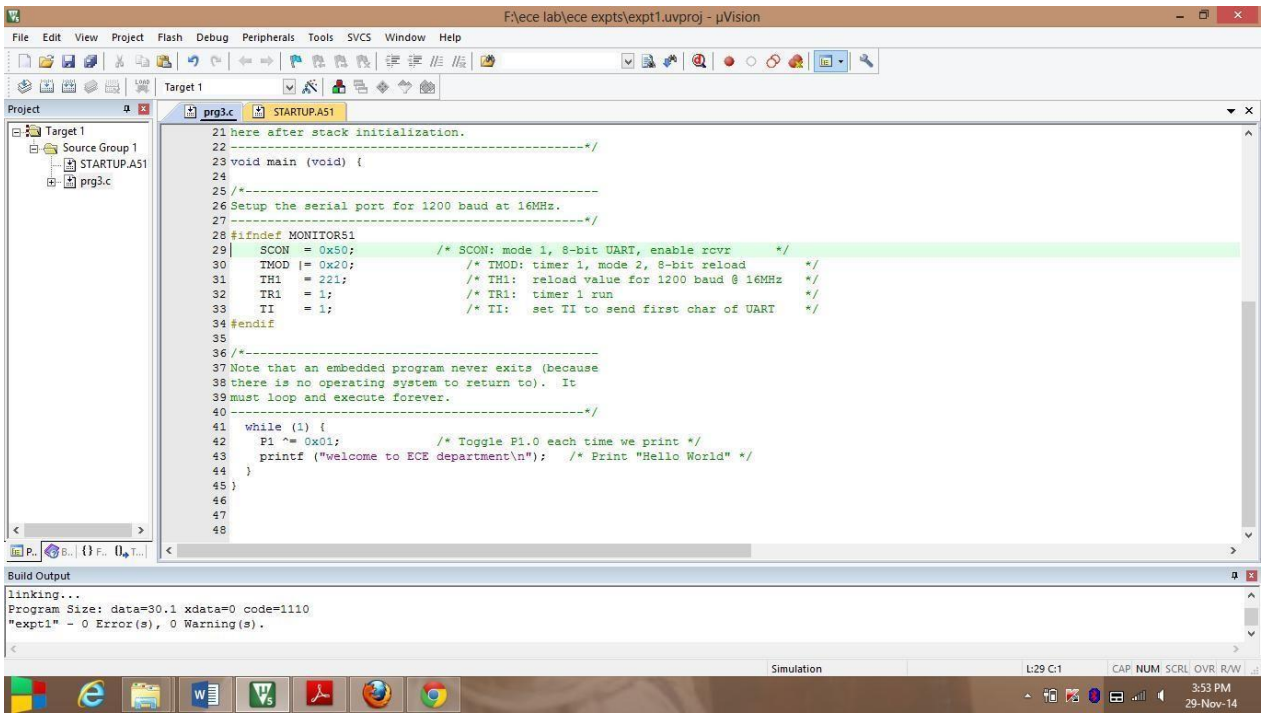
Port1 window is displayed with D0 as 0



Port1 window is displayed with D0 as 0



Stop debugging process



After debugging, window appears in this format

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

```
#include <reg51.h>
#include <stdio.h>
void main()
{
   unsigned int num1 = 0xcc; // 0xCC
   unsigned int num2 = 0xAA; // 0xAA
   unsigned long result = 0;
   unsigned char i;
SCON = 0x50;
TMOD = 0x20;
TH1 = 221;
TR1 = 1;
TI =1;
  // Multiply the two numbers
   for (i = 0; i < 16; i++) {
      if ((num2 & (1 << i)) != 0) {
         result += (num1 << i);
      }
   }
  // Print the result
   printf("Result: %lu \n", result);  // Answer was displayed in Decimal number
}
```

Result:

Input is CC x AA

Answer: Result: 8788 or 34680 in decimal the answer was displayed

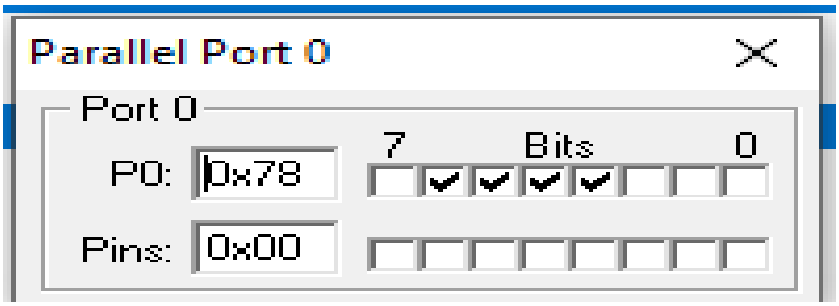
2. Write a 8051 C program to find the sum of first 10 integer numbers.

```
#include <reg51.h>
void main ( )
 {
unsigned char sum=0;
unsigned char a[]={0x1,0x2,0x3,0x4,0x5, 0x6,0x7,0x8,0x9,0xa};
 unsigned char i;
for (i=0; i<10; i++)
 {
sum+=a[i]; ACC=sum;
 }
while(1);
}
```

RESULT: Note: Check the result in D:0XE0 location

3. Write a 8051 C program to find factorial of a given number.

```
#include <reg51.h>
int main()
{
   int n, i;
   unsigned long fact = 1;
      // shows error if the user enters a negative integer
     {                                                    n = 5;   //// INPUT VALUE IS 05H
      for (i = 1; i <= n; ++i)
{
         fact *= i;
      P0 = fact;
}
   }
   return 0;
}
```



(OR)

```
#include <reg51.h>
void main ( )
 {
unsigned int a=1,i;
for(i=7;i>0; i--)
 a*=i;
while(1);
}
```

NOTE: CHECK THE RESULTS IN R6 & R7 REGISTERS *. MAXIMUM " i" VALUE CAN BE 8

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

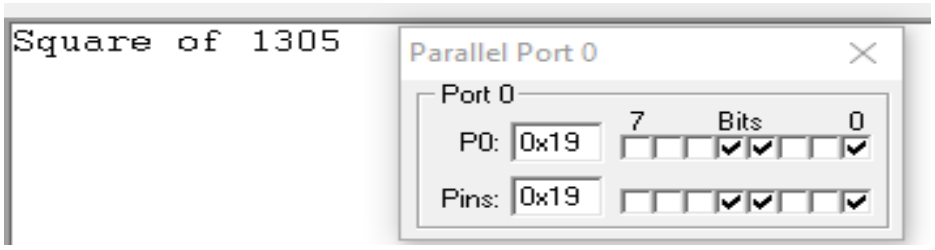
```
#include<reg51.h>
#include <stdio.h>
int main()
{  unsigned int arr[] = { 0x1234, 0x5678, 0x9ABC, 0xDEFF };
    unsigned long result = 0;
    unsigned char i;
SCON = 0x50;
TMOD = 0x20;
TH1 = 221;
TR1 = 1;
TI =1;
 // Add the numbers in the array
    for (i = 0; i < sizeof(arr) / sizeof(arr[0]); i++) {
        result += arr[i];
    }
      // Store the result in internal RAM
    *((unsigned long *)0x8000) = result;

    // Print the result
    printf("Result: %1u\n", result);
    return 0;
}
Result: 123495 // In decimal the result was displayed
                                                    (OR)
#include <reg51.h>
void main ( )
 { unsigned char a[]={0xff,0xff}, b[]={0xff,0xff};
   unsigned int d;
   unsigned char carry=0, c;
  CY=0;
  c= a[0] + b[0];
  if(CY==1) carry=1;
  else carry=0;
  CY=0;
 d= a[1] + b [1] +carry;
 while(1);
 }
```

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

```
#include <reg51.h>
#include<stdio.h>
void main()
{
   unsigned char num, square;
   unsigned char lookup_table[10] = { 1, 4, 9, 16, 25, 36, 49, 64, 81, 100 };
SCON = 0x50;
TMOD = 0x20;
TH1 = 221;
TR1 = 1;
TI =1;
   // Input the number to find the square of
   num = 5;
   // Find the square of the number using the look-up table
   square = lookup_table[num - 1];
     P0 = square;
   // Output the result
   printf("Square of %u:", num, P0);
}
```

Results: Here output was displayed in PORT 0 and also in serial window



(OR)

```
#include <reg51.h>
void main ( )
{
 unsigned char tab[]={1,4,9,16,25,36,49, 64,81,100};
 unsigned char num=4,result; if(num<11) result =tab[num-1];
 else result=0;
 while(1);
}
```

Note: Result will be in I:0x12

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

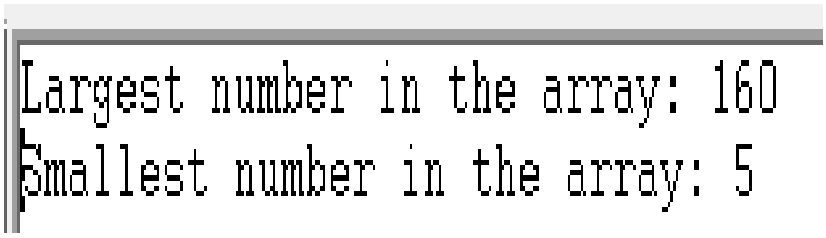
```
#include <reg51.h>
#include<stdio.h>
void main()
{
   unsigned int arr[32] = { 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95,
100, 105, 110, 115, 120, 125, 130, 135, 140, 145, 150, 155, 160 };
   unsigned int largest = arr[0], smallest = arr[0];
   unsigned char i;

SCON = 0x50;
TMOD = 0x20;
TH1 = 221;
TR1 = 1;
TI =1;

   // Find the largest and smallest numbers in the array
   for (i = 1; i < 32; i++) {
      if (arr[i] > largest) {
         largest = arr[i];
      }
      if (arr[i] < smallest) {
         smallest = arr[i];
      }
   }

   // Output the results
   printf("Largest number in the array: %u\n", largest);
   printf("Smallest number in the array: %u\n", smallest);
}
```

Results:



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

```
 #include <reg51.h>
#include <stdio.h>
void bubble_sort(int arr[], int n, int order) {
    int i, j, temp;
    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - i - 1; j++) {
            if (order == 1) { // Ascending order
                if (arr[j] > arr[j + 1]) {
                    temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            } else { // Descending order
                if (arr[j] < arr[j + 1]) {
                    temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }
    }
}

int main() {
    int arr[] = { 10, 5, 20, 15, 30, 25 };
    int n = sizeof(arr) / sizeof(arr[0]);
    int i;

    // Print the original array
    printf("Original array: ");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    // Sort the array in ascending order
    bubble_sort(arr, n, 1);
```

```
    printf("Array in ascending order: ");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    // Sort the array in descending order
    bubble_sort(arr, n, 0);
    printf("Array in descending order: ");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

Result:

Array in descending order: 30 25 20 15 10 5

Array in ascending order: 5 10 15 20 25 30

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

```
#include <reg51.h>
#include<stdio.h>
void main()
{
unsigned char *ptr = (unsigned char *)0x8000;
unsigned char value = *ptr = 0x14;
unsigned char byte1 = *ptr;
unsigned char byte2 = *(ptr + 1);
int ones = 0, zeros = 0;
int i;
SCON = 0x50;
TMOD = 0x20;
TH1 = 221;
TR1 = 1;
TI =1;
   // Count the number of ones and zeros in byte1
   for (i = 0; i < 8; i++) {
      if (byte1 & (1 << i)) {
         ones++;
      } else {
         zeros++;
      }
   }

   // Count the number of ones and zeros in byte2
   for (i = 0; i < 8; i++) {
      if (byte2 & (1 << i)) {
         ones++;
      } else {
         zeros++;
      }
   }

   // Print the results
   printf("Number of ones: %d\n", ones);
   printf("Number of zeros: %d\n", zeros);
}
```

Result:

Input given is 12

Output: Number of ones :2

Number of zeros:14

(OR)

```
#include <reg51.h>
 void main ( )
 {
 unsigned char a[]={0xfa,0xfa},i;
 unsigned char ones, zeros;
CY=0;
for(i=0;i<8;i++)
{
a[0]>>=1;
if(CY==1) ones++; else zeros++;
 }
for(i=0;i<8;i++)
 {
a[1]>>=1;
if(CY==1) ones++;
else zeros++;
}
while(1);
}
```

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

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

void main()
{
   unsigned char i, count = 0;
     unsigned long arr[10] = {0xF5, 0x10, 0x15, 0x20, 0x25, 0xF0, 0xF5, 0xF0, 0xF5, 0x50};

SCON = 0x50;
TMOD = 0x20;
TH1 = 221;
TR1 = 1;
TI =1;

    // Count the number of negative numbers
   for(i = 0; i < 10; i++)
   {
      if(arr[i] & 0x80)
      {
         count++;
      }
   }

   // Output the result
   printf("There are %d negative numbers in the series.\n", count);
}
```

Results

There are 1280 negative numbers in the series

Here 1280 = 5 number

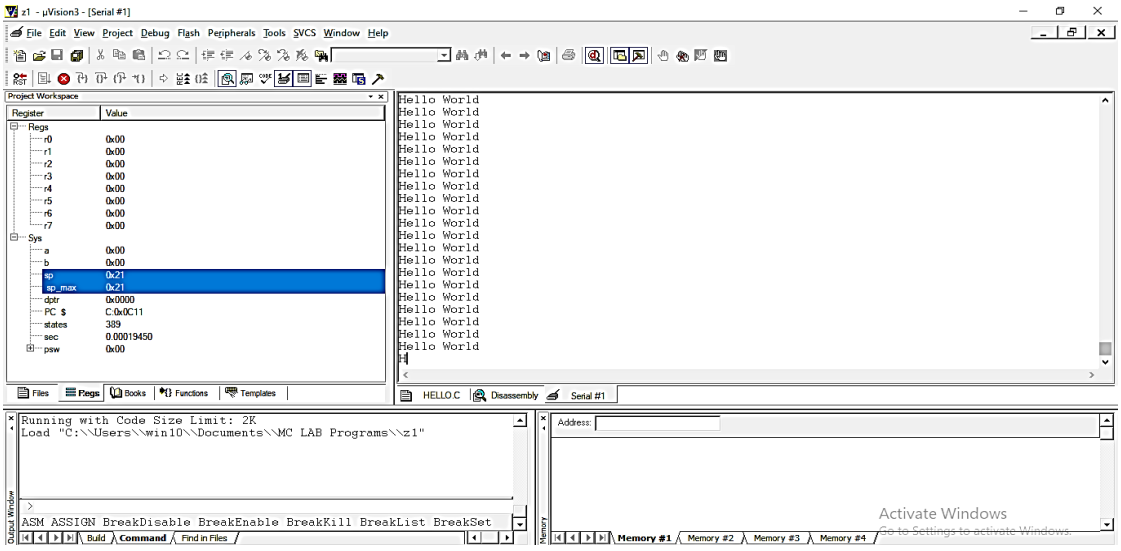
Noting that 256 = 1 number, 512 = 2 numbers like

(OR)

```
#include <reg51.h>
void main ( )
{
 unsigned char a[]={0xfa,0xfa,0x12,0x34,0xaa},i;
unsigned char pos, neg;
CY=0;
for(i=0;i<5;i++)
{
a[i]<<=1;
if(CY==1) neg++;
else pos++;
CY=0;
}
while(1);
}
```

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

```
#include<reg51.h>
#include<stdio.h>
#ifdef MONITOR51
char code reserve [3]_at_0x23;
#endif
Void main (void)
{
#ifndef MONITOR51
SCON = 0x50;
TMOD = 0x20;
TH1 = 221;
TR1 = 1;
TI =1;
#endif
While(1)
{
P1^=0x01;
Printf("Hello Word \n ");
}
```



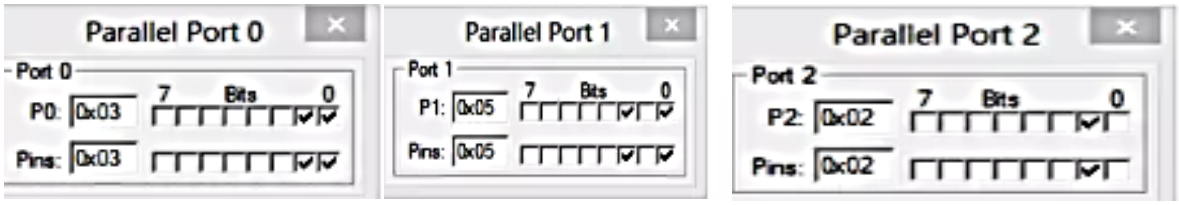
(OR)

```
# include <reg51.h>
void sertx (unsigned char);
void main (void)
{
unsigned char array[100]={ "SMVITM"};
unsigned char i, key;
TMOD =0x20; // use Timer 1, 8 bit auto reload TH1= 0xFD; // 9600 baud rate SCON= 0x50; TR1
=1; // Start Timer for(i=0; i<6; i++)
{
key=array[i];
sertx(key);
 }
while(1);
 }
void sertx(unsigned char x)
{
SBUF=x; // place value in buffer while (TI==0); // wait until transmitted TI=0;
}
```

Note: After debugging go to View ☐Serial window ☐UART#0

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).

```
#include <reg51.h>
void main()
{
unsigned char x, binbyte, d1, d2, d3;
binbyte = 0xfd;
x = binbyte/10;
d1 = binbyte%10;
d2 = x%1;
d3 = x/10;
P0 = d1;
p1 = d2;
p2 = d3;
}
```

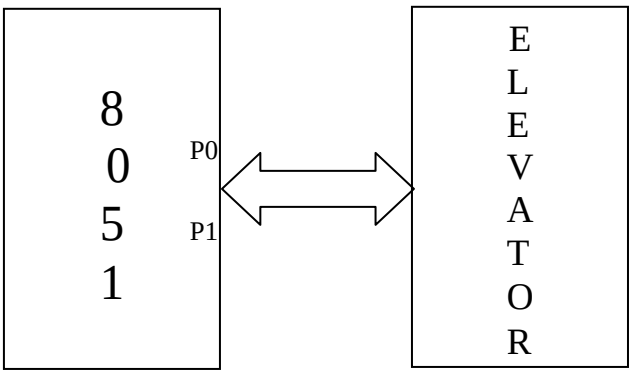


ADDITIONAL PROGRAMMES FOR PRACTICE

Extra Program:

Write a C program to interface Elevator.

Block Diagram:



Theory:

The operation of the elevator is as follows:

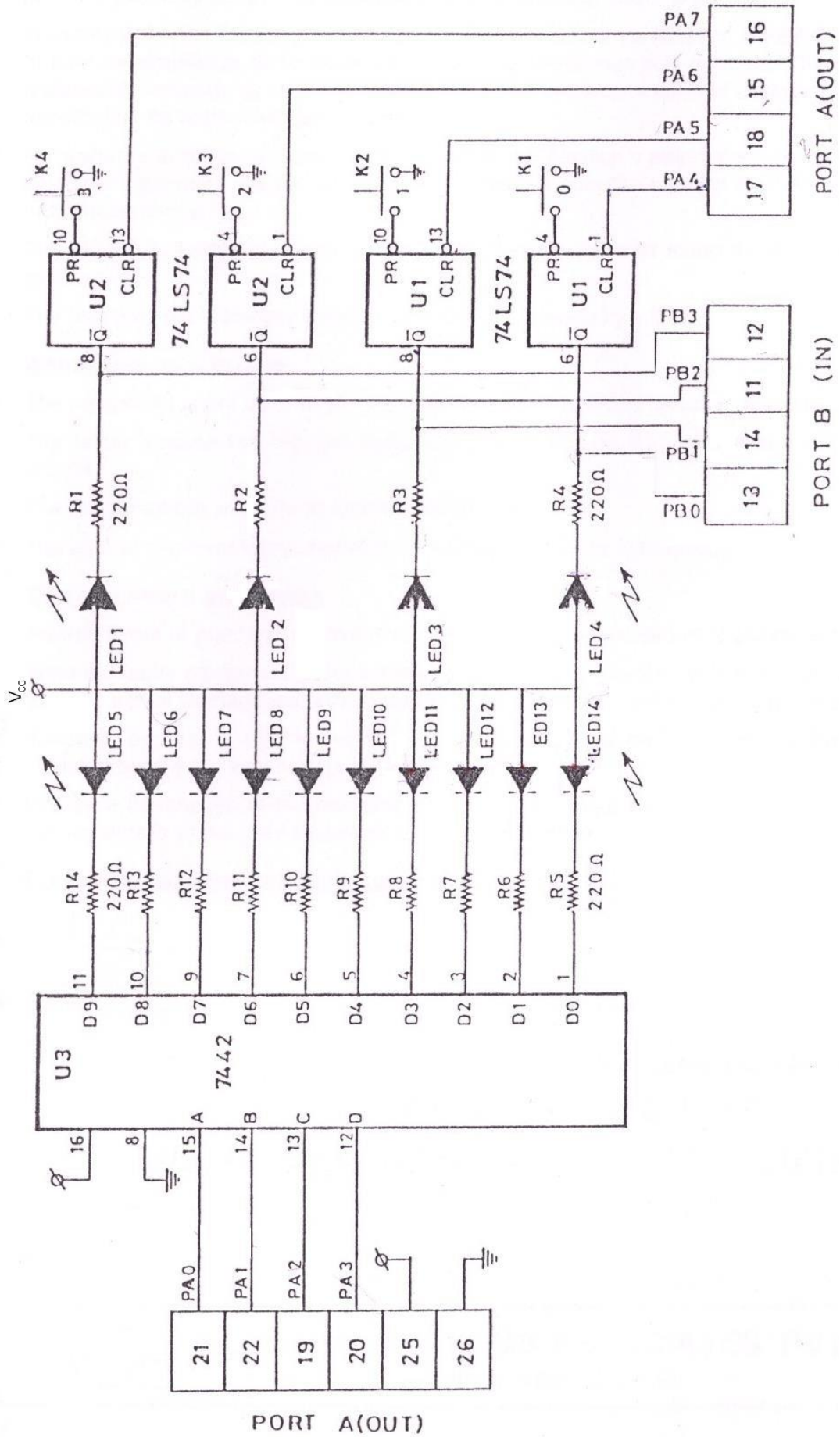
- Initially, the elevator is at ground floor.
- When the elevator reaches any floor, it stays at that floor until a request from another floor is made. When such a request is detected, it moves to that floor.
- The floor request are scanned in fixed order i.e., floors 0, 1, 2 and 3.

This interface simulates the control and operation of an elevator. Four floors assumed and for each floor a key and corresponding LED indicator are provided to serve as request buttons and request status indicator. The elevator itself is represented by a column of ten LEDs. The motion of elevator can be simulated by turning on successive LEDs one at a time. Te delay between turning off one LED and turning on the next LED can simulate the "speed" of the elevator. User can read the request status information through one port, reset the request indicators through another port and control the elevator (LED column) through another port.

Description of the Circuit:

This interface has four keys, marked 0, 1, 2, and 3 representing the request buttons at the four floors. Pressing of key causes a corresponding Flip-Flop to be set. The outputs of the four Flip-flops can be read through port B (PBO, PBI, PB2 and PB3). Also, the status of these signals is reflected by a setoff 4 LEDs. The Flip-Flop can be rest (LEDs are cleared) through port A (PA54, PA5, PA6, and PA7). A column of 10 LEDs, representing the elevator can be controlled through Port A (PA0, PA1, PA2 and PA3). These port lines are fed to the inputs of the decoder 7442 whose outputs are used to

control the on/off states of the LEDs which simulate the motion of the elevator



Program to interface Elevator:

```
#include <REG51xD2.H> void delay(unsigned
int);

main()
{
  unsigned char Flr[9]={0xff,0x00,0x03,0xff,0x06,0xff,0xff,0xff,0x09}; unsigned char
  FClr[9]={0xff,0x0E0,0x0D3,0xff,0x0B6,0xff,0xff,0xff,0x79};unsigned char ReqFlr,CurFlr =
  0x01,i,j;

   P0 = 0x00; P0 = 0x0f0;

   while(1)
   {
     P1 = 0x0f;
     ReqFlr = P1 | 0x0f0; while(ReqFlr == 0x0ff)
     ReqFlr = P1 | 0x0f0; //Read Request Floor from P1ReqFlr = ~ReqFlr;

     if(CurFlr == ReqFlr) //If Request floor is equal to Current Floor
     {
       P0 = FClr[CurFlr]; //Clear Floor Indicator continue;     //Go up
       to read again
     }

     else if(CurFlr > ReqFlr) //If Current floor is > request floor
     {
          i = Flr[CurFlr] - Flr[ReqFlr]; //Get the no of floors to travel
          j = Flr[CurFlr];
          for(;i>0;i--)               // Move the indicator down
          {
            P0 = 0x0f0|j;j--;
            delay(50000);
          }
     }

     else  // If Current floor is < request floor
     {
           i = Flr[ReqFlr] - Flr[CurFlr]; //Get the no of floors to travel
           j = Flr[CurFlr];
           for(;i>0;i--)            // Move the indicator Up
           {
             P0 = 0x0f0 | j;j++;
             delay(50000);
           }
     }
       CurFlr = ReqFlr;             // Update Current floorP0 =
       FClr[CurFlr];                // Clear the indicator
     }
  }

void delay(unsigned int x)
{
      for(;x>0;x--);
}
```

VIVA QUESTIONS

8051 MICROCONTROLLER

1. What do you mean by Embedded System? Give examples?
2. Why are embedded Systems useful?
3. What are the segments of Embedded System?
4. What is Embedded Controller?
5. What is Microcontroller?
6. List out the differences between Microcontroller and Microprocessor.
7. How are Microcontrollers more suitable than Microprocessor for Real Time Applications?
8. What are the General Features of Microcontroller?
9. Explain briefly the classification of Microcontroller.
10. Explain briefly the Embedded Tools.
11. Explain the general features of 8051 Microcontroller.
12. How many pin the 8051 has?
13. Differentiate between Program Memory and Data Memory.
14. What is the size of the Program and Data memory?
15. Write a note on internal RAM. What is the necessity of register banks? Explain.
16. How many address lines are required to address 4K of memory? Show the necessary calculations.
17. What is the function of accumulator?
18. What are SFR's? Explain briefly.
19. What is the program counter? What is its use?
20. What is the size of the PC?
21. What is a stack pointer (SP)?
22. What is the size of SP?
23. What is the PSW? And briefly describe the function of its fields.
24. What is the difference between PC and DPTR?
25. What is the difference between PC and SP?
26. What is ALE? Explain the functions of the ALE in 8051.
27. Describe the 8051 oscillator and clock.
28. What are the disadvantages of the ceramic resonator?
29. What is the function of the capacitors in the oscillator circuit?
30. Show with an example, how the time taken to execute an instruction can be calculated.
31. What is the Data Pointer register? What is its use in the 8051?
32. Explain how the 8051 implement the Harvard Architecture?
33. Explain briefly the difference between the Von Neumann and the Harvard Architecture.
34. Describe in detail how the register banks are organized.

35. What are the bit addressable registers and what is the need?
36. What is the need for the general purpose RAM area?
37. Write a note on the Stack and the Stack Pointer.
38. Why should the stack be placed high in internal RAM?
39. Explain briefly how internal and external ROM gets accessed.
40. What are the different addressing modes supported by 8051 Microcontroller ?
41. Explain the Immediate Addressing Mode.
42. Explain the Register Addressing Mode.
43. Explain the Direct Addressing Mode.
44. Explain the Code Addressing Mode.
45. Explain in detail the Functional Classification of 8051 Instruction set
46. What are the instructions used to operate stack?
47. What are Accumulator specific transfer instructions?
48. What is the difference between INC and ADD instructions?
49. What is the difference between DEC and SUBB instructions?
50. What is the use of OV flag in MUL and DIV instructions?
51. What are single and two operand instructions?
52. Explain Unconditional and Conditional JMP and CALL instructions.
53. Explain the different types of RETURN instructions.
54. What is a software delay?
55. What are the factors to be considered while deciding a software delay?
56. What is a Machine cycle?
57. What is a State?
58. Explain the need for Hardware Timers and Counters?
59. Give a brief introduction on Timers/Counter.
60. What is the difference between Timer and Counter operation?
61. How many Timers are there in 8051?
62. What are the three functions of Timers?
63. What are the different modes of operation of timer/counter?
64. Give a brief introduction on the various Modes.
65. What is the count rate of timer operation?
66. What is the difference between mode 0 and mode 1?
67. What is the difference Modes 0,1,2 and 3?
68. How do you differentiate between Timers and Counters?
69. Explain the function of the TMOD register and its various fields?
70. How do you control the timer/counter operation?
71. What is the function of TF0/TF1 bit
72. Explain the function of the TCON register and its various fields?
73. Explain how the Timer/Counter Interrupts work.

74. Explain how the 8051 counts using Timers and Counters.
75. Explain Counting operation in detail in the 8051.
76. Explain why there is limit to the maximum external frequency that can be counted.
77. What's the benefit of the auto-reload mode?
78. Write a short note on Serial and Parallel communication and highlight their advantagesand disadvantages.
79. Explain Synchronous Serial Data Communication.
80. Explain Asynchronous Serial Data Communication.
81. Explain Simplex data transmission with examples.
82. Explain Half Duplex data transmission with examples.
83. Explain Full Duplex data transmission with examples.
84. What is Baud rate?
85. What is a Modem?
86. What are the various registers and pins in the 8051 required for Serial communication?Explain briefly.
87. Explain SCON register and the various fields.
88. Explain serial communication in general (synchronous and asynchronous). Also explainthe use of the parity bit.
89. Explain the function of the PCON register during serial data communication.
90. How is data transmitted serially in the 8051? Explain briefly.
91. How is data received serially in the 8051? Explain briefly.
92. What are the various modes of Serial Data Transmission? Explain each mode briefly.
93. Explain with a timing diagram the shift register mode in the 8051.
94. What is the use of the serial communication mode 0 in the 8051?
95. Explain in detail the Serial Data Mode 1 in the 8051.
96. Explain how the Baud rate is calculated for the Serial Data Mode 1.
97. How is the Baud rate for the Multiprocessor communication Mode calculated?
98. Explain in detail the Multiprocessor communication Mode in the 8051.
99. Explain the significance of the 9th bit in the Multiprocessor communicationmode.
100. Explain the Serial data mode 3 in the 8051.
101. What are interrupts and how are they useful in Real Time Programming?
102. Briefly describe the Interrupt structure in the 8051.
103. Explain about vectored and non-vectored interrupts in general.
104. What are the five interrupts provided in the 8051?
105. What are the three registers that control and operate the interrupts in 8051?
106. Describe the Interrupt Enable (IE) special function register and its variousbits.
107. Describe the Interrupt Priority (IP) special function register and its need.
108. Explain in detail how the Timer Flag interrupts are generated.
109. Explain in detail how the Serial Flag interrupt is generated.

110. Explain in detail how the External Flag interrupts are generated.
111. What happens when a high logic is applied on the Reset pin?
112. Why the Reset interrupt is called a non-maskable interrupt?
113. Why do we require a reset pin?
114. How can you enable/disable some or all the interrupts?
115. Explain how interrupt priorities are set? And how interrupts that occur simultaneously are handled.
116. What Events can trigger interrupts, and where do they go after getting triggered?
117. What are the actions taken when an Interrupt Occurs?
110. What are Software generated interrupts and how are they generated?
111. What is RS232 and MAX232?
112. What is the function of RS and E pins in an LCD?
113. What is the use of R/W pin in an LCD?
114. What is the significance of DA instruction?
115. What is packed and unpacked BCD?
116. What is the difference between CY and OV flag?
117. When will the OV flag be set?
118. What is an ASCII code?

'C' PROGRAMMING

1. Which type of language is C?
2. What is a compiler?
3. What is an algorithm?
4. What is a c token and types of c tokens?
5. How many Keywords (reserve word)are in C?
6. What is an identifier?
7. What are the Back Slash character constants or Escape sequence characters available in C?
8. What is a variable?
9. Why is C called a mid-level programming language?
10. Who developed C language?
11. Which type of language is C?
12. What is a compiler?
13. What is IDE?
14. What is a program?
15. What is an algorithm?
16. What is structure of C program?
17. What is a C token and types of C tokens?

QUESTION BANK

Sl.No	Experiments
Conduct the following experiments by writing C Program using Keil microvision simulator (any 8051 microcontroller can be chosen as the target).	
1	Write & execute 8051 C program to multiply two 16 bit binary numbers using Keil microvision simulator
2	Write & execute 8051 C program to find the sum of first 10 integer numbers using Keil microvision simulator.
3	Write & execute 8051 C program to find factorial of a given number using Keil microvision simulator.
4	Write & execute 8051 C program to add an array of 16 bit numbers and store the 32 bit result in internal RAM using Keil microvision simulator.
5	Write & execute 8051 C program to find the square of a number (1 to 10) using look-up table using Keil microvision simulator.
6	Write & execute 8051 C program to find the largest/smallest number in an array of 32 numbers using Keil microvision simulator.
7	Write & execute 8051 C program to arrange a series of 32 bit numbers in ascending/descending order using Keil microvision simulator.
8	Write & execute 8051 C program to count the number of ones and zeros in two consecutive memory locations using Keil microvision simulator.
9	Write & execute 8051 C program to scan a series of 32 bit numbers to find how many are negative using Keil microvision simulator.
10	Write & execute 8051 C program to display "Hello World" message (either in simulation mode or interface an LCD display) using Keil microvision simulator.
11	Write & execute 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).

Conduct of Practical Examination:

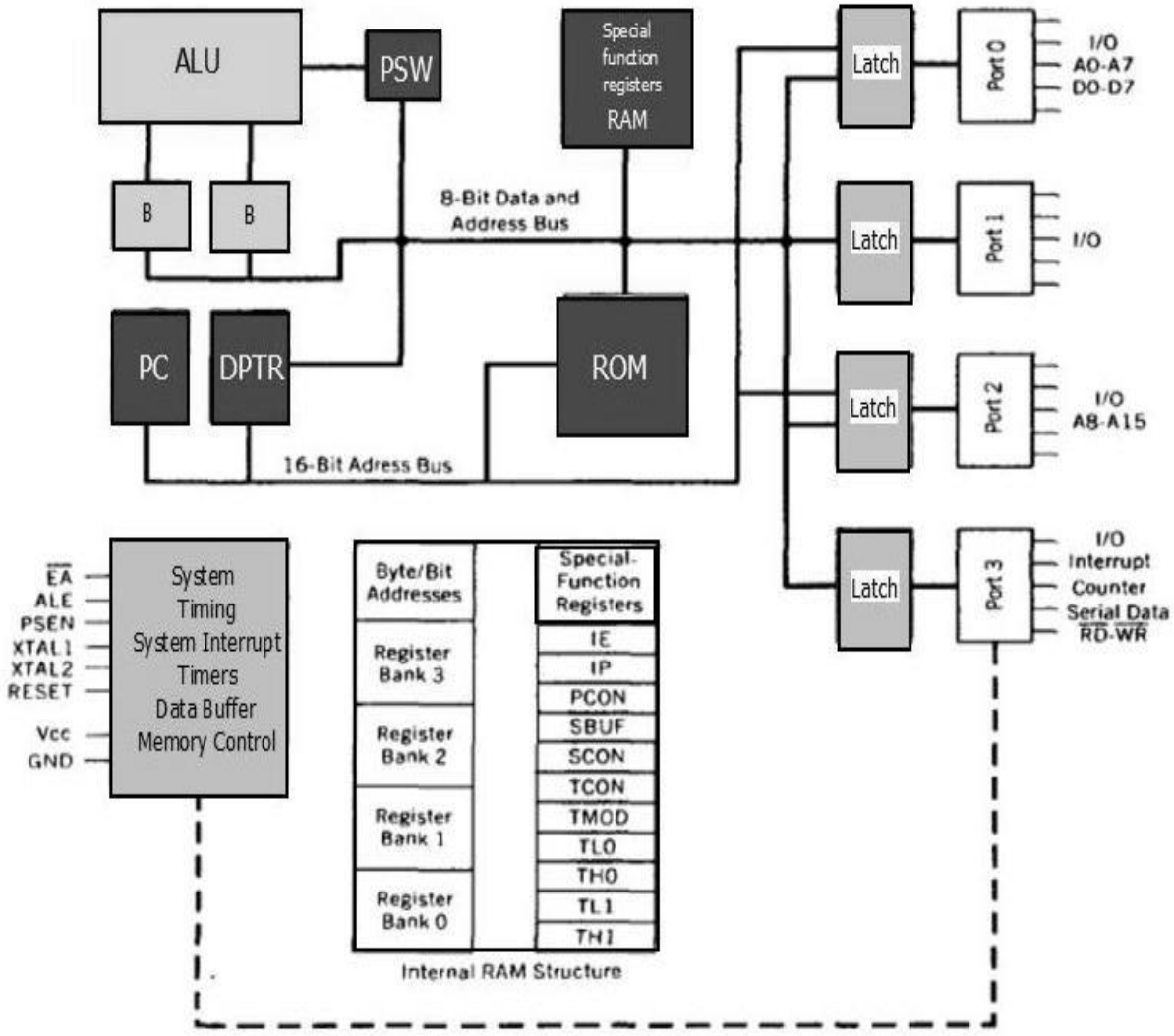
1. All laboratory experiments are to be included for practical examination.
2. Students are allowed to pick one experiment from the lot.
3. Strictly follow the instructions as printed on the cover page of answer script for breakup of marks.
4. Change of experiment is allowed only once and Marks allotted to the Write_up part to be made zero.

REFERENCES

1. Muhammad Ali Mazidi and Janice Gillespie Mazidi and Rollin D. McKinlay, "The 8051 Microcontroller and Embedded Systems – using assembly and C ", PHI, 2006 / Pearson, 2006.
2. Kenneth J. Ayala, "The 8051 Microcontroller Architecture, Programming & Applications", Second edition; Penram International, 1996 / Thomson Learning 2005.
3. V.Udayashankar and MalikarjunaSwamy , "The 8051 Microcontroller", TMH, 2009.
4. Raj Kamal , "Microcontrollers: Architecture, Programming, Interfacing and System Design", "Pearson Education, 2005.

Appendix

8051 - ARCHITECTURE



Features of 8051 Microcontroller

An 8051 microcontroller comes bundled with the following features –

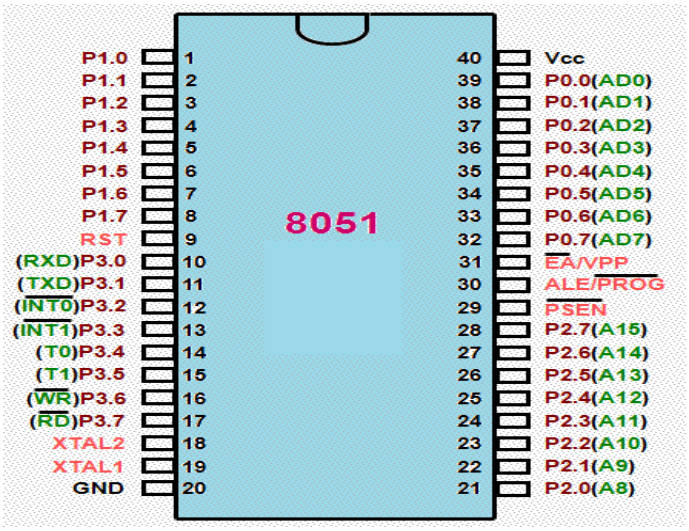
- 4KB bytes on-chip program memory (ROM)
- 128 bytes on-chip data memory (RAM)
- Four register banks
- 128 user defined software flags
- 8-bit bidirectional data bus
- 16-bit unidirectional address bus
- 32 general purpose registers each of 8-bit

- 16 bit Timers (usually 2, but may have more or less)
- Three internal and two external Interrupts
- Four 8-bit ports,(short model have two 8-bit ports)
- 16-bit program counter and data pointer
- 8051 may also have a number of special features such as UARTs, ADC, Op-amp, etc.

The 8051 microcontroller is a popular 8-bit microcontroller widely used in embedded systems. It is a single-chip microcontroller with a Harvard architecture that includes a CPU, RAM, ROM, and several peripherals. The 8051 microcontroller has a 40-pin dual in-line package (DIP) that provides various inputs and outputs for communication with external devices.

8051 microcontroller is a 40 pin Dual Inline Package (DIP). These 40 pins serve different functions like read, write, I/O operations, interrupts etc. 8051 has four I/O ports wherein each port has 8 pins which can be configured as input or output depending upon the logic state of the pins. Therefore, 32 out of these 40 pins are dedicated to I/O ports. The rest of the pins are dedicated to VCC, GND, XTAL1, XTAL2, RST, ALE, EA' and PSEN'. Pin diagram of 8051 microprocessor is as given below:

8051 Pin Diagram and Special Function Registers:



Pin diagram of 8051

Description of the Pins :

- **Pin 1 to Pin 8 (Port 1) –** Pin 1 to Pin 8 are assigned to Port 1 for simple I/O operations. They can be configured as input or output pins depending on the logic control i.e. if logic zero (0) is applied to the I/O port it will act as an output pin and if logic one (1) is applied the pin will act as an input pin. These pins are also referred to as P1.0 to P1.7 (where P1 indicates that it is a pin in port 1 and the number after '.' tells the pin number i.e. 0 indicates first pin of the port. So, P1.0 means first pin of port 1, P1.1 means second pin of the port 1 and so on). These pins are bidirectional pins.
- **Pin 9 (RST) –** Reset pin. It is an active-high, input pin. Therefore if the RST pin is high for a minimum of 2 machine cycles, the microcontroller will reset i.e. it will close and terminate

all activities. It is often referred as "power-on-reset" pin because it is used to reset the microcontroller to it's initial values when power is on (high).

- **Pin 10 to Pin 17 (Port 3) –** Pin 10 to pin 17 are port 3 pins which are also referred to as P3.0 to P3.7. These pins are similar to port 1 and can be used as universal input or output pins. These pins are bidirectional pins. These pins also have some additional functions which are as follows:
- **P3.0 (RXD) :** 10th pin is RXD (serial data receive pin) which is for serial input. Through this input signal microcontroller receives data for serial communication.
- **P3.1 (TXD) :** 11th pin is TXD (serial data transmit pin) which is serial output pin. Through this output signal microcontroller transmits data for serial communication.
- **P3.2 and P3.3 (INT0', INT1') :** 12th and 13th pins are for External Hardware Interrupt 0 and Interrupt 1 respectively. When this interrupt is activated(i.e. when it is low), 8051 gets interrupted in whatever it is doing and jumps to the vector value of the interrupt (0003H for INT0 and 0013H for INT1) and starts performing Interrupt Service Routine (ISR) from that vector location.
- **P3.4 and P3.5 (T0 and T1) :** 14th and 15th pin are for Timer 0 and Timer 1 external input. They can be connected with 16 bit timer/counter.
- **P3.6 (WR') :** 16th pin is for external memory write i.e. writing data to the external memory.
- **P3.7 (RD') :** 17th pin is for external memory read i.e. reading data from external memory.
- **Pin 18 and Pin 19 (XTAL2 And XTAL1) –** These pins are connected to an external oscillator which is generally a quartz crystal oscillator. They are used to provide an external clock frequency of 4MHz to 30MHz.
- **Pin 20 (GND) –** This pin is connected to the ground. It has to be provided with 0V power supply. Hence it is connected to the negative terminal of the power supply.
- **Pin 21 to Pin 28 (Port 2) –** Pin 21 to pin 28 are port 2 pins also referred to as P2.0 to P2.7. When additional external memory is interfaced with the 8051 microcontroller, pins of port 2 act as higher-order address bytes. These pins are bidirectional.
- **Pin 29 (PSEN) –** PSEN stands for Program Store Enable. It is output, active-low pin. This is used to read external memory. In 8031 based system where external ROM holds the program code, this pin is connected to the OE pin of the ROM.
- **Pin 30 (ALE/ PROG) –** ALE stands for Address Latch Enable. It is input, active-high pin. This pin is used to distinguish between memory chips when multiple memory chips are used. It is also used to de-multiplex the multiplexed address and data signals available at port 0. During flash programming i.e. Programming of EPROM, this pin acts as program pulse input (PROG).

- **Pin 31 (EA/ VPP) –** EA stands for External Access input. It is used to enable/disable external memory interfacing. In 8051, EA is connected to Vcc as it comes with on-chip ROM to store programs. For other family members such as 8031 and 8032 in which there is no on-chip ROM, the EA pin is connected to the GND.
- **Pin 32 to Pin 39 (Port 0) –** Pin 32 to pin 39 are port 0 pins also referred to as P0.0 to P0.7. They are bidirectional input/output pins. They don't have any internal pull-ups. Hence, 10 K? pull-up registers are used as external pull-ups. Port 0 is also designated as AD0-AD7 because 8051 multiplexes address and data through port 0 to save pins.
- **Pin 40 (VCC) –** This pin provides power supply voltage i.e. +5 Volts to the circuit.

The pin diagram of the 8051 microcontroller is as follows:

1. **VCC:** This pin is connected to the power supply and provides a voltage of +5V to the microcontroller.
2. **GND:** This pin is connected to the ground and serves as the reference voltage for the microcontroller.
3. **XTAL1 and XTAL2:** These pins are used for connecting an external crystal oscillator that provides the clock signal for the microcontroller.
4. **Reset:** This pin is used for resetting the microcontroller. A high pulse on this pin resets the microcontroller to its initial state.
5. **Port 1:** This is an 8-bit bidirectional input/output port that can be used for interfacing with external devices.
6. **Port 2:** This is an 8-bit bidirectional input/output port that can be used for interfacing with external devices.
7. **Port 3:** This is an 8-bit bidirectional input/output port that can be used for interfacing with external devices.
8. **Port 4:** This is an 8-bit bidirectional input/output port that can be used for interfacing with external devices.
9. **INT0:** This is an external interrupt 0 input pin.
10. **INT1:** This is an external interrupt 1 input pin.
11. **T0:** This is an external timer 0 input pin.
12. **T1:** This is an external timer 1 input pin.
13. **WR:** This is the write signal for external memory.
14. **RD:** This is the read signal for external memory.
15. **ALE**: This is the address latch enable signal that is used to latch the address for external memory.
16. **PSEN**: This is the program store enable signal that is used for accessing the program memory.
17. **17-24. Address bus**: These pins are used for transmitting the address information to the external memory.
18. **25-32. Data bus:** These pins are used for transmitting the data between the microcontroller and external memory.

19. **RST:** This is the output signal that indicates the microcontroller is being reset.
20. **EA:** This is the external access enable signal that is used for selecting the program memory.
21. **VPP:** This pin is used for programming the microcontroller.
22. **36-39. XTAL:** These pins are used for connecting an external crystal oscillator.

P1.0 (AD0) – P1.7 (AD7): These pins are used for interfacing with external analog devices

1. Timer Mode Control Register (TMOD):

TMOD can be considered to be two duplicate 4-bit registers, each of which controls the action of one of the timers. The "Timer" or "Counter" function is selected by control bits C/T, and in different operating modes, which are selected by bit-pairs (M1, M0) in TMOD.

MSB LSB

GATE	C / T	M1	M0	GATE	C / T	M1	M0

Timer 1 Timer 0

GATE		Gating control when set. Counter "x" is enabled only while "INTx" pin is high and "TRx" control pin is set. When cleared Timer "x" is enabledwhenever "TRx" control bit is set.
C/T		Timer or Counter Selector cleared for Timer operation (input from internal system clock.) Set for Counter operation (input from "Tx" input pin).
M_1	**M_0**	**OPERATI0N**
0	**0**	13-bit Timer/Counter 5-bits of "TLx" and 8-bits of "THx" are used.
0	**1**	16-bit Timer/Counter 8-bits of "TLx" and 8-bits of "THx" are cascaded.
1	**0**	8-bit auto-reload Timer/Counter "THx" holds a value which is to be reloaded into "TLx" each time it overflows.
1	**1**	(Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standard Timer0 control bits. TH0 is an 8-bit timer only controlled by Timer 1 control bits. Timer/Counter 1 stopped.

2. Interrupt Enable (IE) Register :

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EA	x	x	ES	ET1	EX1	ET0	EX0

Symbol	Name and Function
EA	Enable All. If 0, Disables all interrupts and no interrupt is acknowledged. If1, each interrupt can be individually enabled or disabled by programming appropriate bit.
x	Reserved
x	-
ES	Enable Serial Interrupt. If 1, enables TI or RI to generate interrupt.

ET1	Enable Timer 1 interrupt. If 1, Enables the TF1 to generate the interrupt.
EX1	Enable External interrupt 1. If 1, Enables the INT1 to generate the interrupt.
ET0	Enable Timer 0 interrupt. If 1, Enables the TF0 to generate the interrupt.
EX0	Enable External interrupt 0. If 1, Enables the INT0 to generate the interrupt.

3. Timer Control Register (TCON):

TCON has control bits and flags for the timers in the upper nibble, and control bitsand flags for the external interrupts in lower nibble.

MSB LSB

TF1	TR1	TF 0	TR 0	IE1	IT1	IE0	IT0

Bit	Symbol	Function
TCON.7	TF1	Timer 1 overflow flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors to interrupt routine, or clearing the bit in software.
TCON.6	TR1	Timer 1 Run control bit. Set/cleared by software to turn Timer/Counter on/off.
TCON.5	TF0	Timer 0 overflow flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors to interrupt routine, or by clearing the bit in software.
TCON.4	TR0	Timer 0 Run control bit. Set/cleared by software to turn Timer/Counter on/off.
TCON.3	IE1	Interrupt 1 Edge flag. Set by hardware when external interruptsedge detected. Cleared when interrupt processed.
TCON.2	IT1	Interrupt 1 type control bit. Set/cleared by software to specifyfalling edge/low level triggered external interrupts.
TCON.1	IE0	Interrupt 0 Edge flag. Set by hardware when external interruptsedge detected. Cleared when interrupt processed.
TCON.0	IT0	Interrupt 0 Type control bit. Set/cleared by software to specify falling edge/low Level triggered external interrupts.

4. Interrupt Priority (IP) Register:

Each source of the interrupt can be individually programmed to be in either of the two priority levels. The priorities can be assigned to each interrupt by programming appropriate bits in the SFR Interrupt Priority Register.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
x	x	x	PS	PT1	PX1	PT0	PX0

Symbol	Name and Function
x	Reserved
PS	Priority of Serial Interrupt. If 1, Priority of Serial Interrupt is higher
PT1	Priority of Timer 1 interrupt. If 1, Priority of Timer 1 interrupt is higher
PX1	Priority of External interrupt 1. If 1, Priority of the INT1 is higher
PT0	Priority of Timer 0 interrupt. If 1, Priority of Timer 0 Interrupt is higher
PX0	Priority of External interrupt 0. If 1, Priority of the INT0 is higher

5. Serial Port Control Register (SCON):

The serial port control and status register is the Special Function Register **SCON.** This register contains not only the mode selection bits, but also the 9th data bit for transmit and receive (TB8 and RB8) and the serial port interrupt bits (TI and RI).

MSB LSB

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Where SM0, SM1 specify the serial port mode, as follows:

SM0	SM1	Mode	Description	Baud Rate
0	0	0	Shift register	f osc / 12
0	1	1	8-bit UART	Variable
1	0	2	9-bit UART	f osc / 64 or fosc /32
1	1	3	9-bit UART	variable

SM2	Enables the multiprocessor communication feature in Modes 2 and 3. In Mode 2 or 3,if SM2 is set to 1, then Rl will not be activated if the received 9th data bit (RB8) is 0. In Mode 1, if SM2=1 then RI will not be activated if a valid stop bit was not received. In Mode 0, SM2 should be 0.
REN	Enables serial reception. Set by software to enable reception. Clear by software to disable reception.
TB8	The 9th data bit that will be transmitted in Modes 2 and 3. Set or clear by software as desired.
RB8	In Modes 2 and 3, is the 9th data bit that was received. In Mode 1, it SM2=0, RB8 is the stop bit that was received. In Mode 0, RB8 is not used.
TI	Transmit interrupt flag. Set by hardware at the end of the 8th bit time in Mode 0, or at the beginning of the stop bit in the other modes, in any serial transmission. Must becleared by software only.
RI	Receive interrupt flag. Set by hardware at the end of the 8th bit time in Mode 0, or halfway through the stop bit time in the other modes, in any serial reception (except see SM2). Must be cleared by software only.