LABORATORY MANUAL FOR THE COURSE

MICROCONTROLLERS & EMBEDDED SYSTEMS LABORATORY

(ECE 328)

(Mr. N.Srinivasa Naidu)
Signature of the Physical Lab Incharge:

(Dr. V.Rajya Lakshmi)
Signature of the HOD:

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY & SCIENCES(A)
(Affiliated to AU, Approved by AICTE &Accredited by NBA)
Sangivalasa-531162, Bheemunipatnam Mandal, Visakhapatnam Dt.
Phone: 08933- 225084,226395

MICROCONTROLLER & EMBEDDED SYSTEMS LABORATORY	
ECE328	**Credits:2**
Instruction: 3 Lab periods	Sessional Marks:50
End Exam: 3 Hours	End Exam Marks:50

Prerequisites:

Microprocessors and Interfacing, Microcontroller & Embedded Systems

COURSE OBJECTIVES

- To program both 8051 to meet the requirements of the user.
- To interface various peripherals
- To handle interrupts
- To design a microcomputer to meet the requirement of the user

COURSE OUTCOMES

At the end of the course student will be able to	
1.	Program 8051 microcontroller to meet the requirements of the user.
2.	Interface peripherals like switches, LEDs, stepper motor, Traffic lights controller, etc..,
3.	Handle interrupts
4.	Design a microcontroller development board to meet the requirements of the user

Mapping of Course Outcomes with Program Outcomes:

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

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

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.

PROGRAM SPECIFIC OUTCOMES

PSO1: Professional Skills: An ability to apply the knowledge of mathematics, science, engineering fundamentals in ECE to various areas, like Analog & Digital Electronic Systems, Signal & Image Processing, VLSI & Embedded systems, Microwave & Antennas, wired & wireless communication systems etc., in the design and implementation of complex systems.

PSO2: Problem-Solving Skills: An ability to solve complex Electronics and communication engineering problems, using latest hardware and software tools, along with significant analytical knowledge in Electronics and Communication Engineering

PSO3: Employability and Successful career: Acquire necessary soft skills, aptitude and technical skills to work in the software industry and/or core sector and able to participate and succeed in competitive examinations.

List of Experiments:

1. Study and familiarization of 8051 Microcontroller trainer kit
2. Assembly Language Program for addition of 8-bit numbers stored in an array
3. Assembly Language Program for Multiplication by successive addition of two 8-bit numbers
4. Assembly Language Program for finding largest no. from a given array of 8-bit numbers
5. Assembly Language program to arrange 8-bit numbers stored in an array in ascending order
6. Stepper motor control by 8051 Microcontroller
7. Interfacing of 8-bit ADC 0809 with 8051 Microcontroller
8. Interfacing of 8-bit DAC 0800 with 8051 Microcontroller and Waveform generation using DAC
9. Implementation of Serial Communication by using 8051 serial ports
10. Assembly Language Program for use of Timer/Counter for various applications
11. Traffic light controller/Real-time clock display
12. Simple test program using ARM 9 mini 2440 kit (Interfacing LED with ARM 9 mini 2440 kit)

NOTE:

1. It is compulsory for each student to create their own Microcontroller Development Board for personal use
2. A student has to perform a minimum of 10 experiments.

Scheme of evaluation for MCES 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 atleast one flowchart, Calculations, Input/Output observations & Result – 1M, Daily Performance 3M)

III. Lab Project – 10M
(It is compulsory for each student to create their own Microcontroller Development Board for personal use based on 8051)

IV. Attendance – 5M

V. Internal End Exam – 20M
(Aim, Apparatus – 2M, Program – 10M (Mnemonics/code – 5M, Relevant Comments – 2M, Algorithm/flow chart – 3M), Calculations, Input/Output observations & Result – 5M, Performance – 3M)

Lab External:

I. Writeup – 10M
(Aim– 2M, Apparatus – 1M, Theory – 2M, Algorithm/flowchart – 5M)

II. Program – 15M
(Mnemonics/Code – 10M, Comments – 3M, Optimization– 2M)

III. Performance – 5M
(Experimentation skill - Connections,.etc)

IV. Result – 10M
(Identifying & Showing the inputs and outputs – 2M and/or theoretical calculations – 2M, Output Verification – 6M (Partial output – 3M, No Output – 0M)

V. Viva – 10M

Experiment-1

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) and software (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 from embedded

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 following table shows the Keil C51 Product Line (across the top) and the Components that are included (along the left side). You may use this information to find the development tool kit that best fits your needs.

Introduction

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:

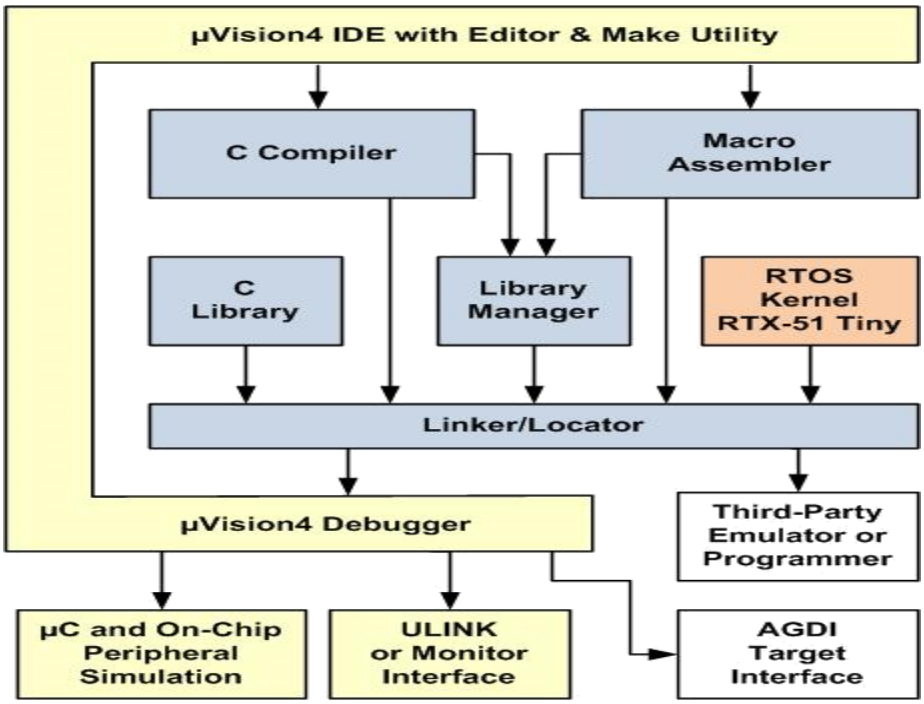
- Software Development Cycle that describes the steps and tools involved to create a project.
- Development Tools that describes the major features of the Keil C51 development tools including the µVision IDE and Debugger.
- Folder Structure that describes the default location of µVision and the C51 tool 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 and organize 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:

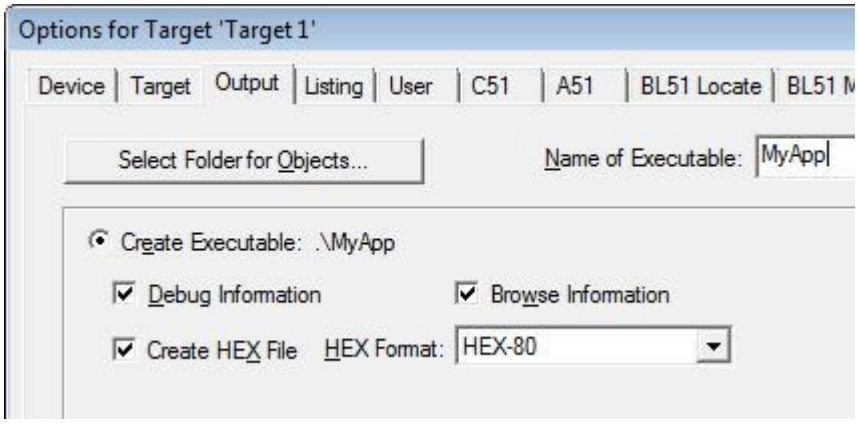
- Use the Keil ULINK USB-JTAG adapter for downloading and testing your program.
- Install a target monitor on your target system and download your program using the built-in monitor interface of the µVision Debugger.
- Use the Advanced GDI interface to attach and use the µVision Debugger front end with 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.

Creation of HEX File

Some applications require a HEX file to download the application software into the physical device using a Flash programming utility. µVision creates HEX files with each build process when Create HEX File is enabled in the dialog Options for Target Output.



If code banking is used, then the application has to be converted with the OC51 Banked Object File Converter prior to using the OH51 Object/Hex converter.

When the extended LX51 Linker is used, it is mandatory to use the OHx51 Extended Object-HEX Converter to generate an Intel HEX-386 file that contains the common area and all the code banks.

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.

Differences Between µVision and C

A number of differences exist between ANSI C and the language subset to support features in user- and signal functions.

- µVision does not differentiate between uppercase and lowercase. The names of objects and control statements may be written in either uppercase or lowercase.

- µVision has no preprocessor. Preprocessor directives like #define, #include, and #ifdef are not supported.

- µVision does not support global declarations. Scalar variables must be declared within a function definition. You may define symbols with the DEFINE command and use them like you would use a global variable.

in µVision, variables may not be initialized when they are declared. Explicit assignment statements must be used to initialize variables.

- µVision functions only support scalar variable types. Structures, arrays, and pointers are not allowed. This applies to the function return type as well as the function parameters.

- µVision functions may only return scalar variable types. Pointers and structures may not be returned.

- µVision functions cannot be called recursively. During function execution, µVision recognizes recursive calls and aborts function execution if one is detected.

- µVision functions may only be invoked directly using the function name. Indirect function calls via pointers are not supported.

- µVision supports only the ANSI style for function declarations with a parameter list. The old K&R format is not supported. For example, the following ANSI style function is acceptable.

High Level Language –‘C’ V/s Embedded C

 ‘C’ is a well-structured, well defined and standardized general purpose programming language with extensive bit manipulation support.

 ‘C’ offers a combination of the features of high level language and assembly and help sinhard ware access programming (system
level programming) as well as business package developments (Application developments like payroll systems, banking applications etc).

 The conventional ‘C’ language follows ANSI standard and it incorporates various library files for different operating systems.

 A platform (Operating System) specific application, known as, compiler is used for the conversion of programs written in ‘C’ to the target processor (on which the OS is running) specific binary files.

 Embedded C can be considered as a subset of conventional ‘C’ language.

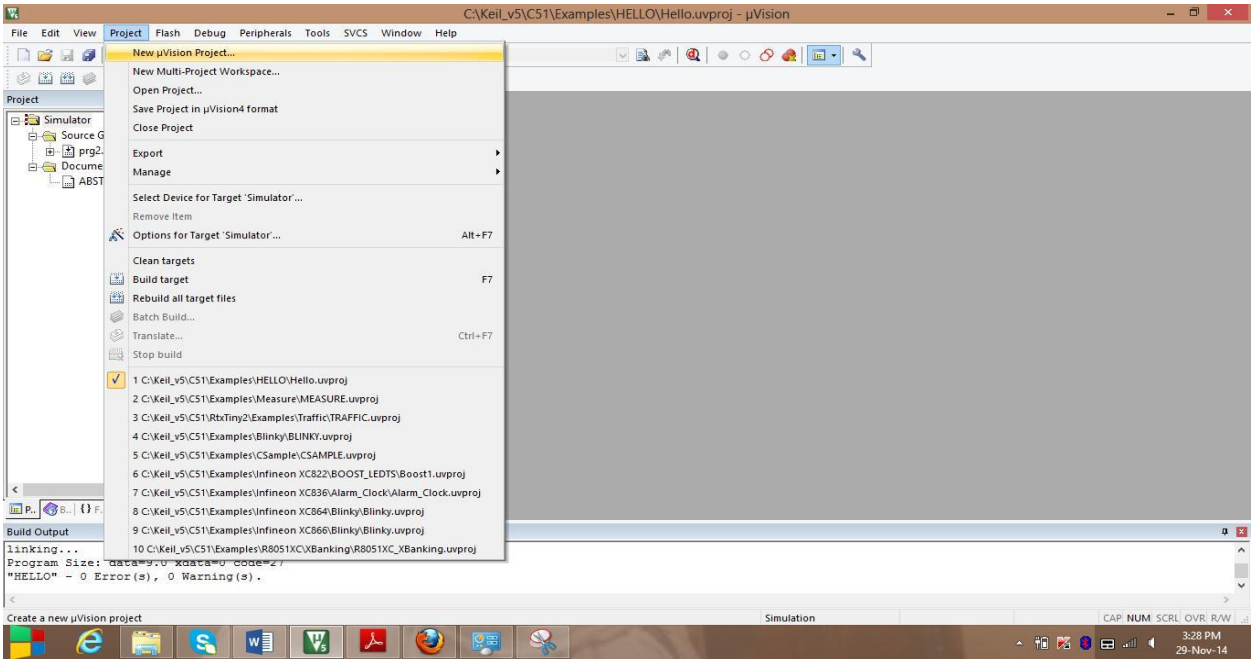
 Embedded C supports all ‘C’ instructions and incorporates a few target processor specific functions /instructions.

 The standard ANSI ‘C’ library implementation is always tailored to the target processor /controller library files in Embedded C.

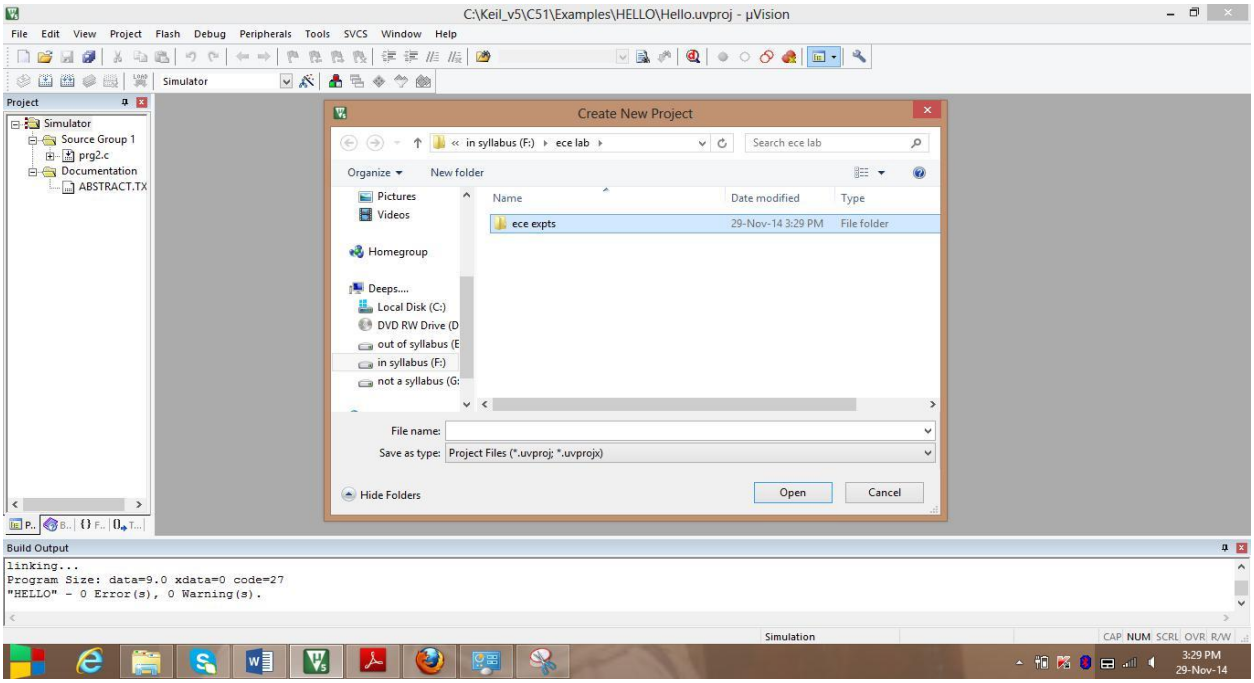
 The implementation of target processor /controller specific functions /instructions depends upon the processor /controller as well as the supported cross-compiler for the particular Embedded C language.

 A software program called ‘Cross-compiler’ is used for the conversion of programs written in Embedded C to target processor /controller specific instructions.

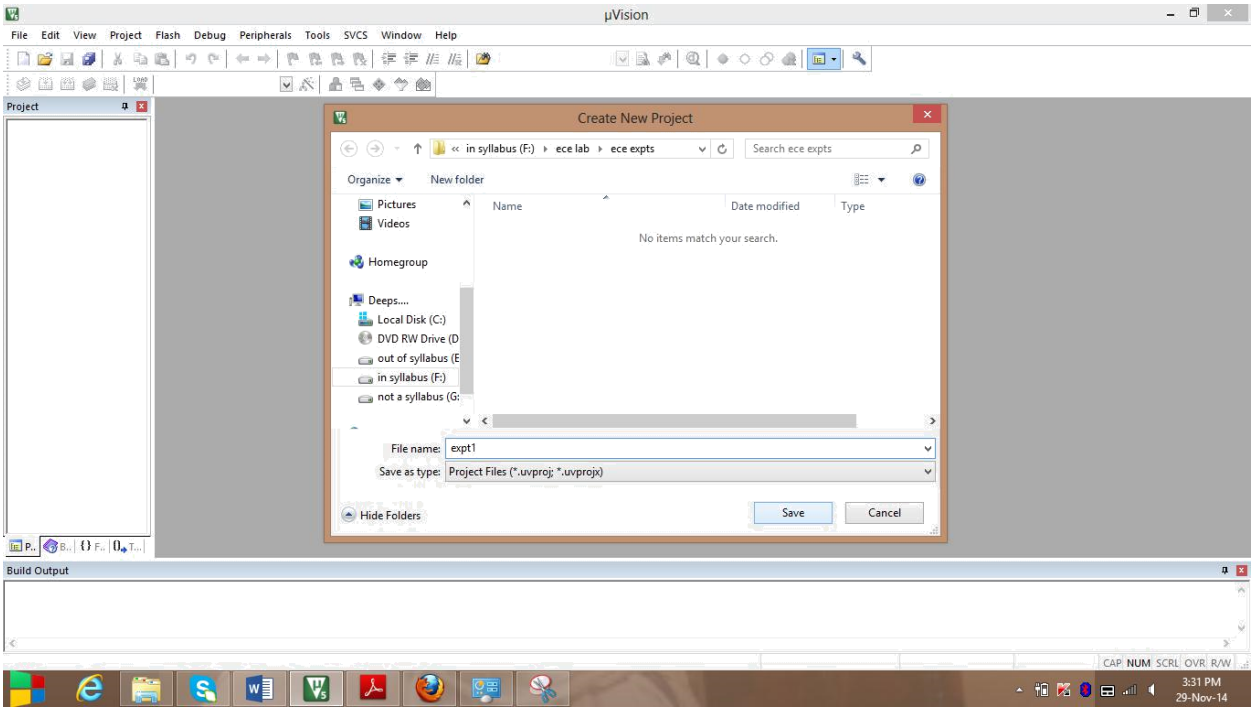
Procedure:-



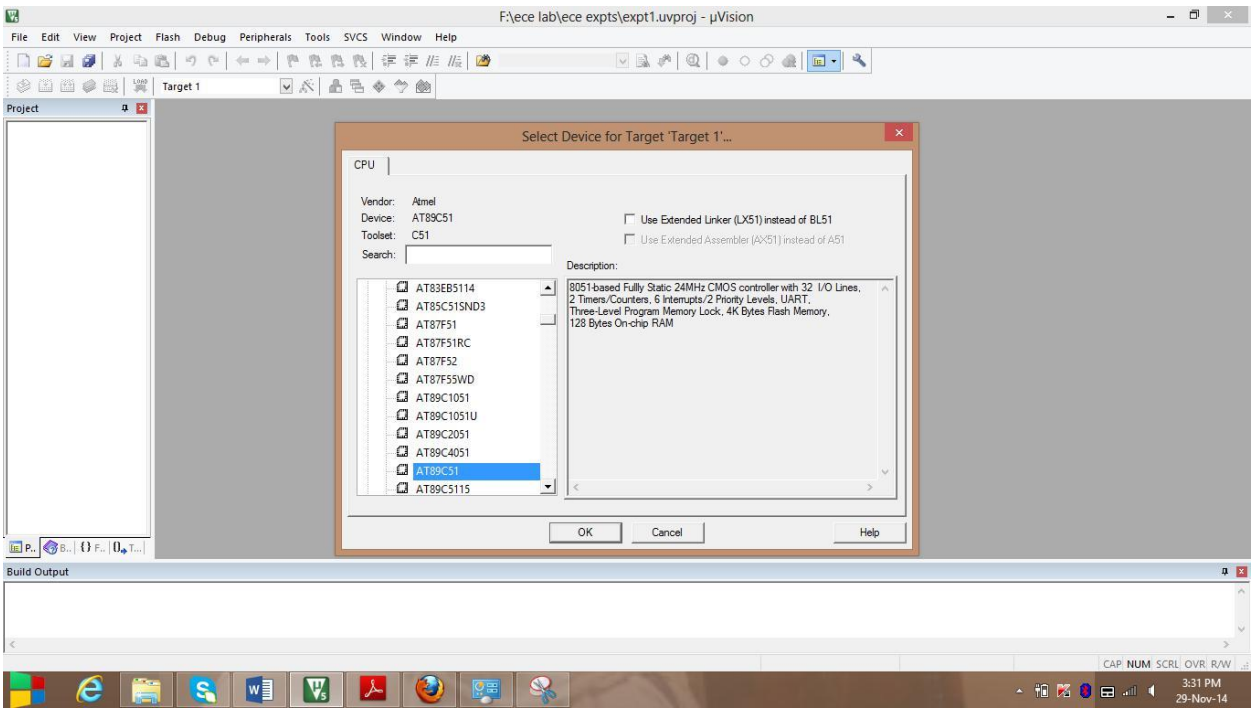
Create new u vision project



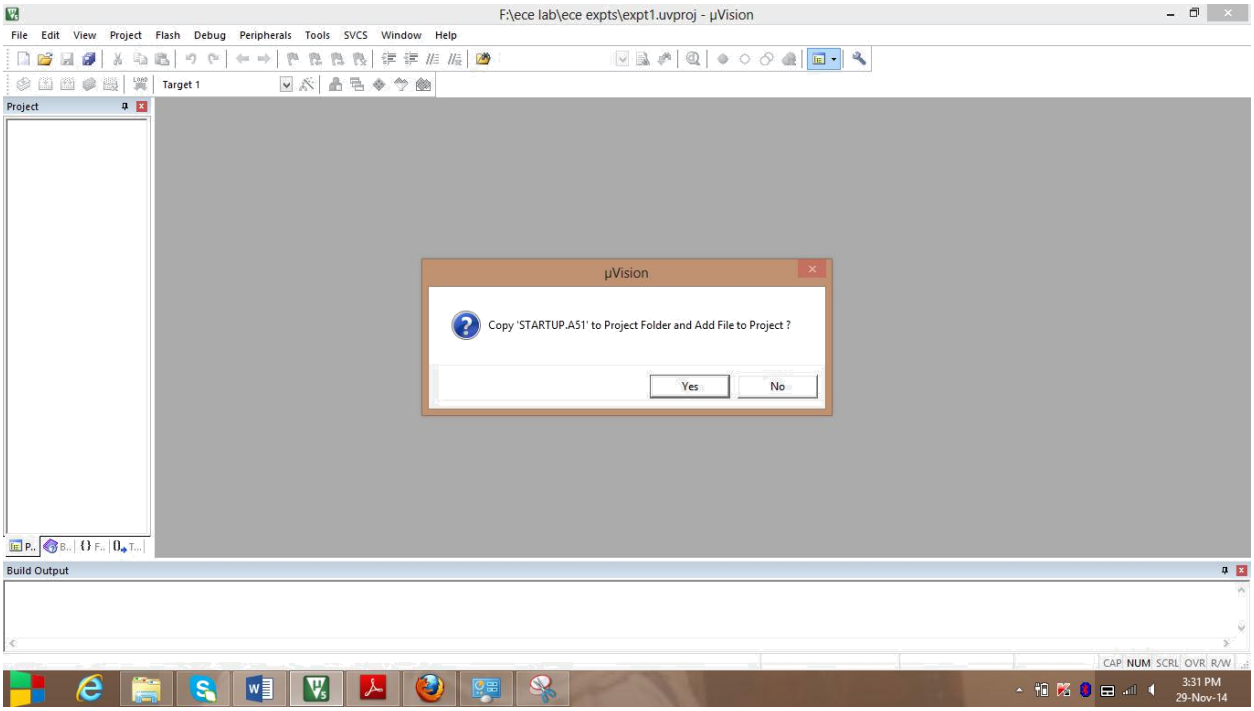
Select the folder (newly created) to save the project



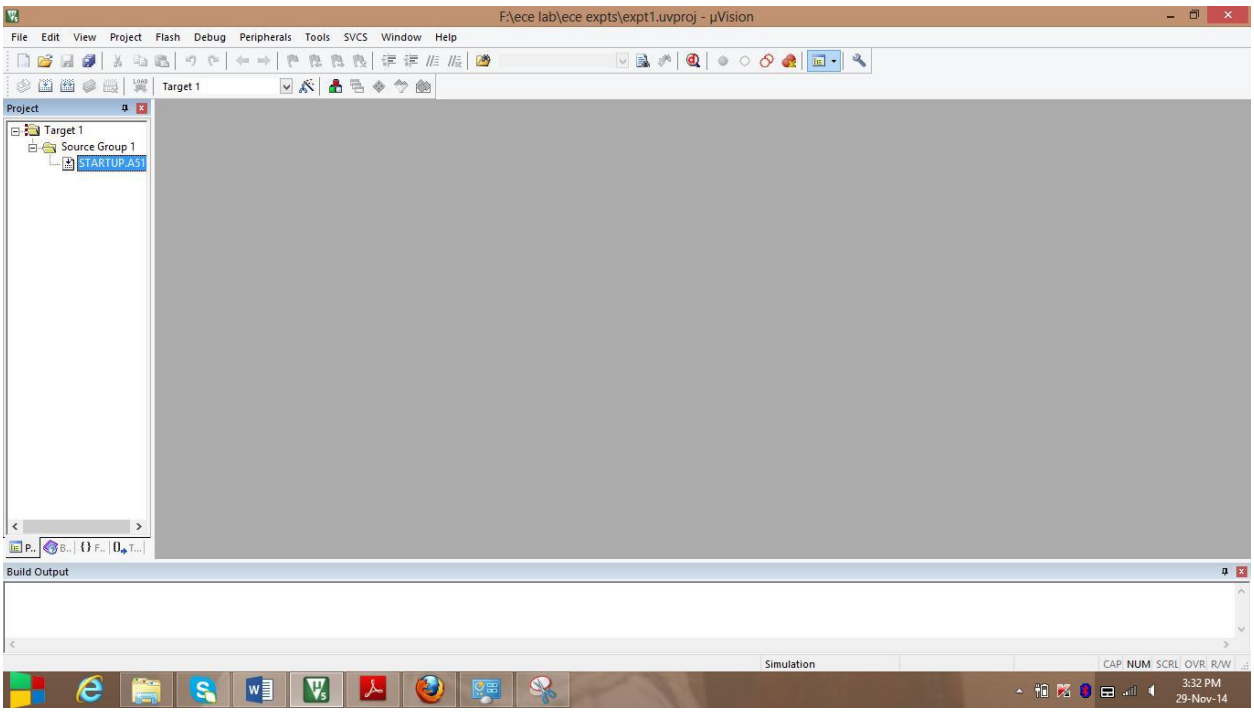
Save the project



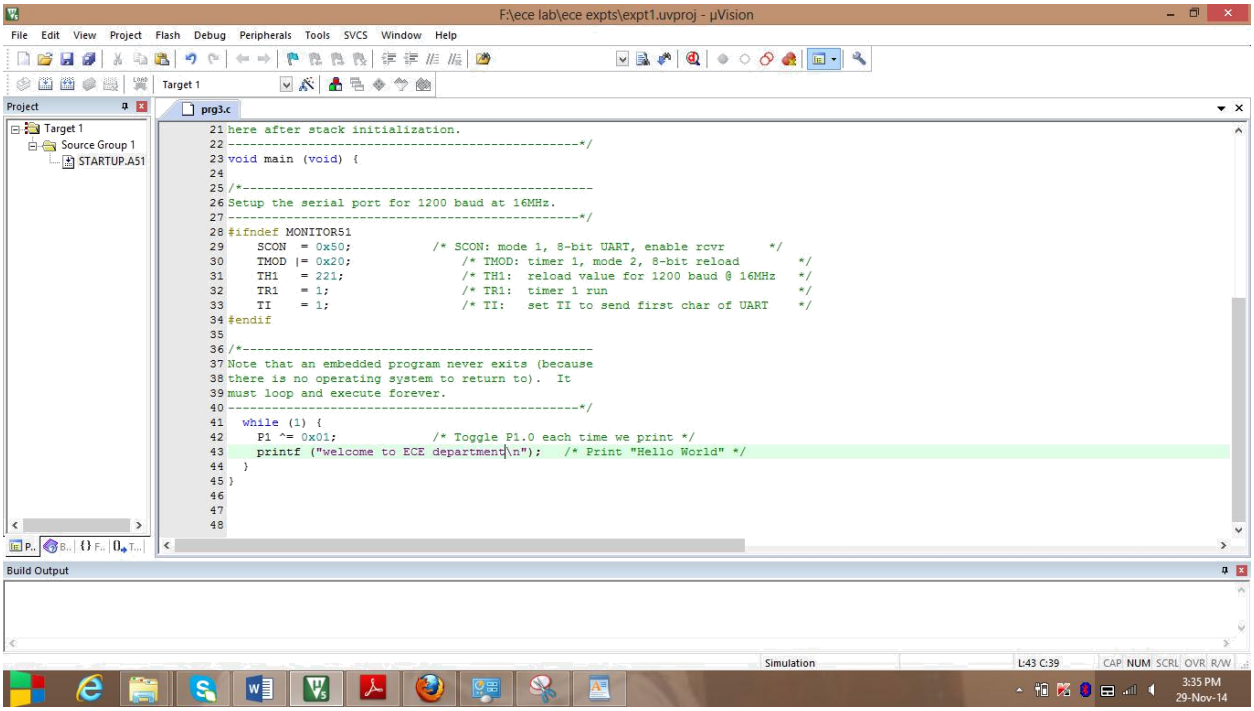
Select the vendor “Atmel” and device “AT89C51”



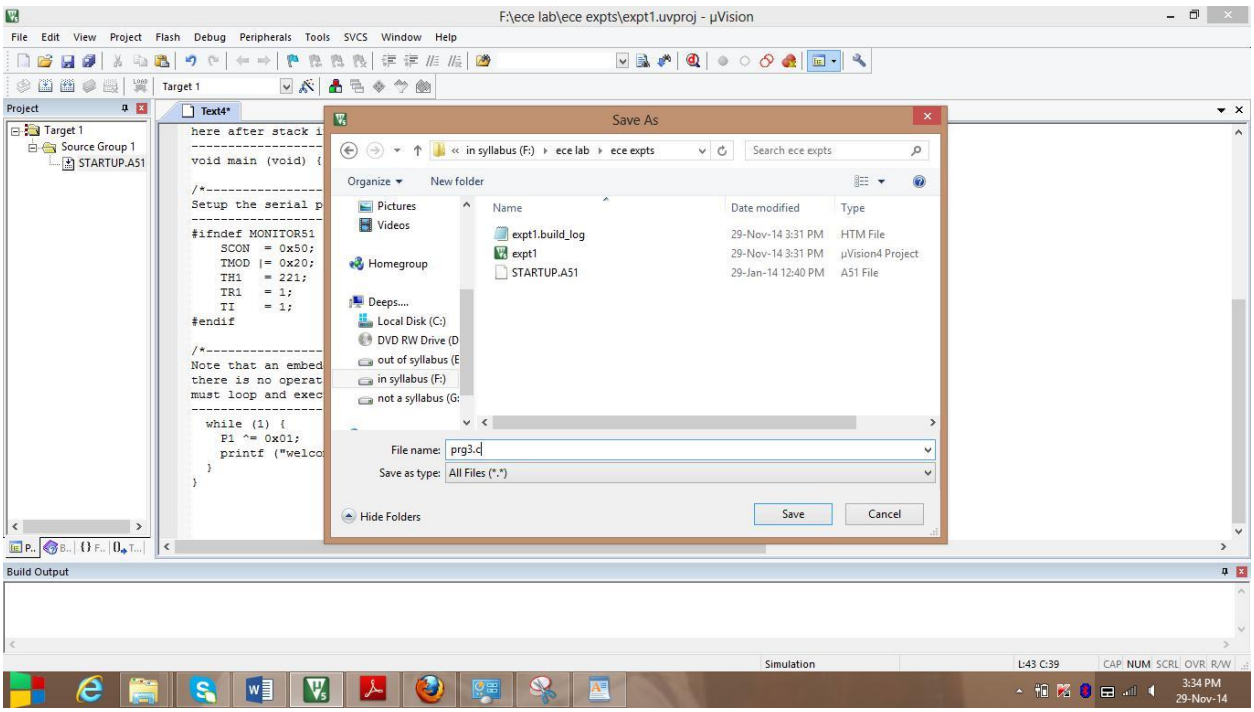
Addition of STARTUP.A51 to project folder



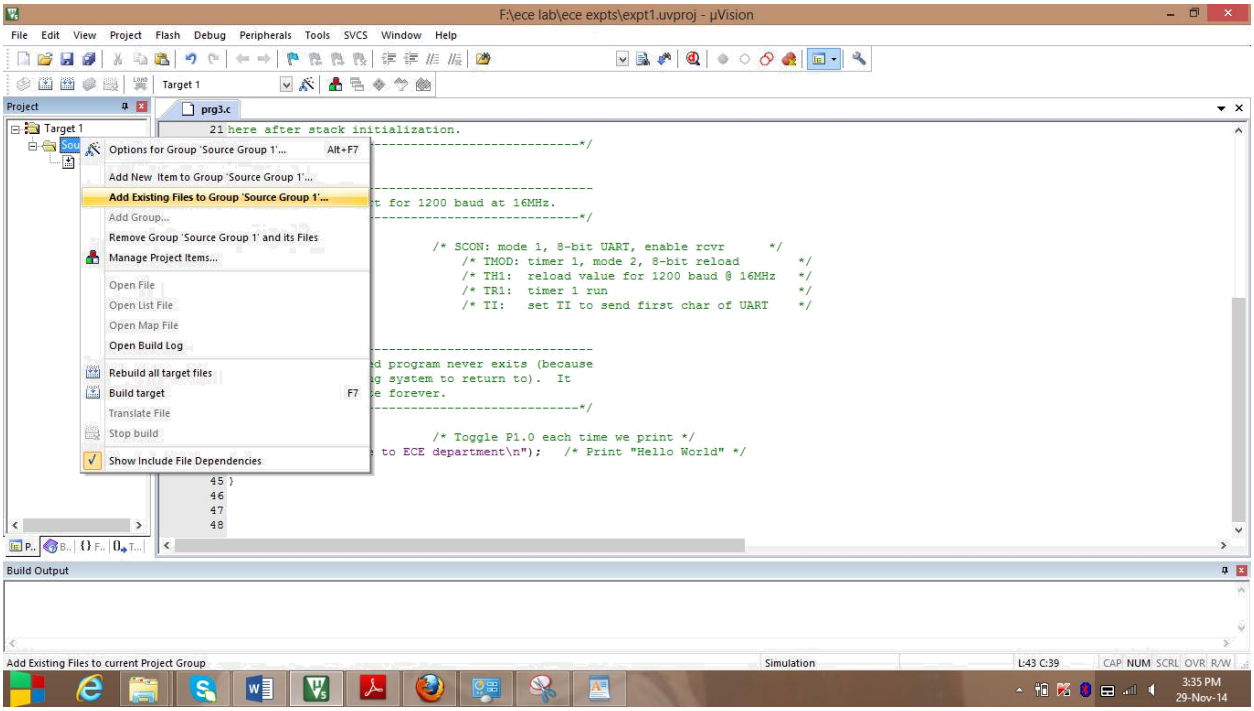
STARTUP.A51 is added



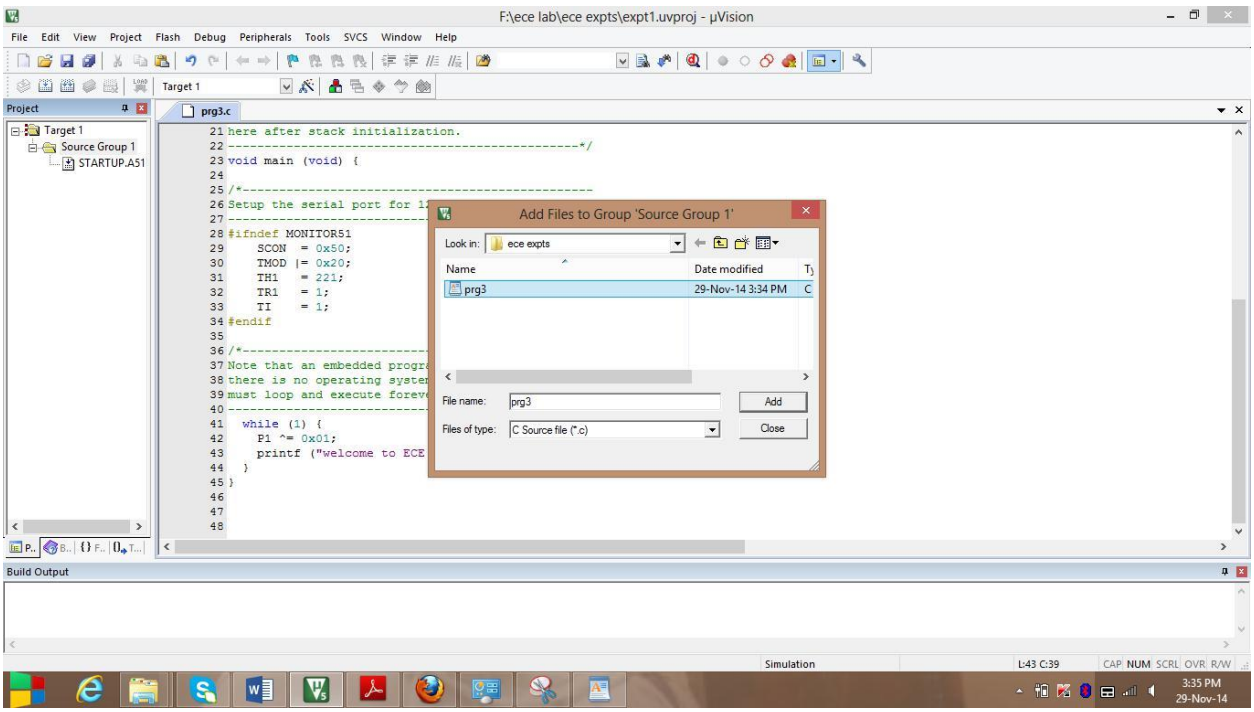
The program to print “welcome to ECE department” is written



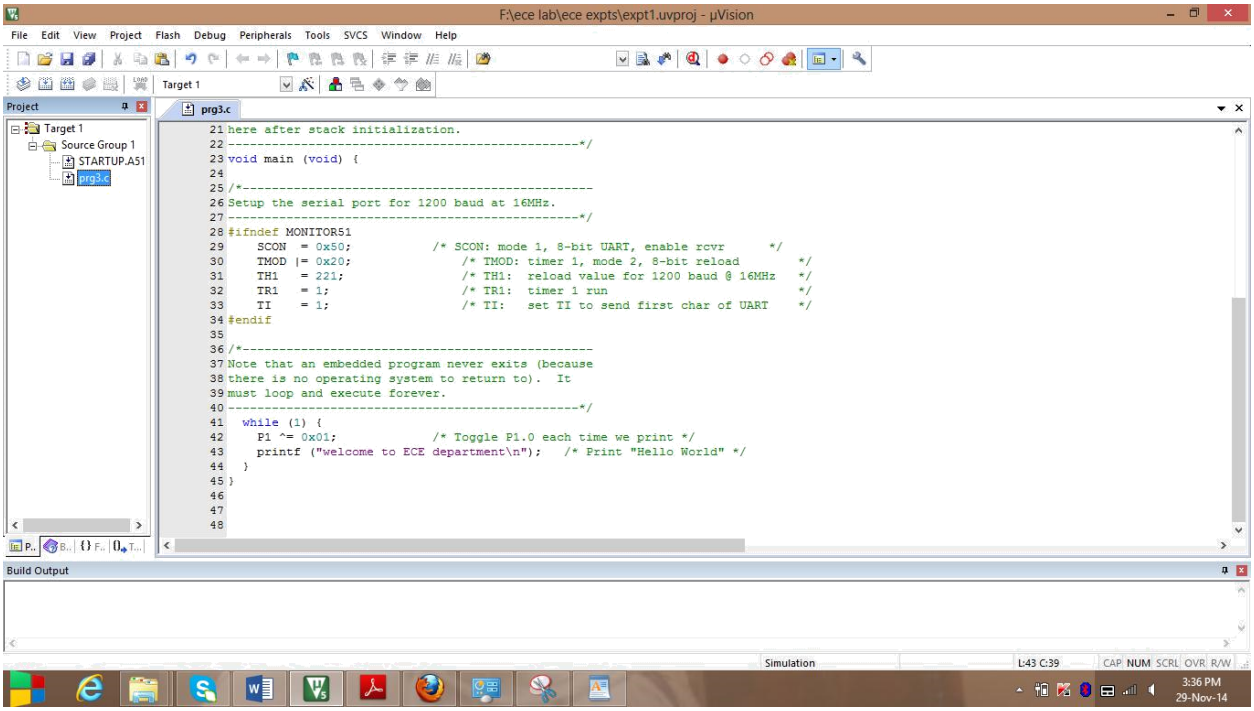
The program is saved as prg3.c



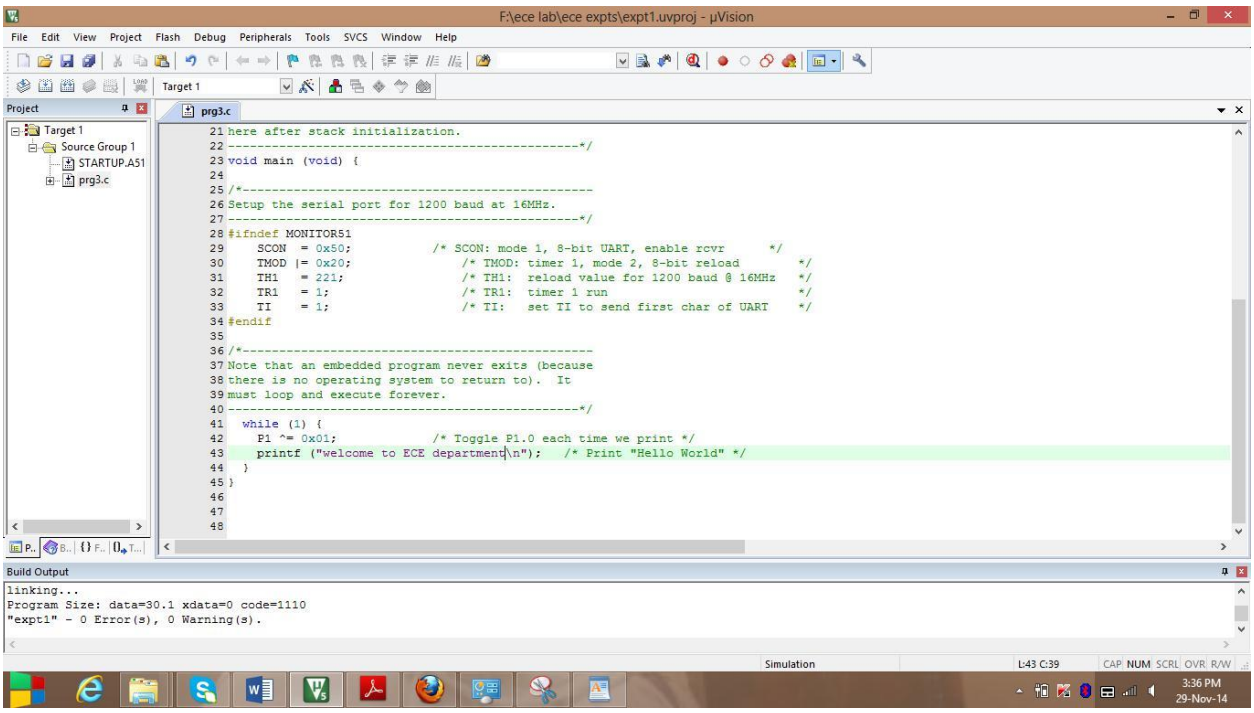
Prg3.c is to be added to Source Group1



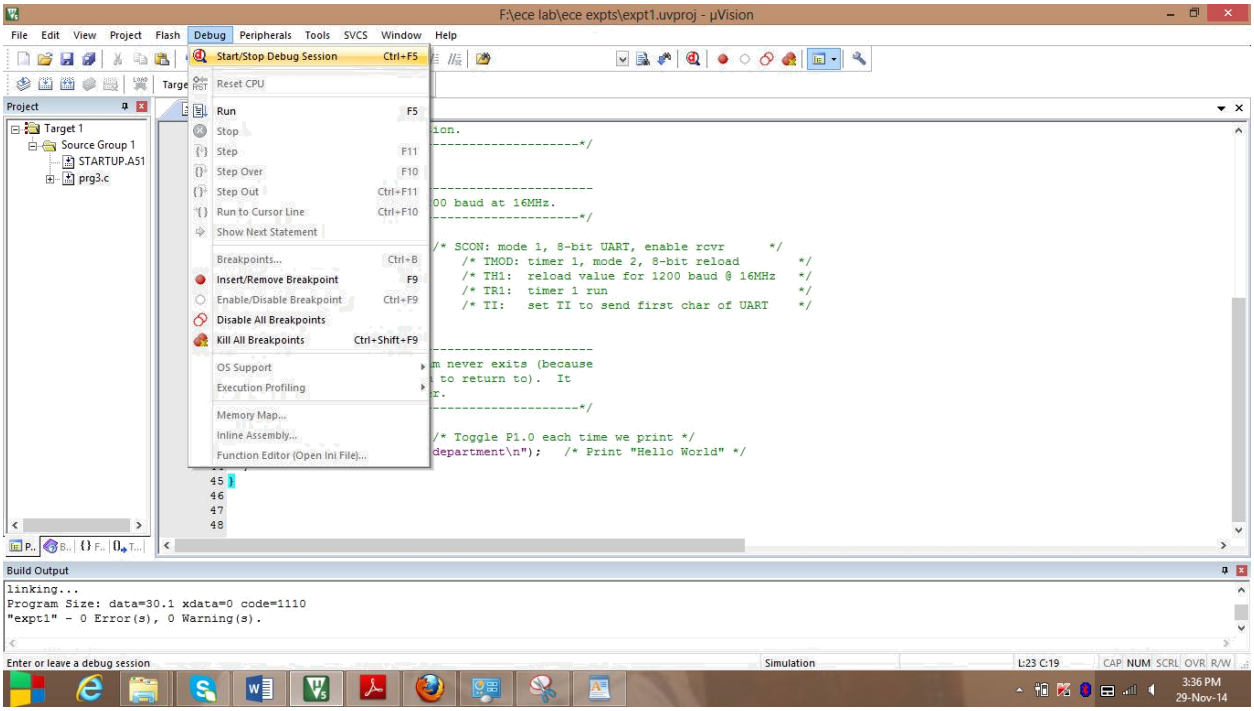
Select the program prg3



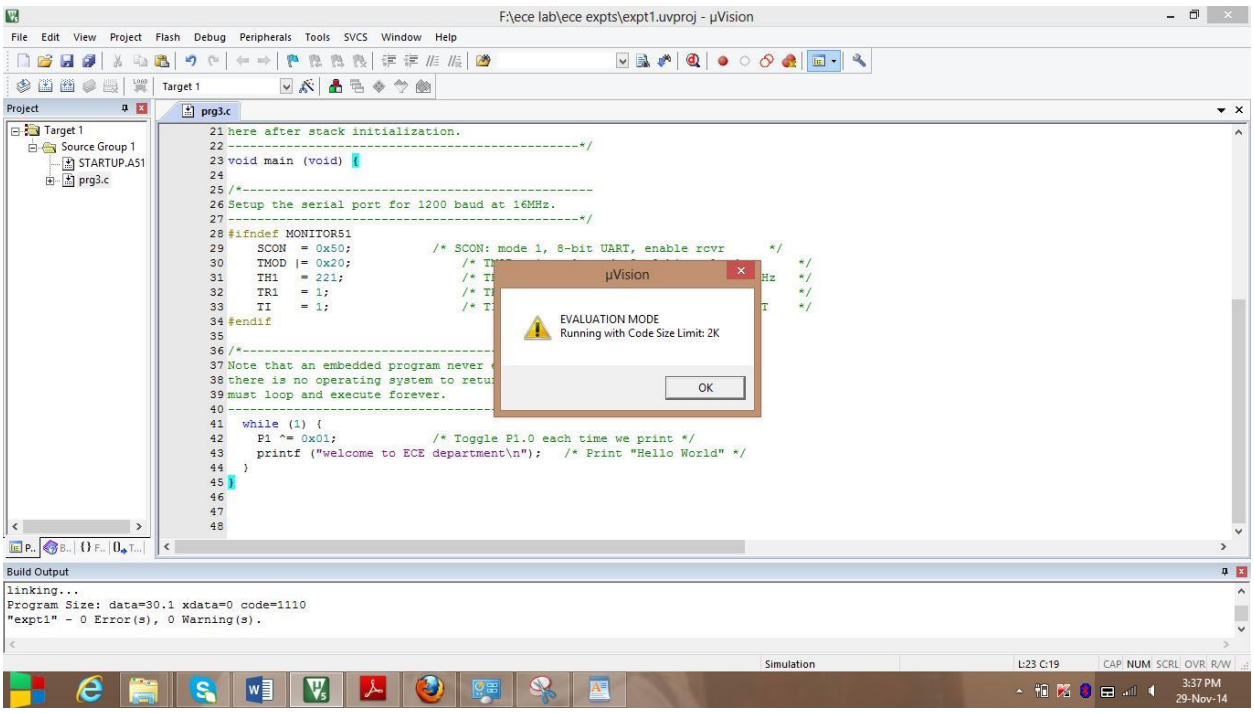
Now prg3.c is added to Source Group 1



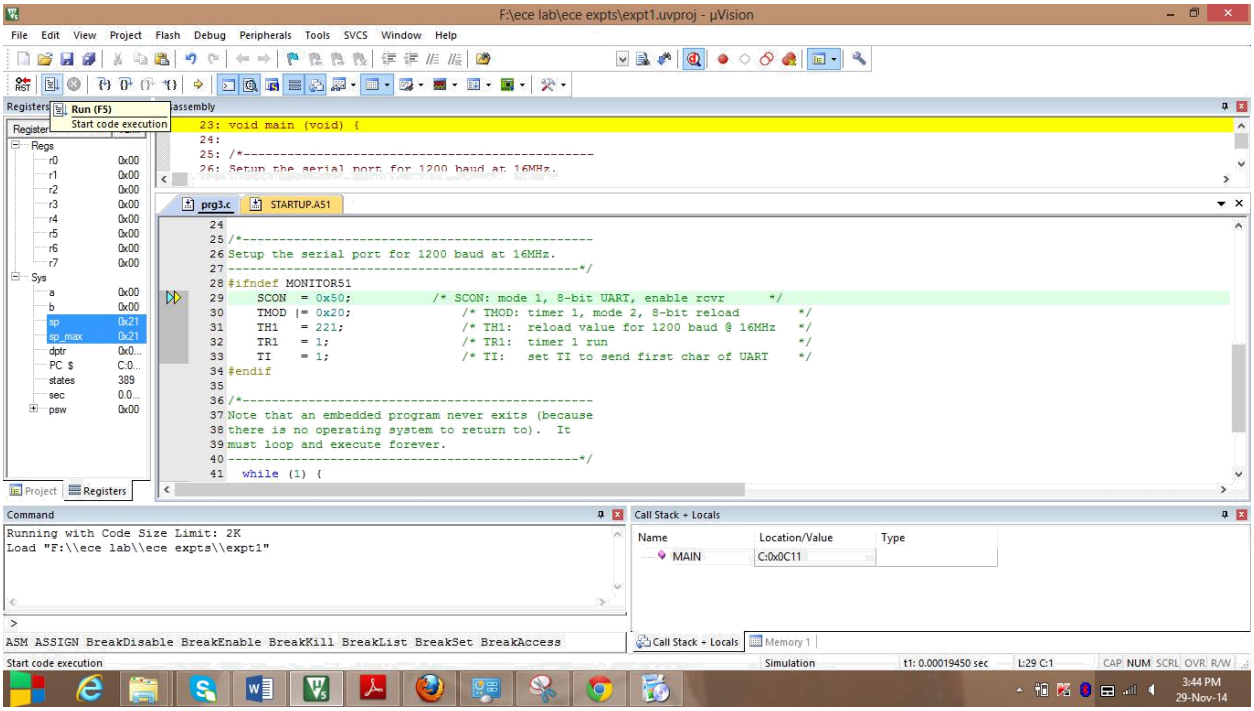
Build the target



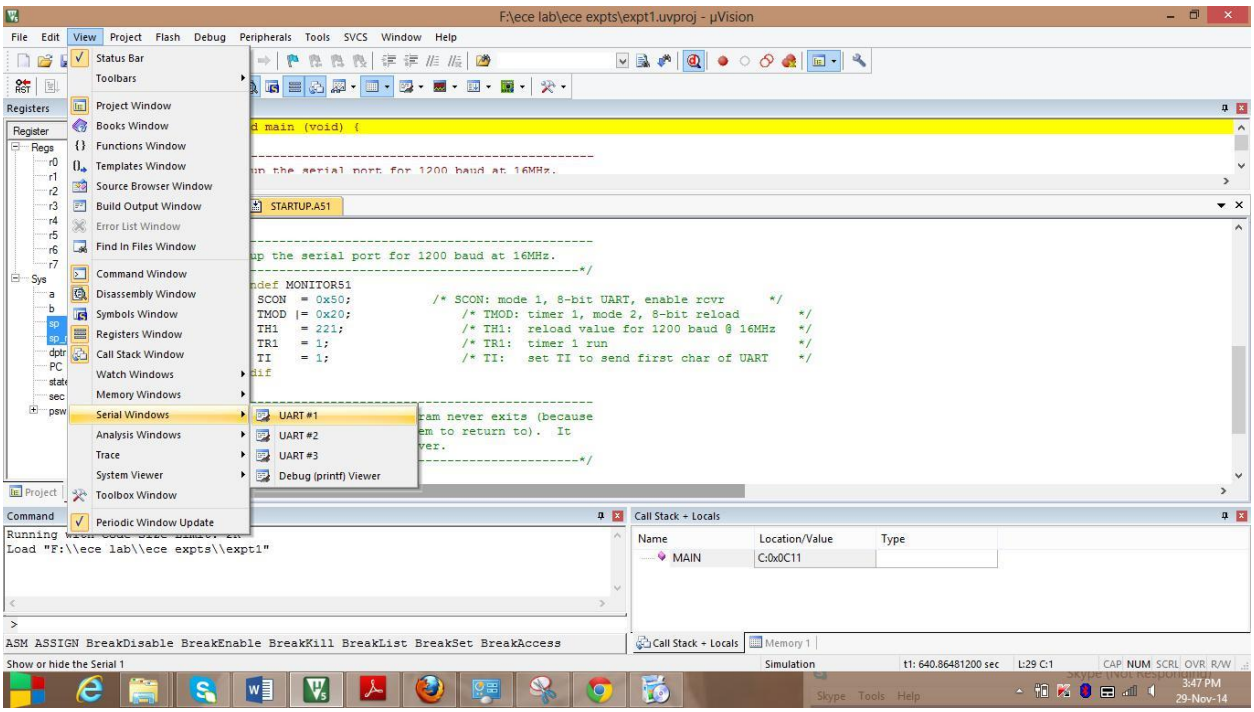
Debugging the target



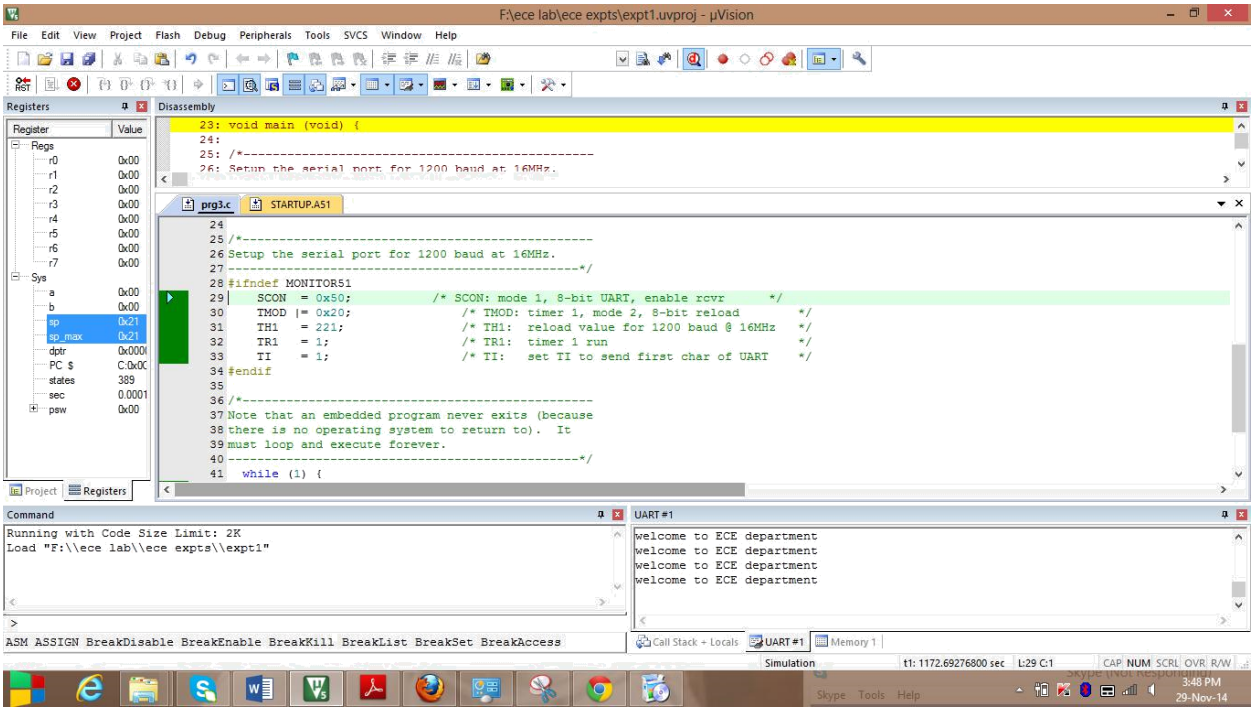
New window evaluation mode appeared. Press ok



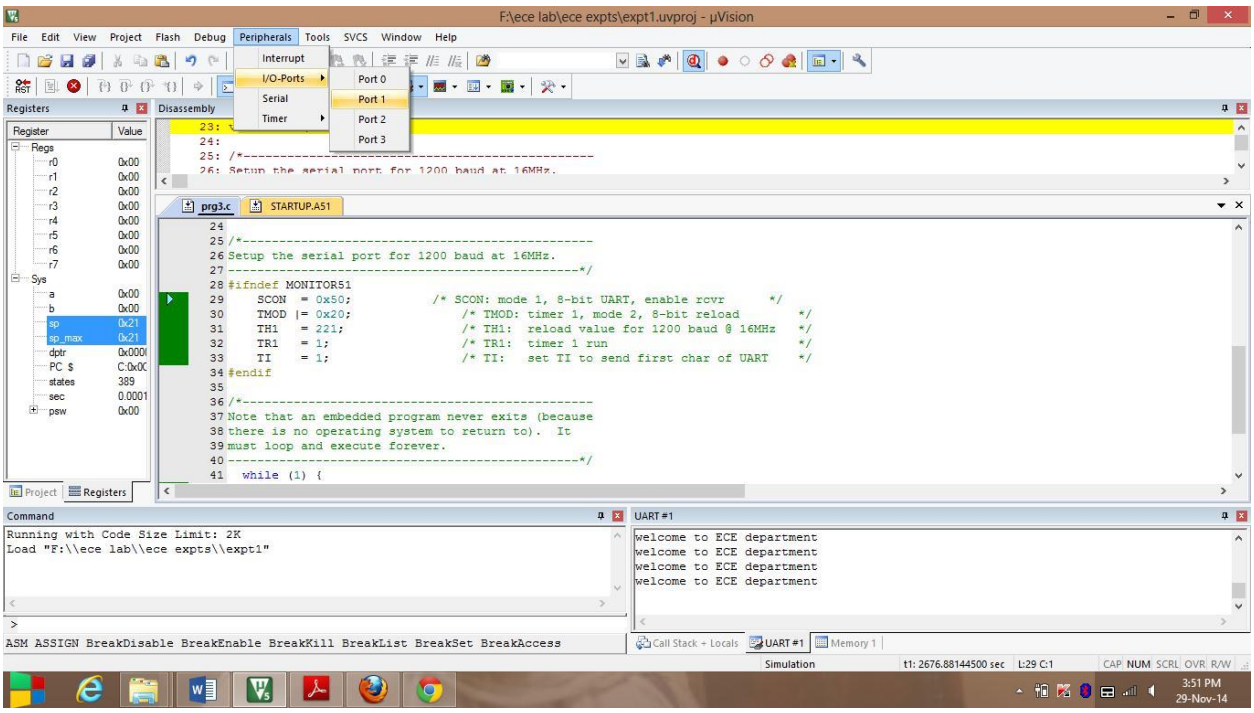
Run the program



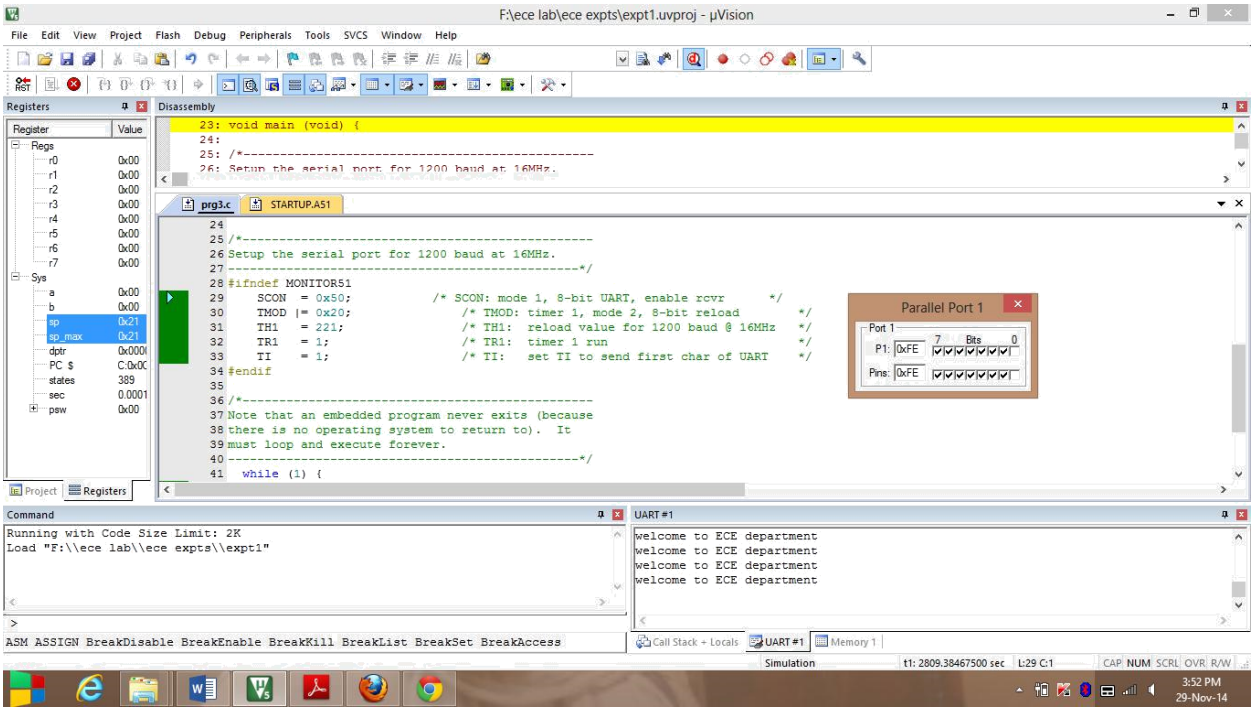
Selecting for UART#1 from serial windows



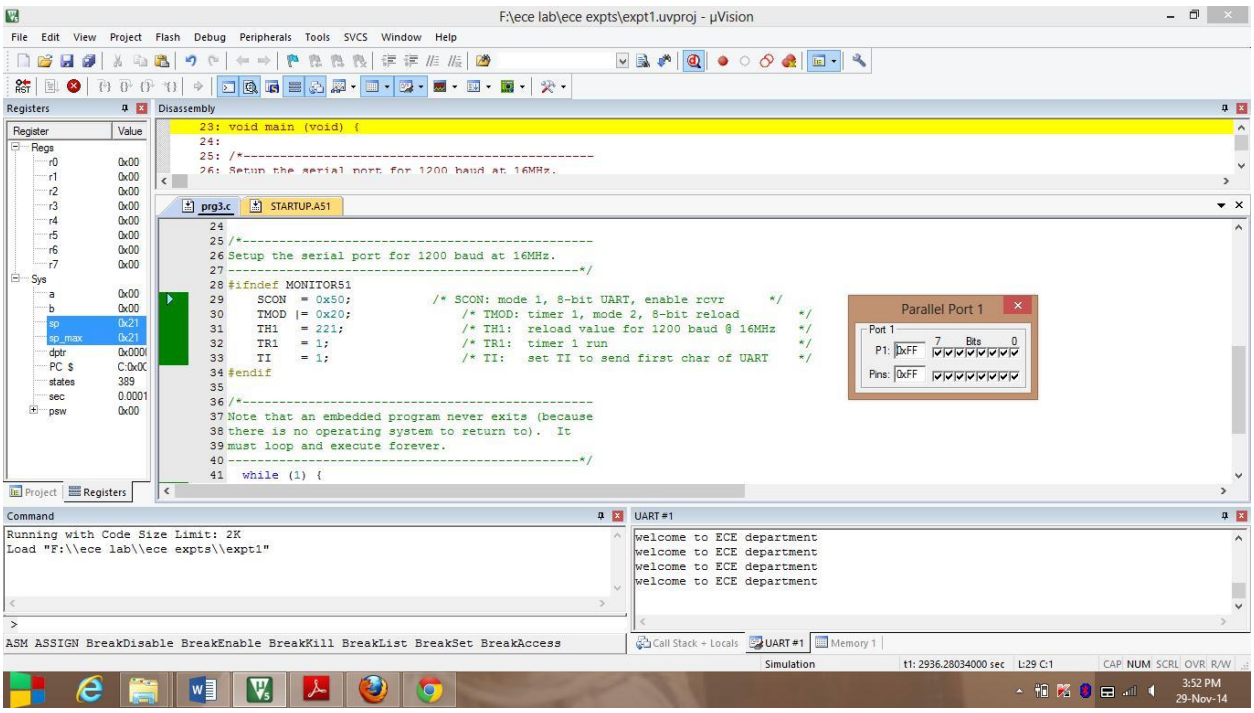
Check the output at UART#1 window



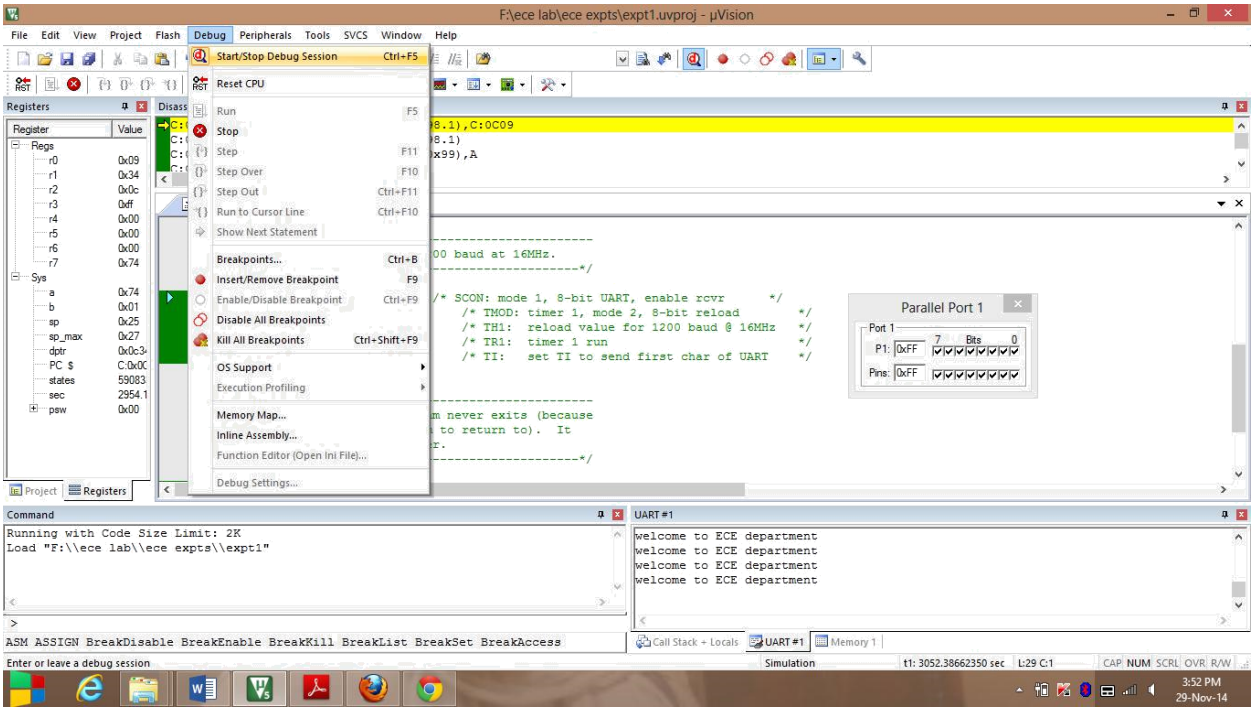
Select Port1 from i/o ports in peripherals



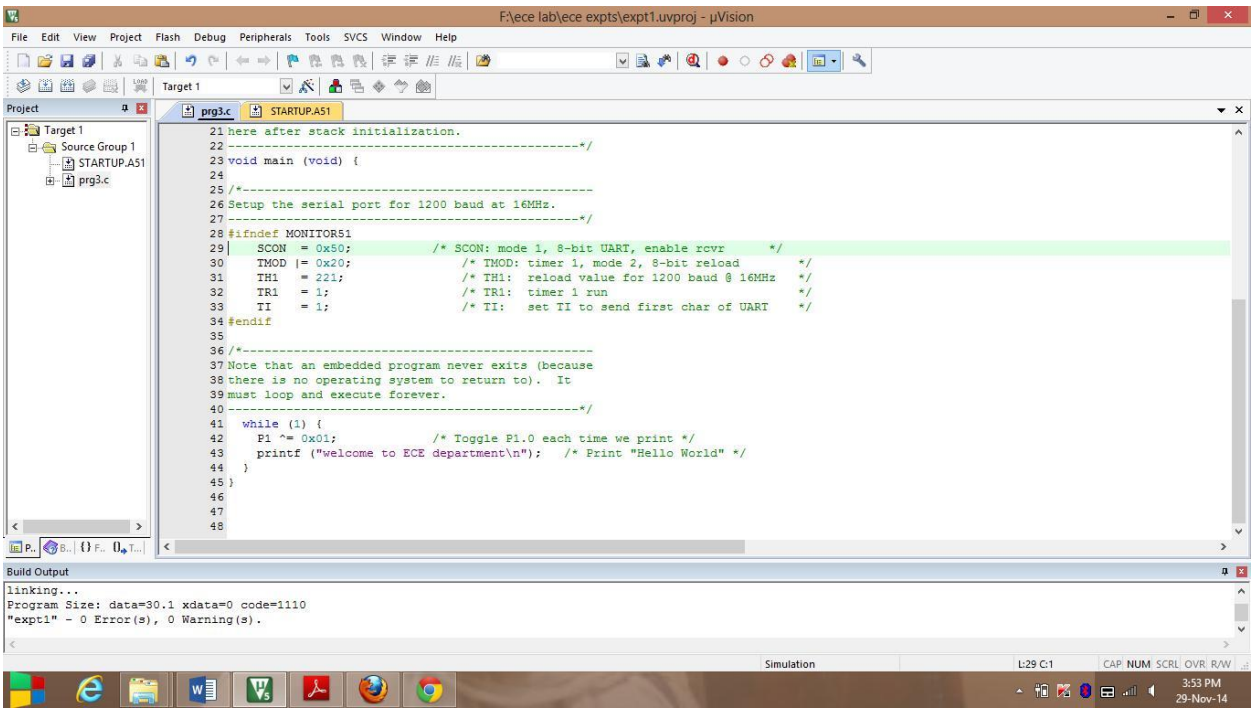
Port1 window is displayed with D0 as 0



Port1 window is displayed with D0 as 0



Stop debugging process



After debugging, window appears in this format

Experiment-2

Aim of the Experiment: - Write a C Program to print hello world.

Software required: - Keil u vision 5

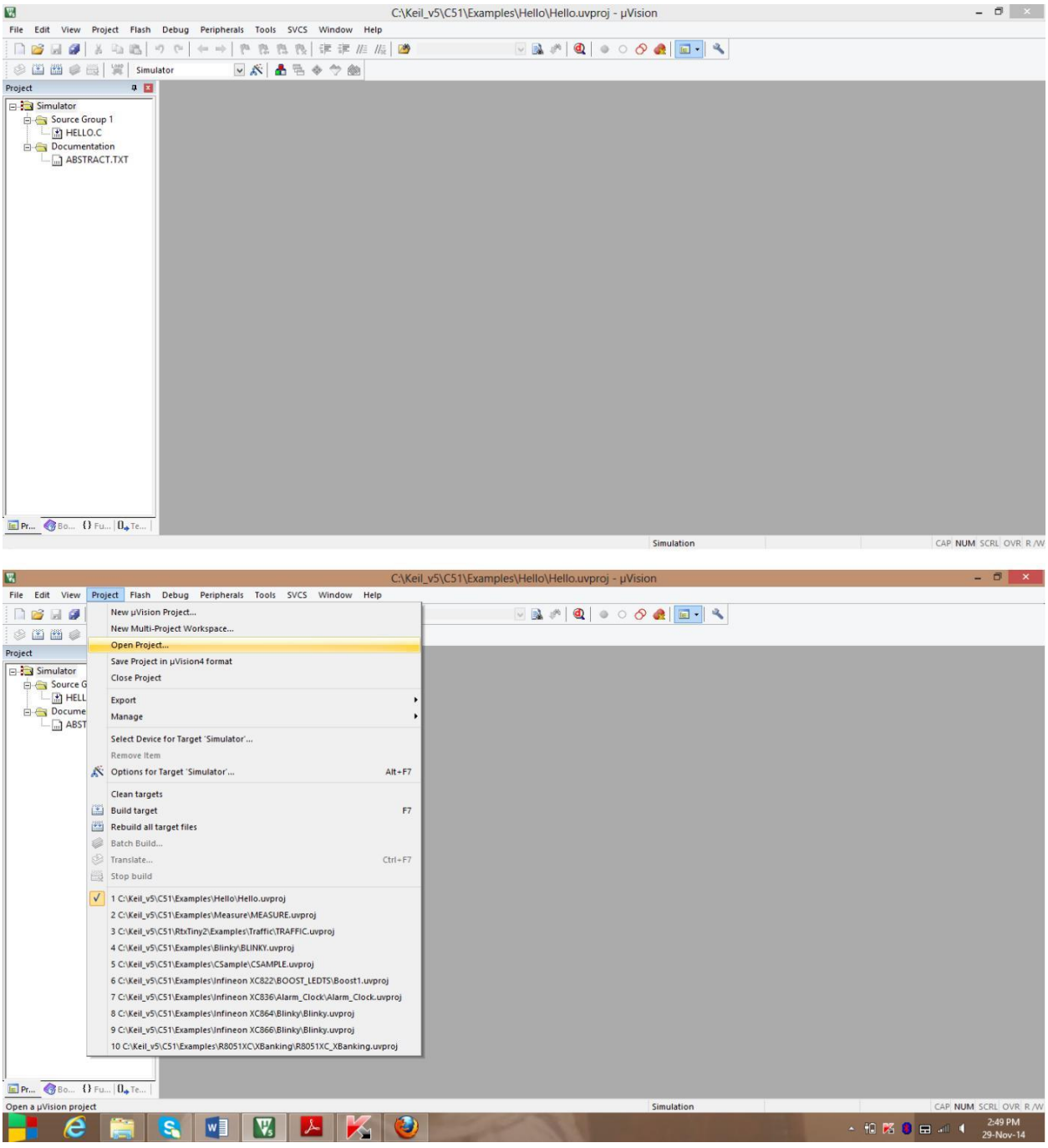
Theory: - This experiment aims to print “hello world” using the software keil u vision 5. The program is by default present after installation of this software. Firstly the header files REG52.H, stdio.h are declared for the intended 8051. Programming for debugging with Monitor-51is made. Now the main function starts. The serial port for 1200 baud at 16MHz is set up. An embedded program never exits (because there is no operating system to return to). It must loop and execute forever. So an infinite loop is made that Toggle P1.0 each time we print "Hello World".

Program:-

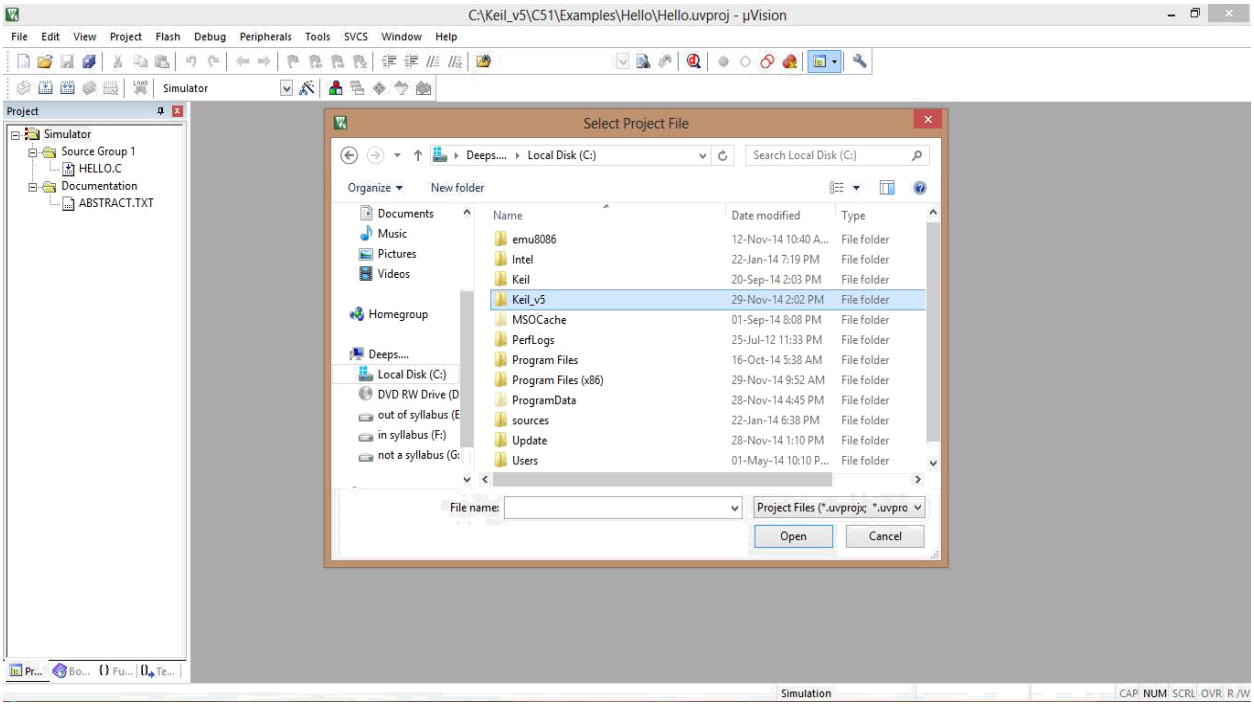
```
#include <REG52.H>
#include <stdio.h>
#ifdef MONITOR51
char code reserve [3] _at_ 0x23;
#endif
void main (void)
{
#ifndef MONITOR51
   SCON = 0x50;
   TMOD |= 0x20;
   TH1   = 221;
   TR1   = 1;
   TI    = 1;
#endif
  While (1) {
   P1 ^= 0x01;
   printf ("Hello World\n");
  }
}
```

Procedure:-

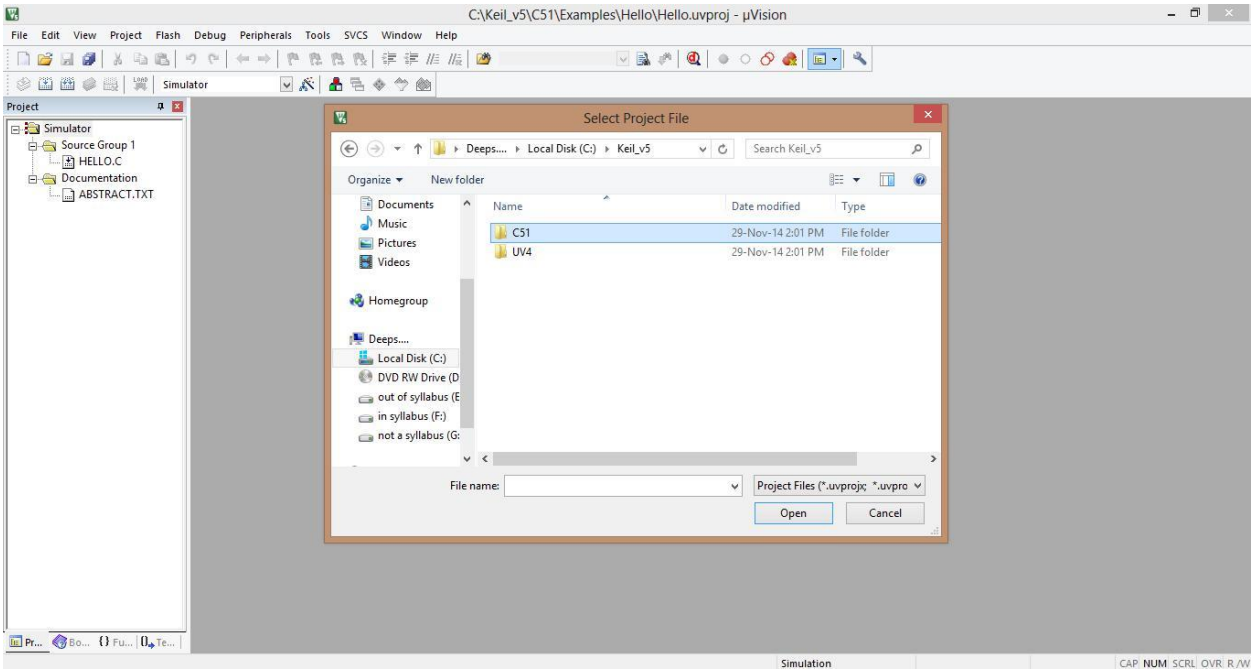
As said the program for hello world is by default present after the installation of the keil u vision software, the program is loaded by following certain steps which is as follows:-



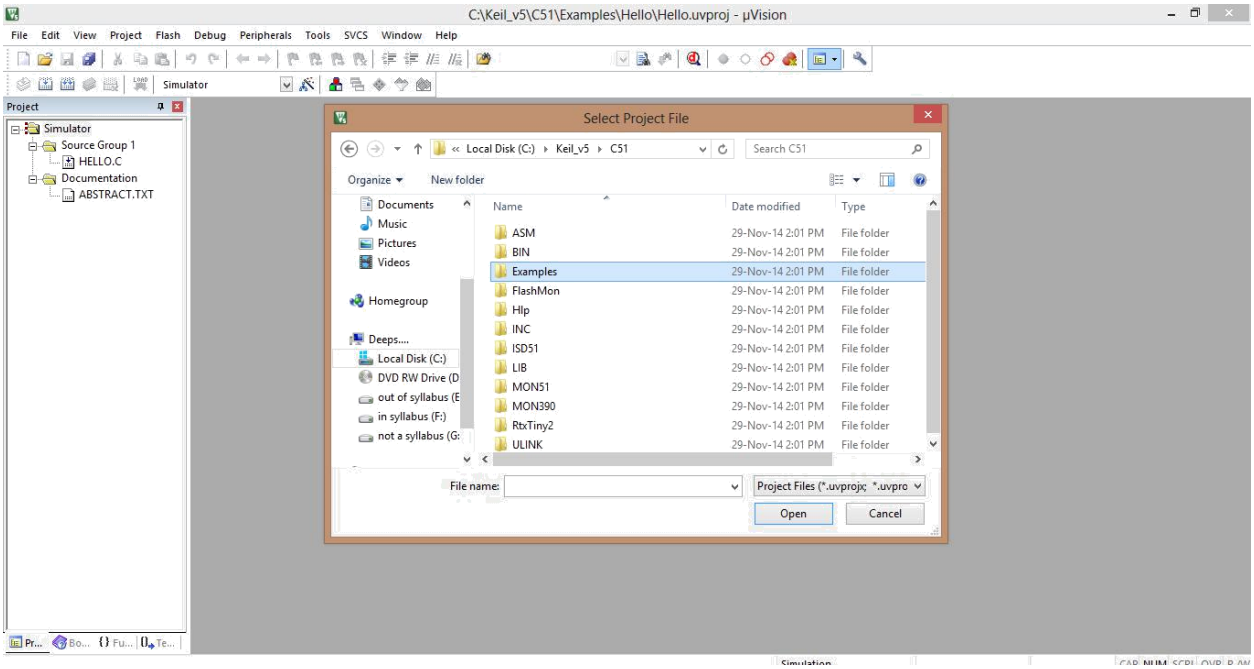
Select open project from project



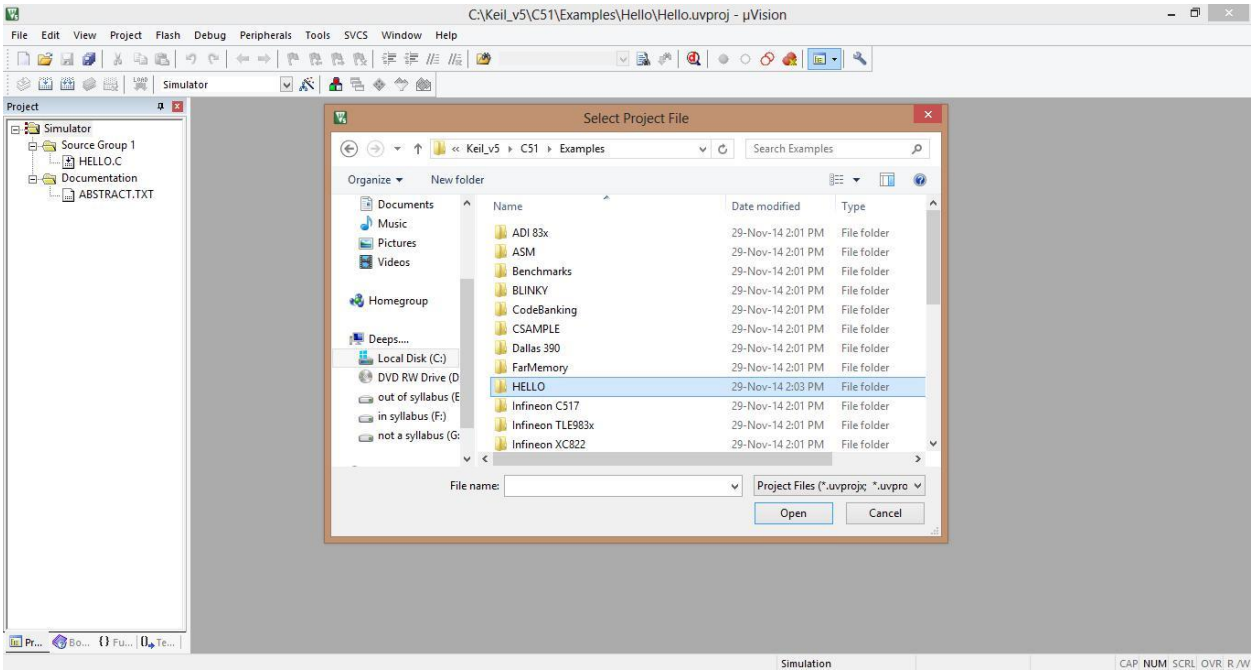
Select Keil_v5 present in Local Disk (C:)



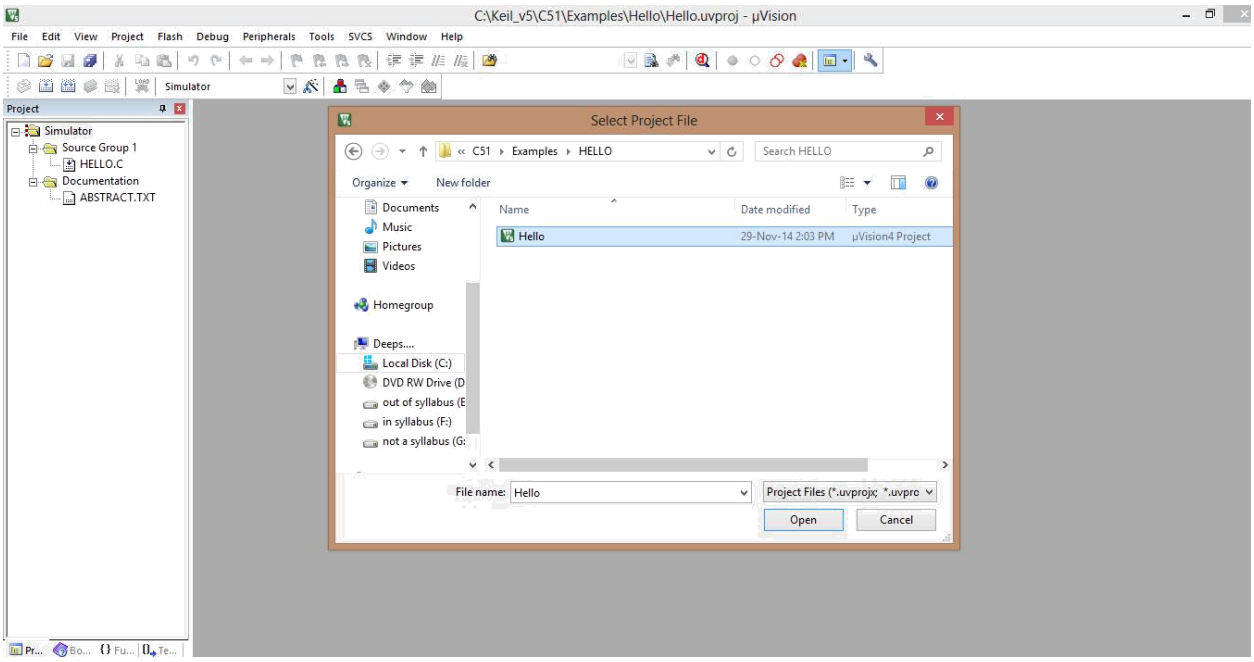
Next select the folder C51



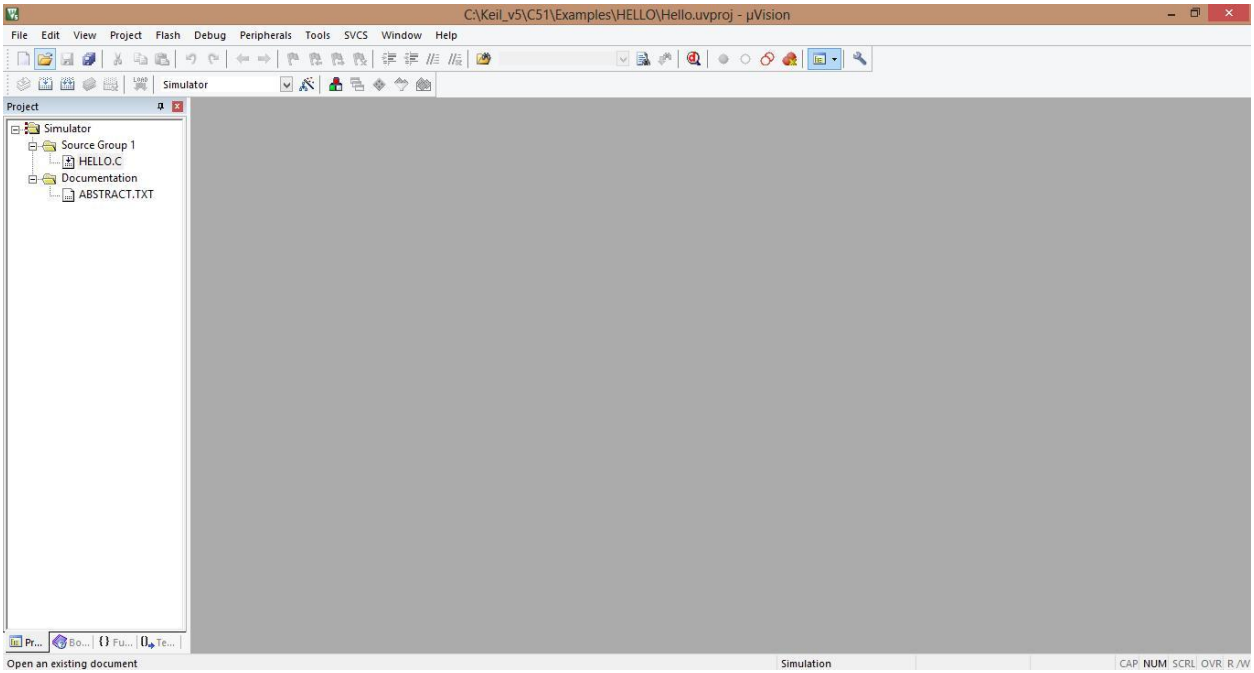
Next select the folder examples



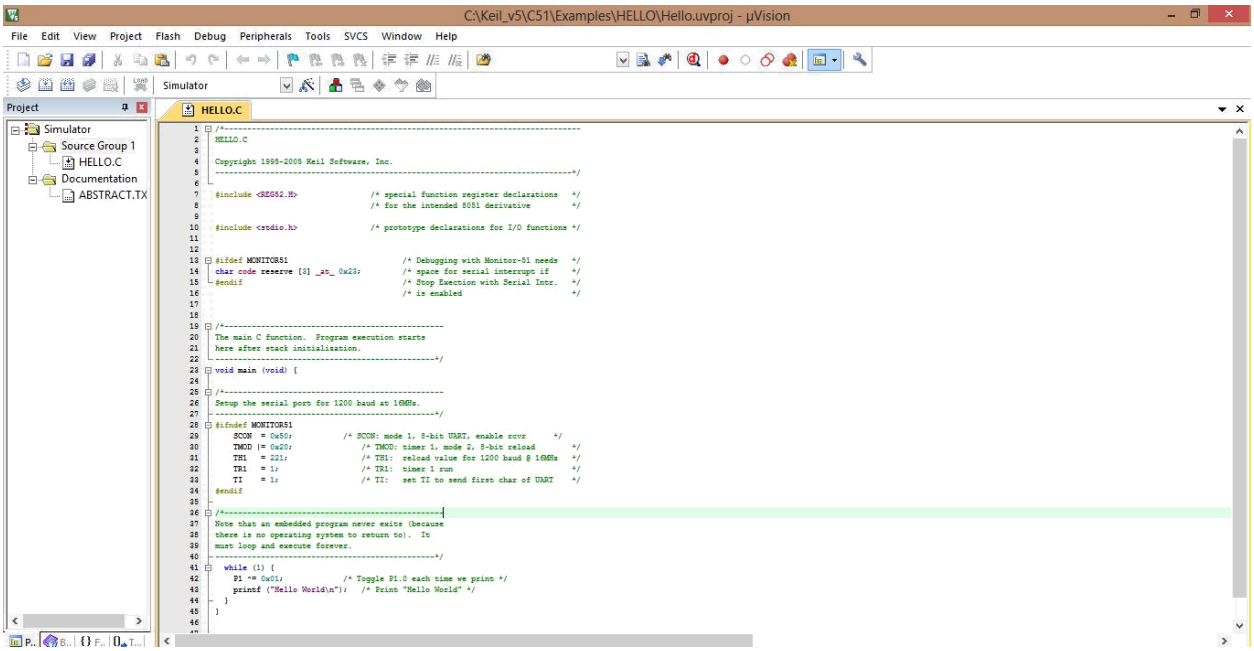
Next select the folder HELLO



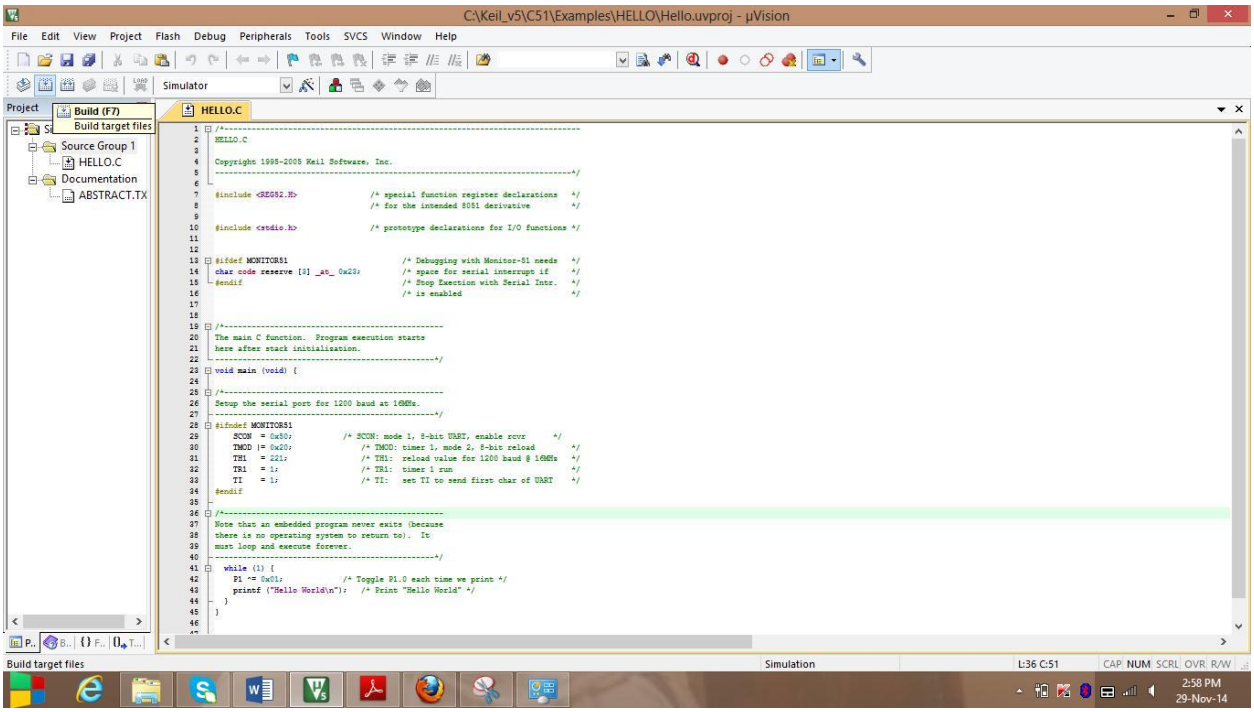
Now select and open uvision4 project file



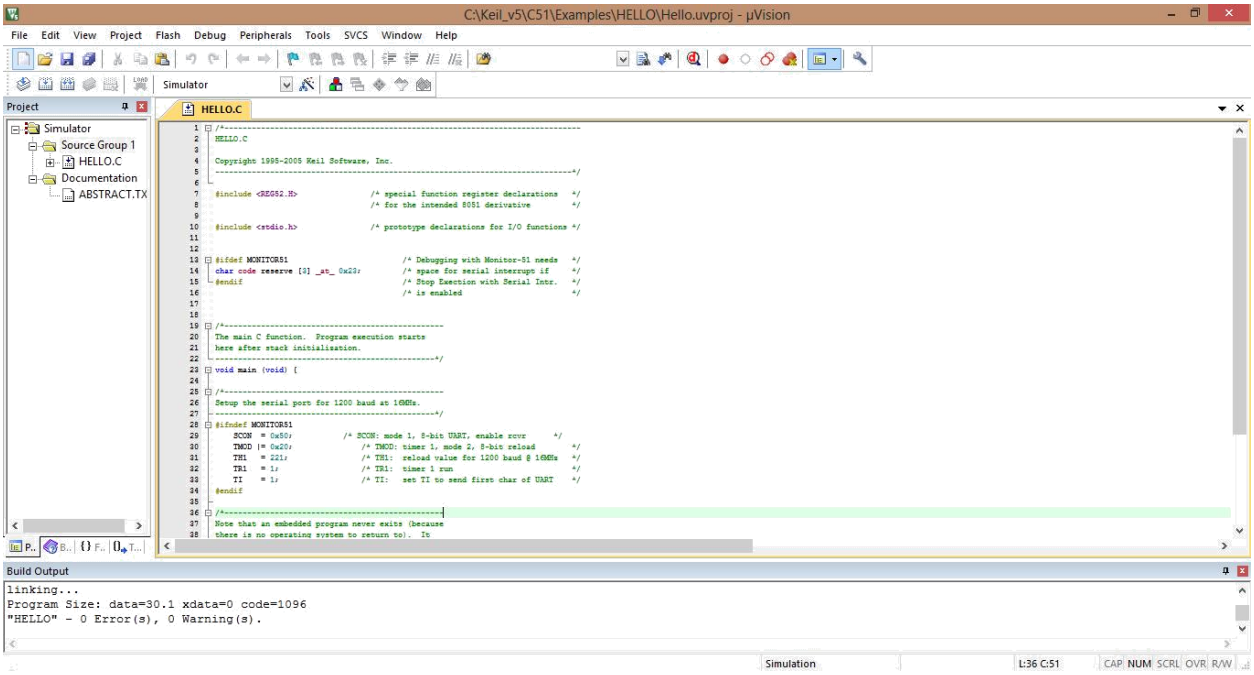
After selecting the window appears in this format (HELLO.C is added to Source Group 1)



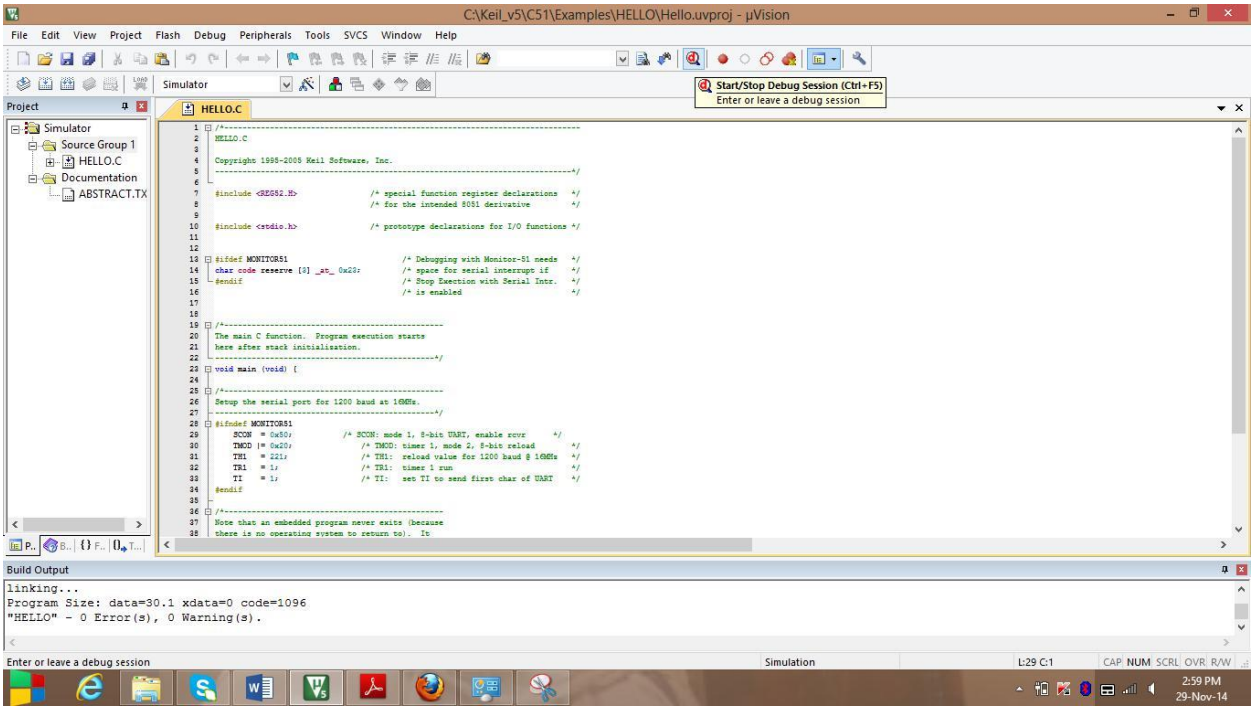
Program of HELLO.C



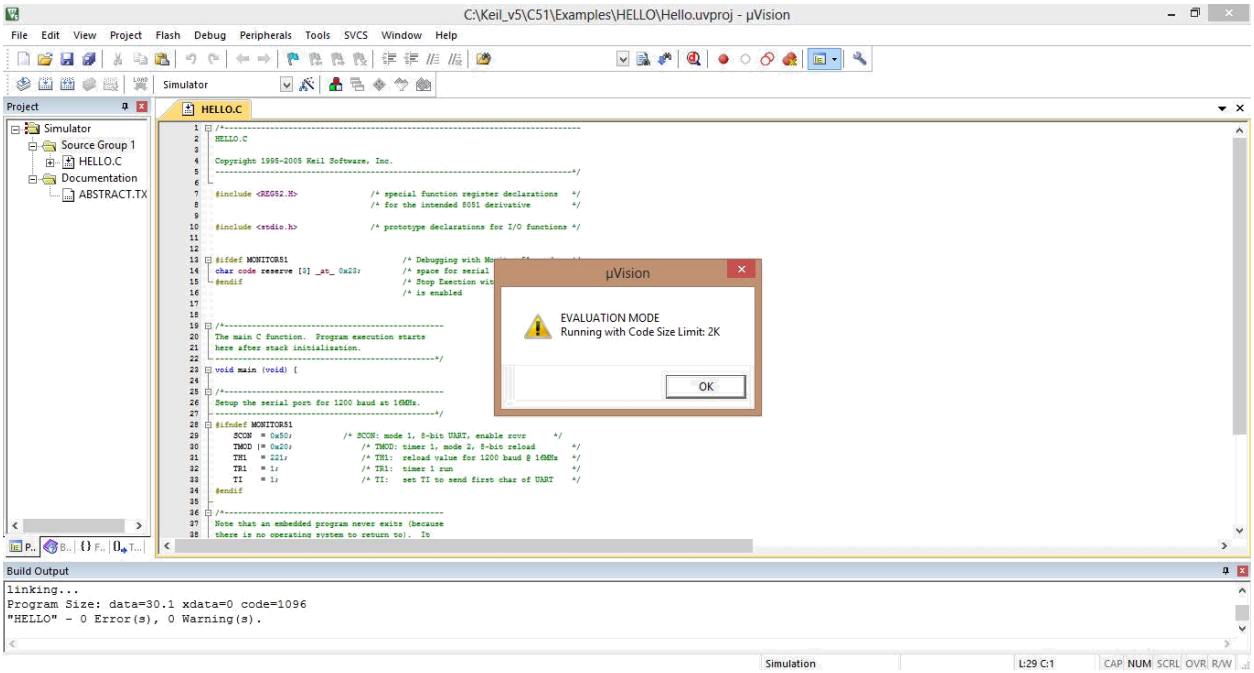
Build the target



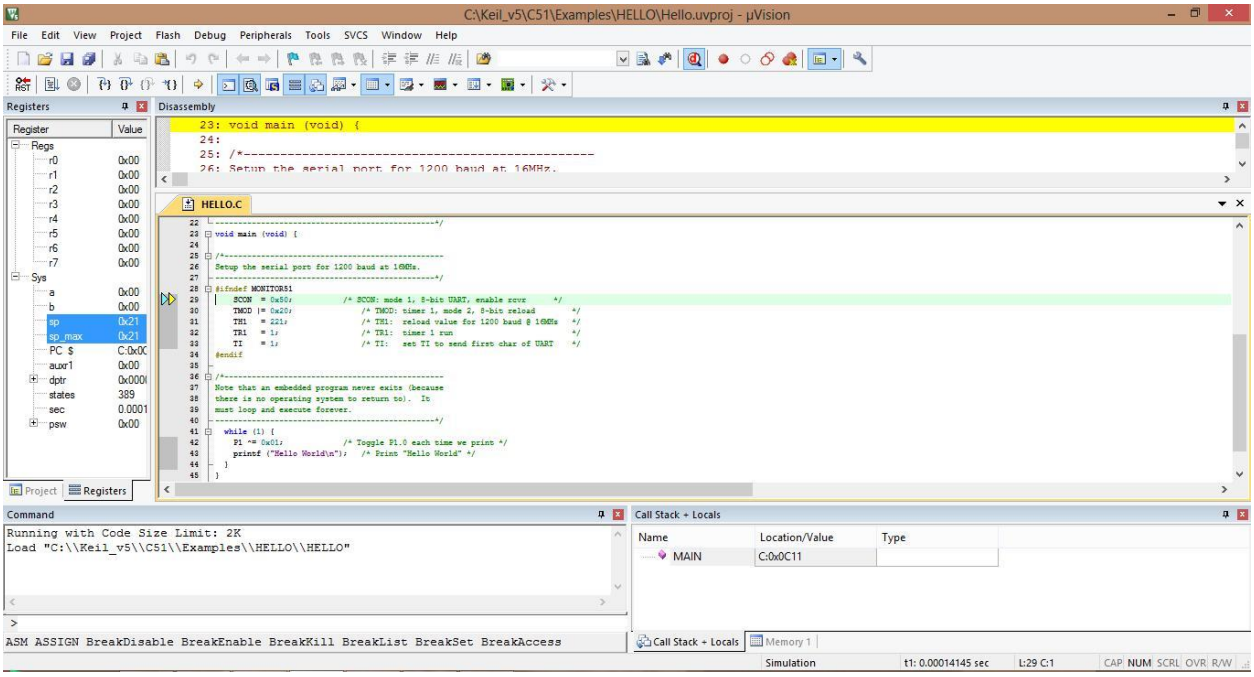
Displaying 0 errors and 0 warnings



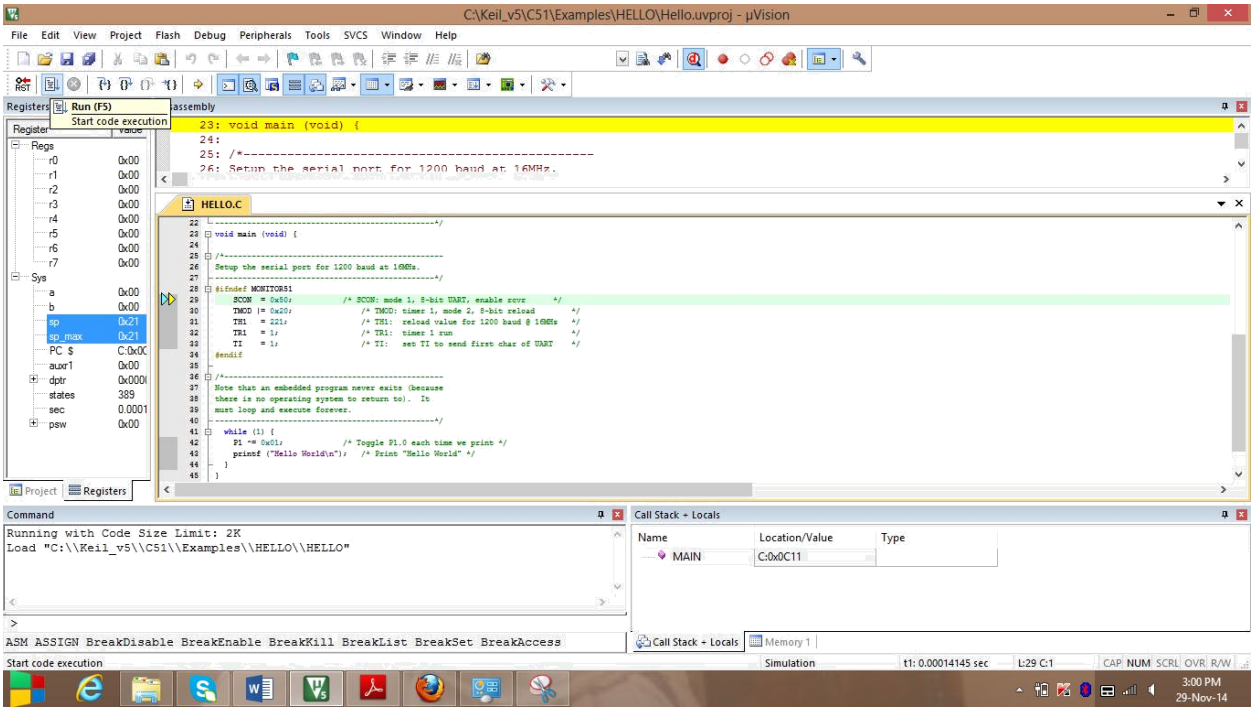
Debugging the target



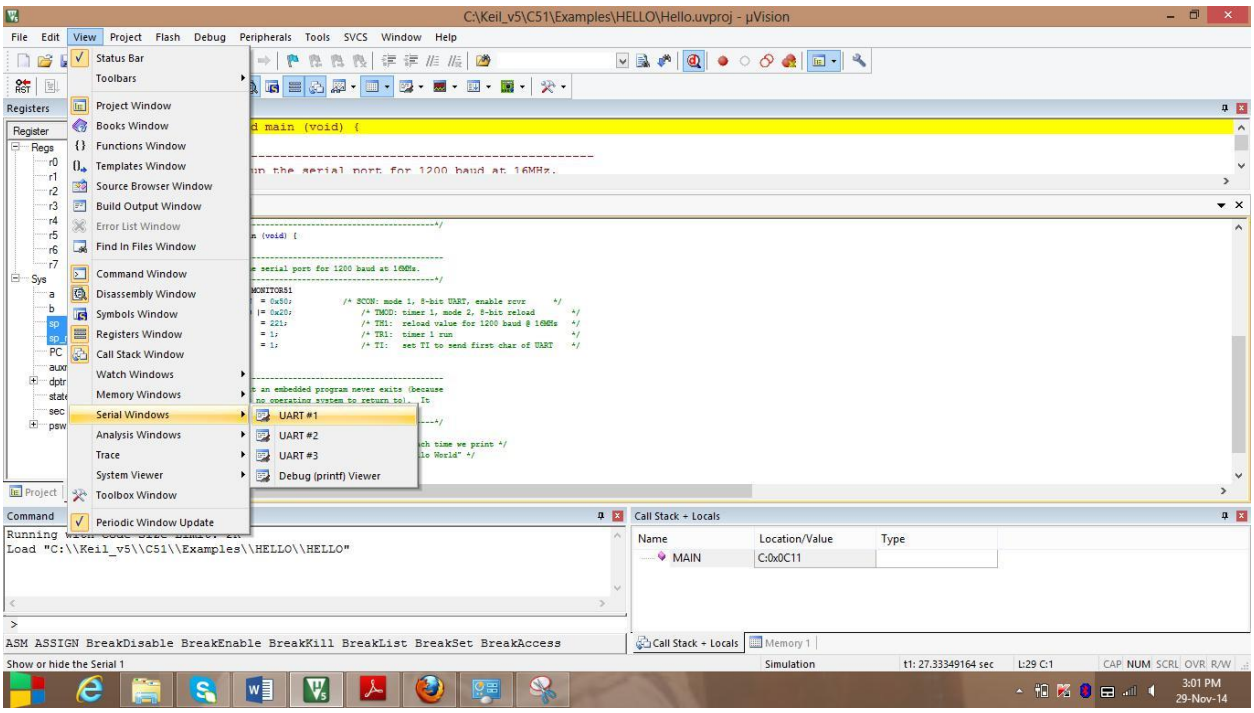
New window evaluation mode appeared. Press ok



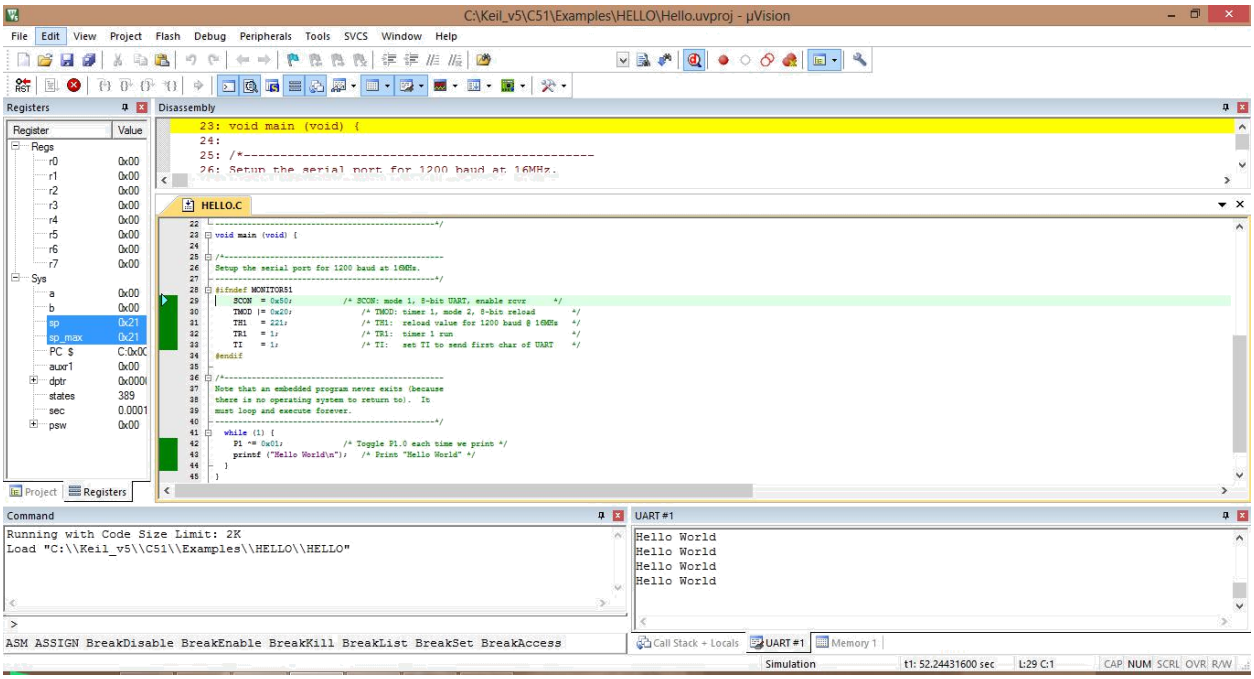
Now the window appears in this format



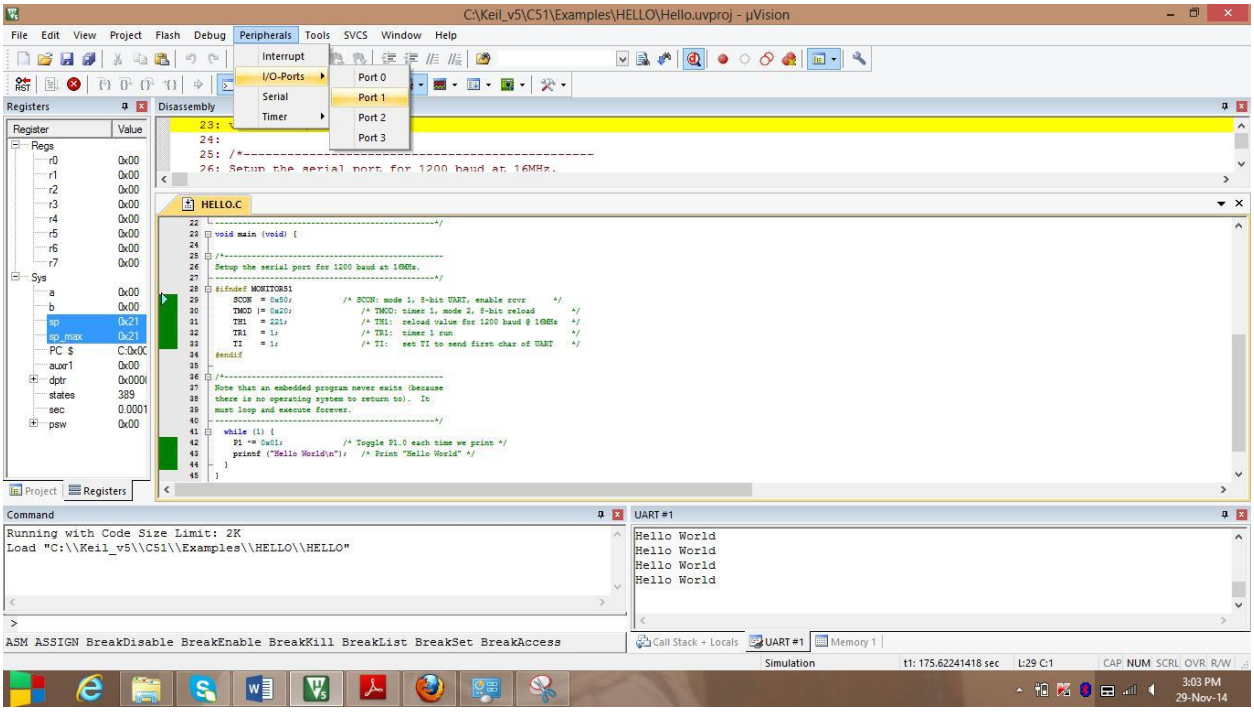
Run the program



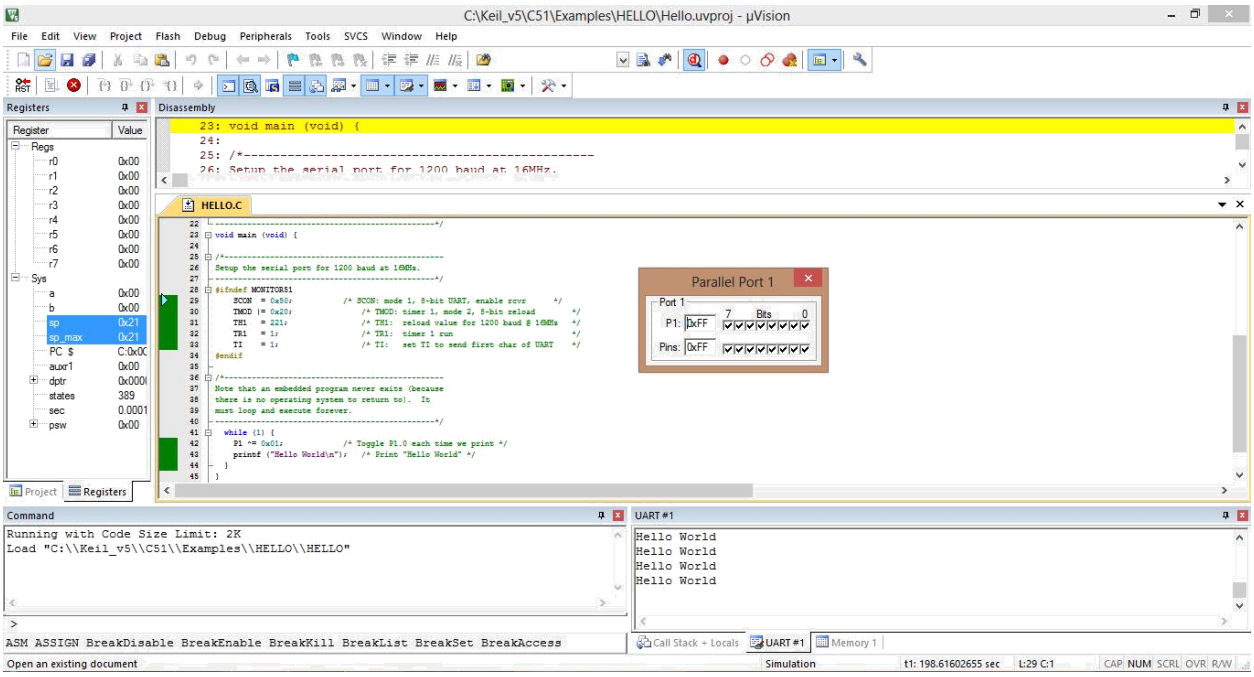
Selecting for UART#1 from serial windows



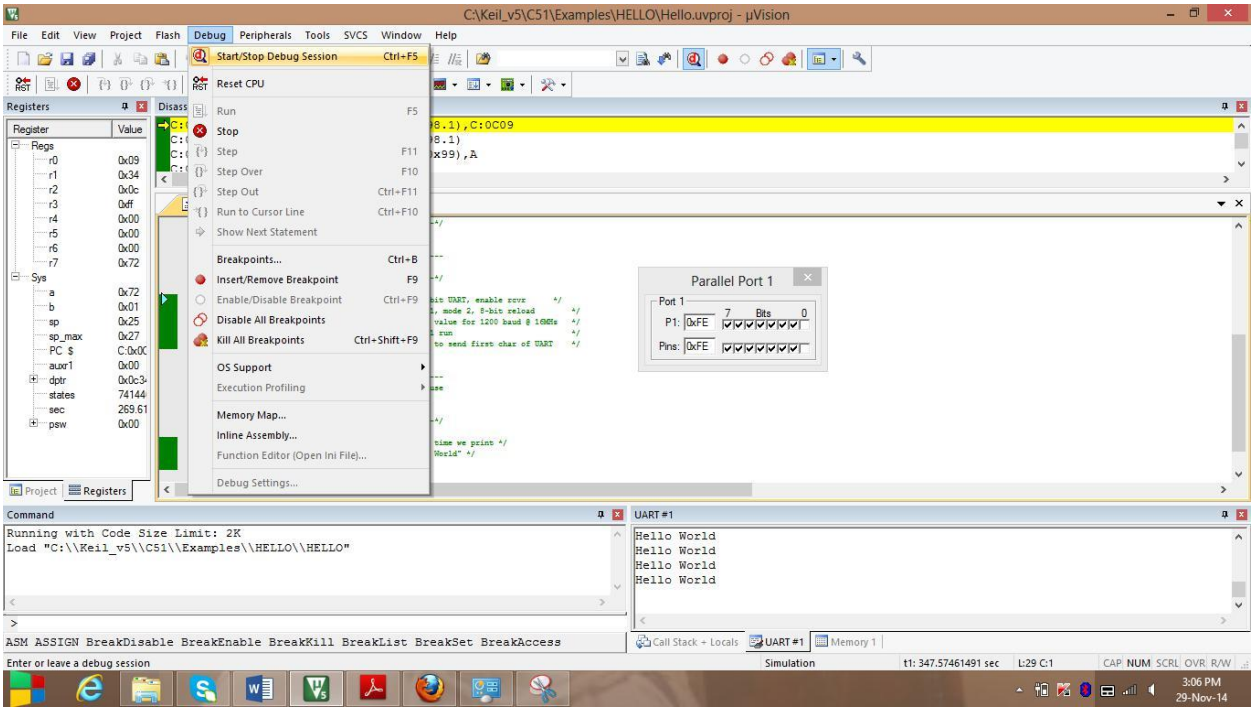
Check the output at UART#1 window



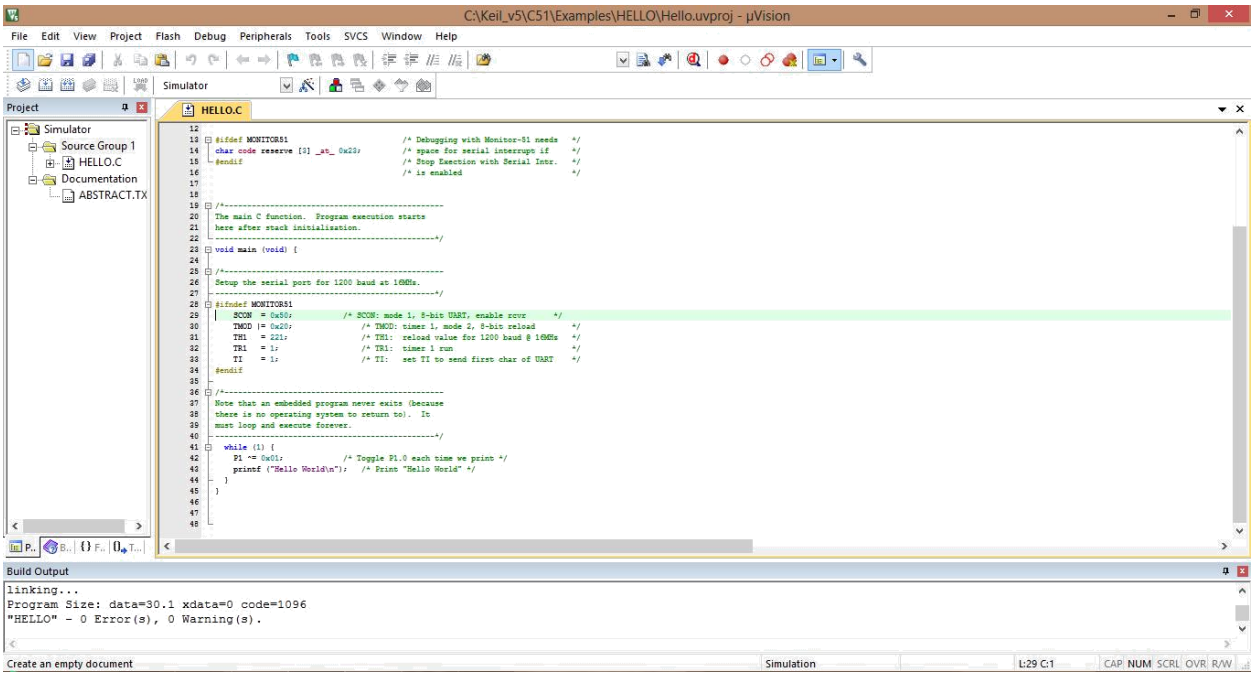
Select Port1 from i/o ports in peripherals



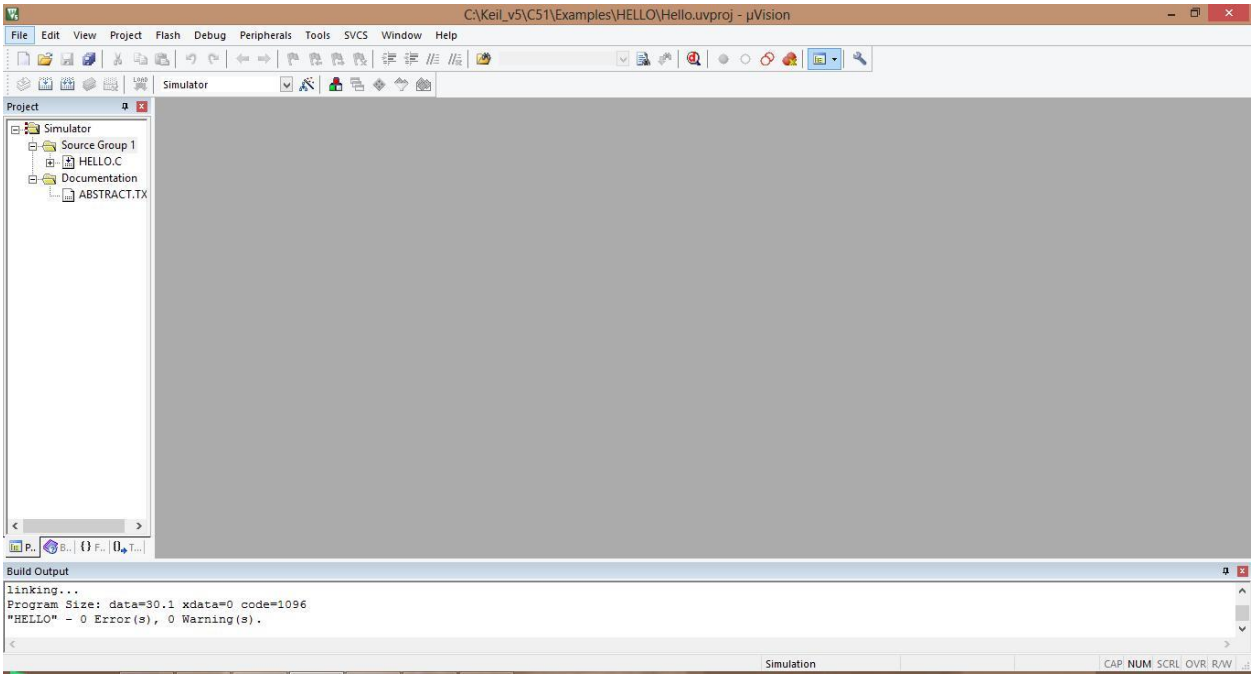
Port1 window is displayed with D0 as 1



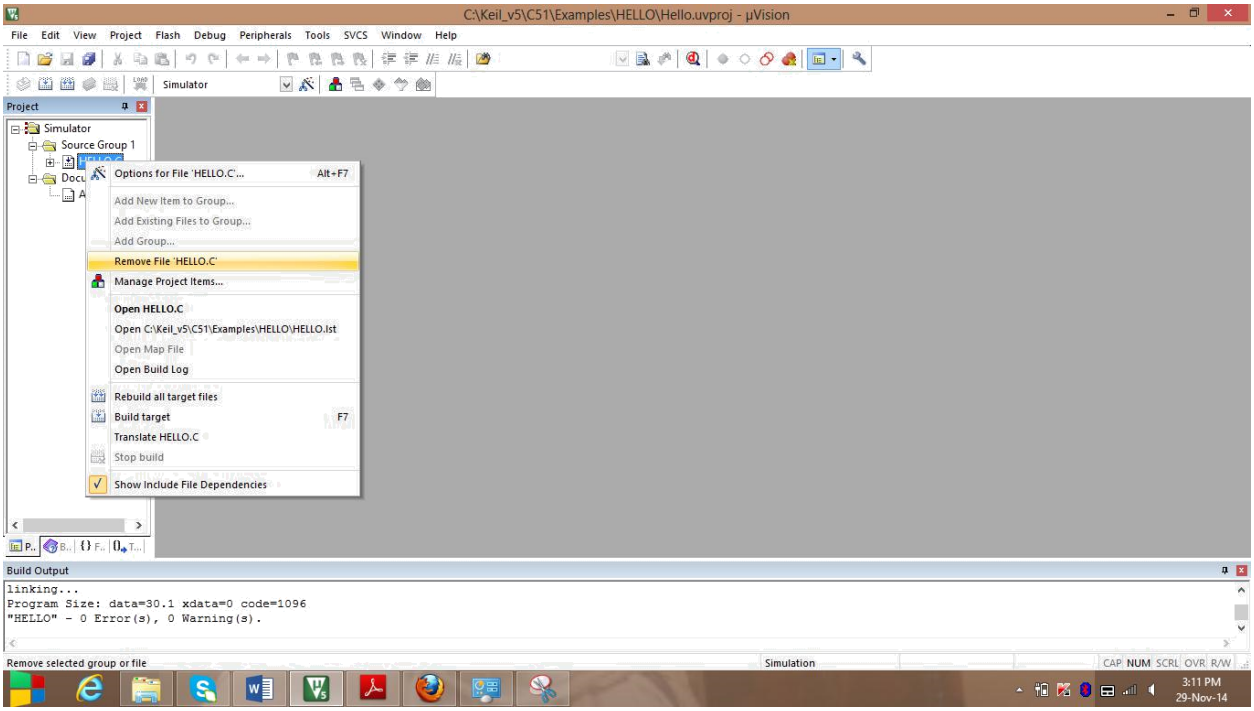
Port1 window is displayed with D0 as 0



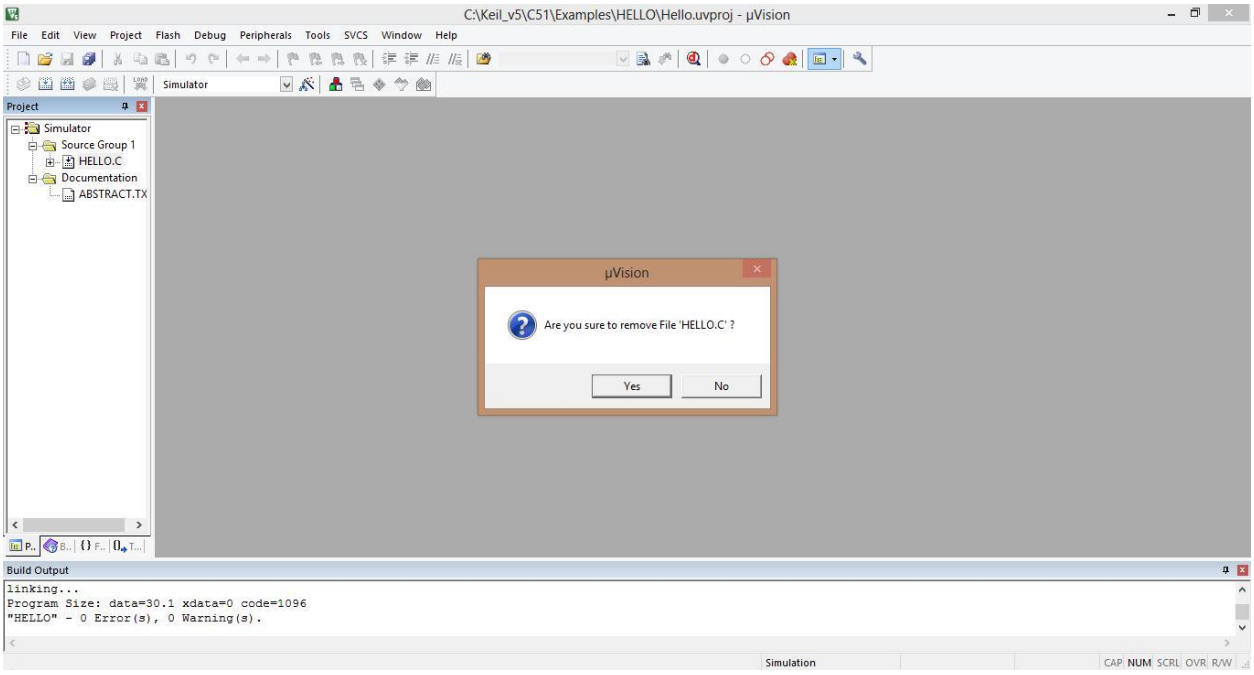
Then stop debugging the process



After closing the program, the window appears in this format

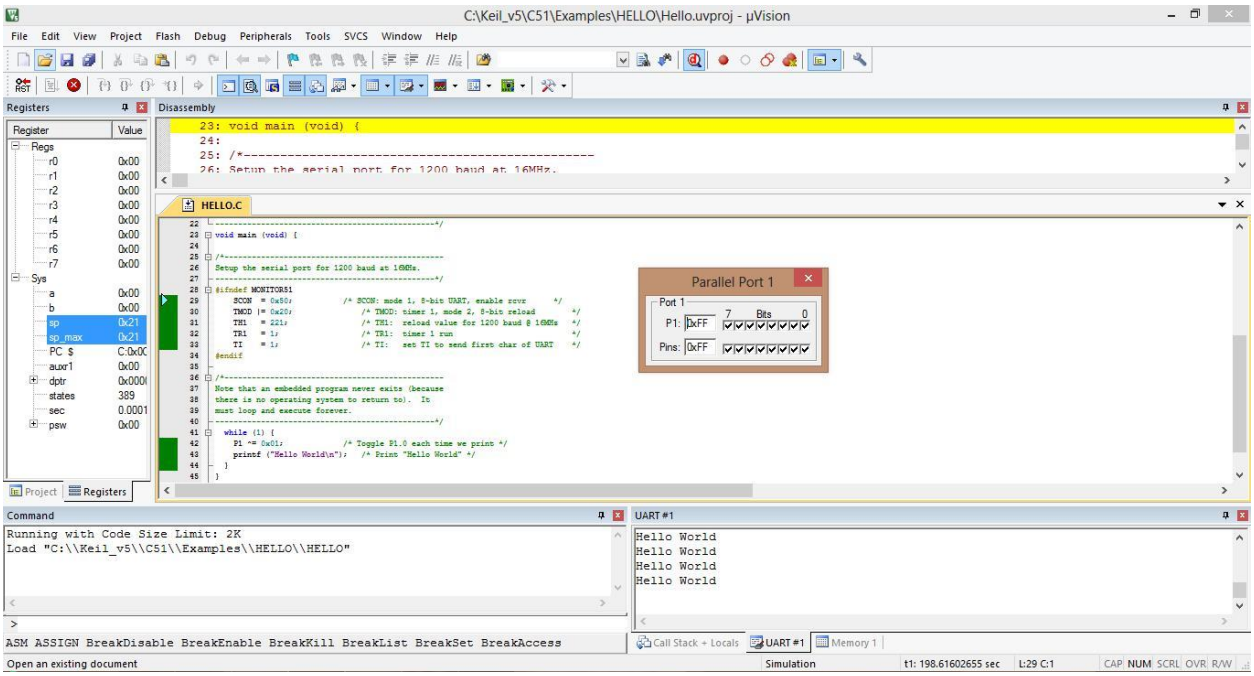


Now remove HELLO.C from source group 1

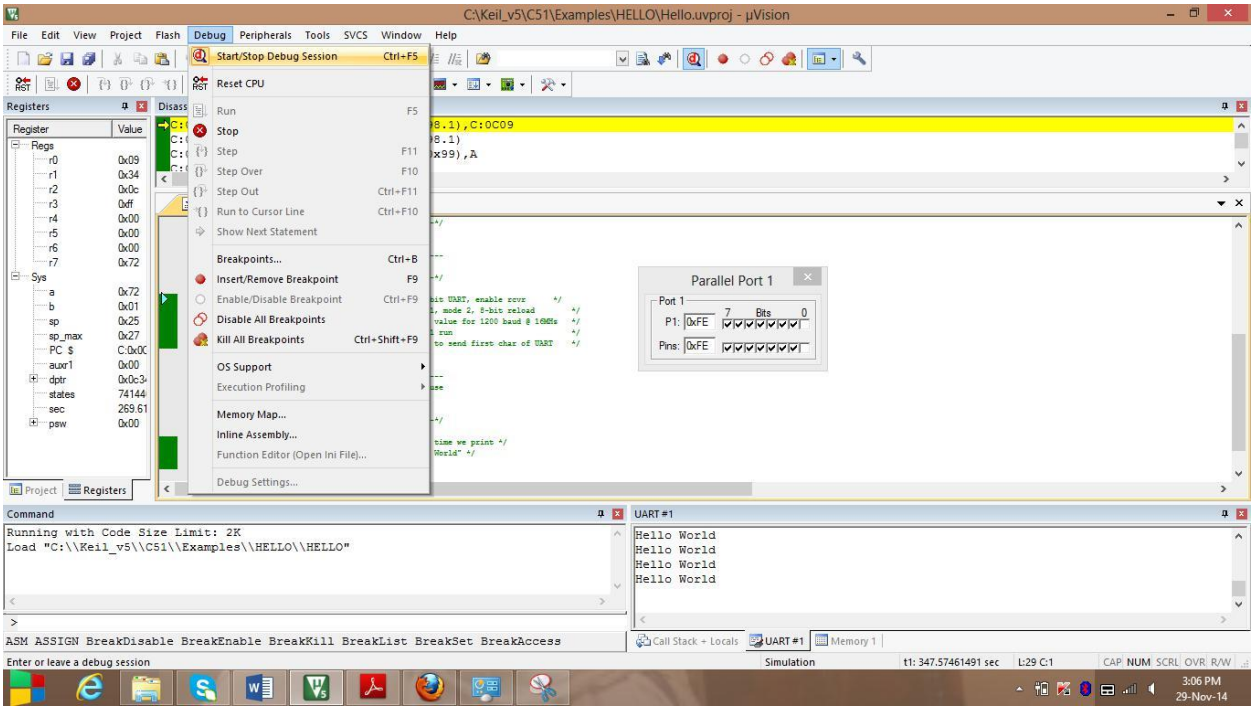


Now select yes

Output:-



HELLO WORLD with D0=1



HELLO WORLD with D0=0

Observation:-

From the above outputs it is analyzed that the message “hello world” is printed at UART#1 at each time P1.0 is toggled.

Conclusion:-

The experiment to print hello world using keil u vision successfully.

Experiment-3

Aim of the experiment:- Write a C program to store the data in the accumulator.

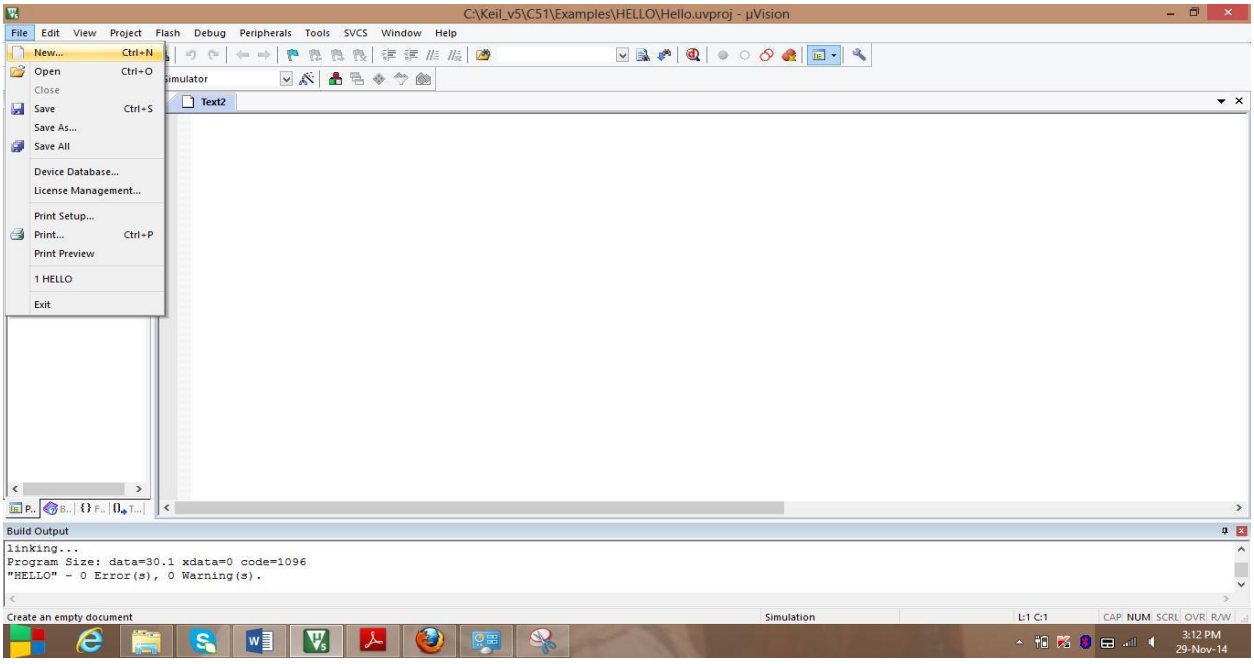
Software required: - Keil u vision 5

Theory: - This experiment aims to store any data in the Accumulator using the software keil u vision 5. In this program, the data 0x05 is to be stored in accumulator i.e. value 05 in hexadecimal. Firstly the header files REG51.H is declared for the intended 8051. Then the main function starts. In the main function 0x05 is stored in Acc variable.

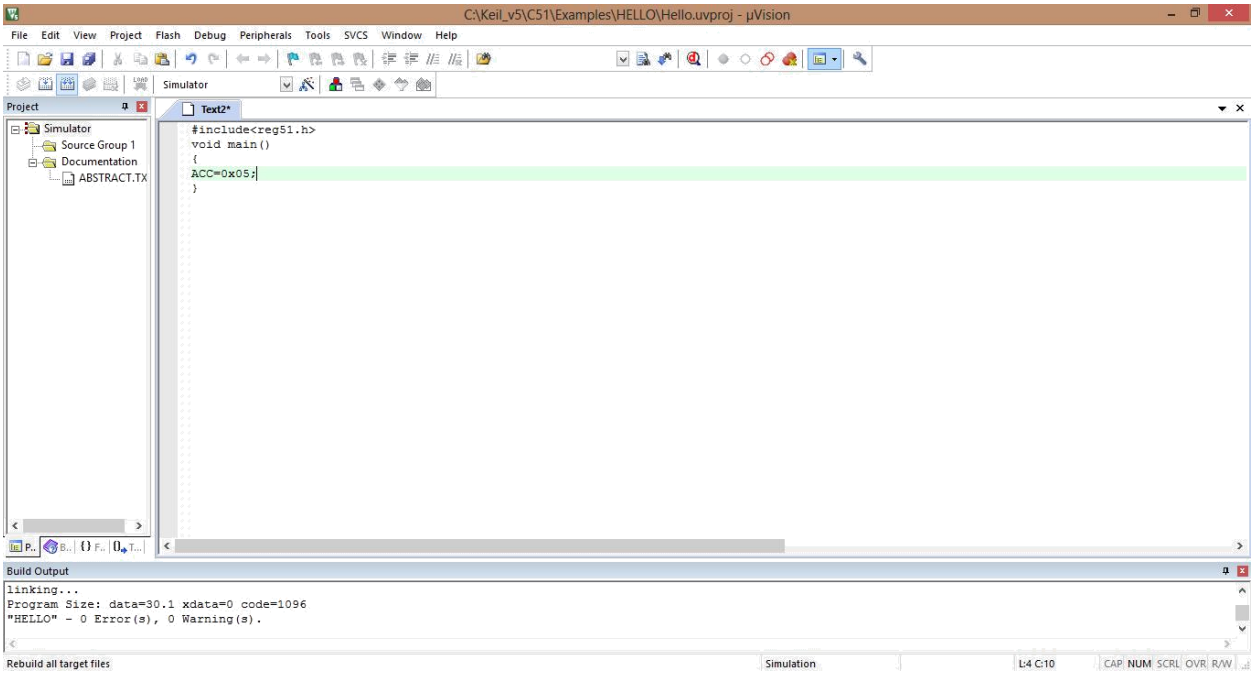
Program:-

```
#include<reg51.h>
void main()

{
Acc=0x05;
}
```

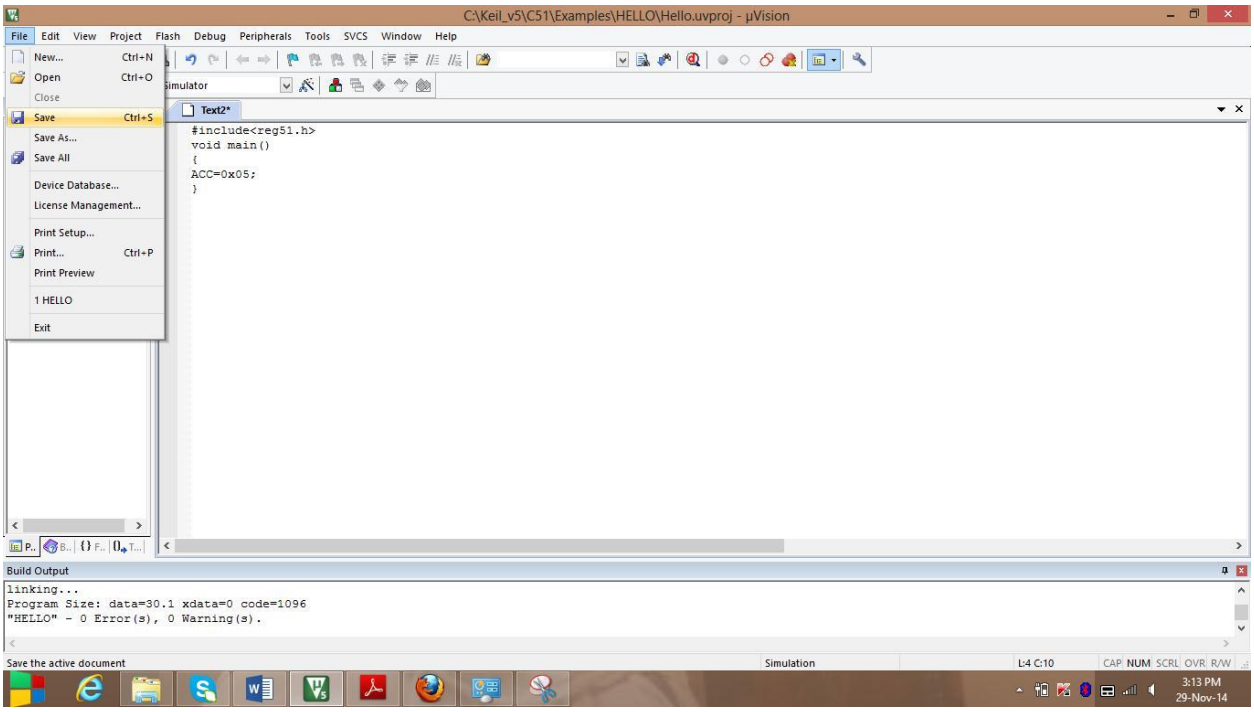
Procedure :- The procedure for the program of storing a data in accumulator is as follows.



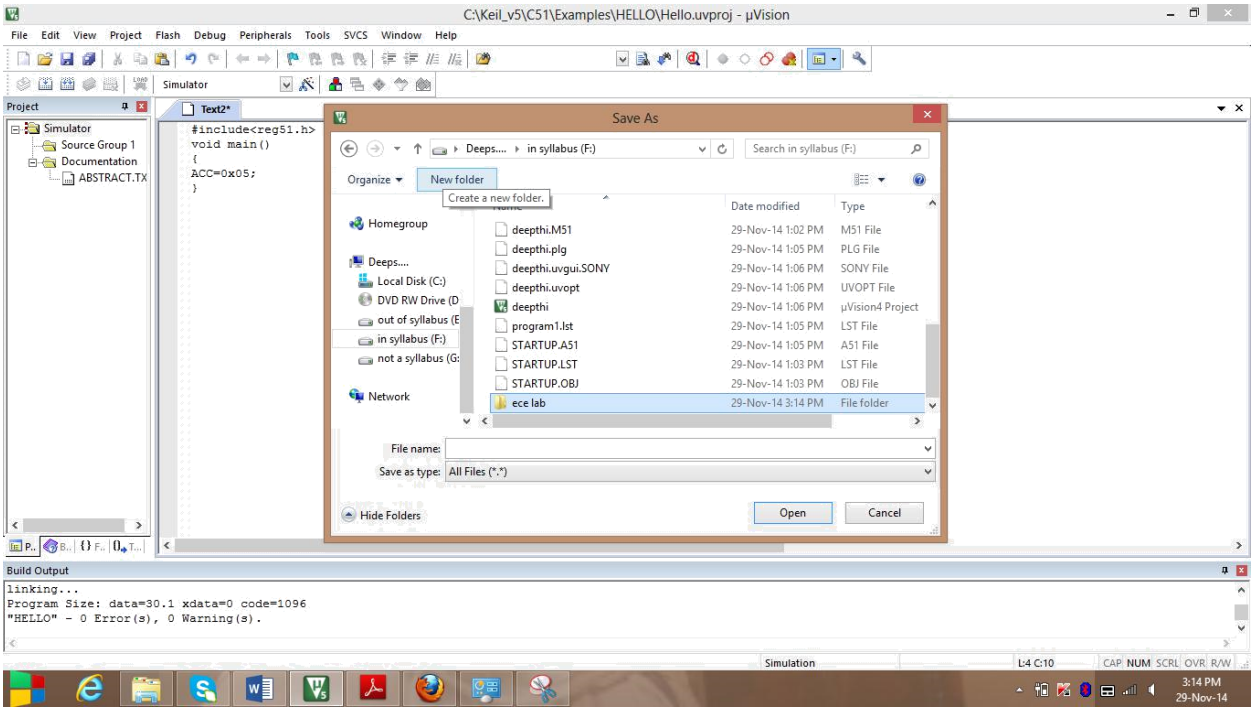
A new file is is created



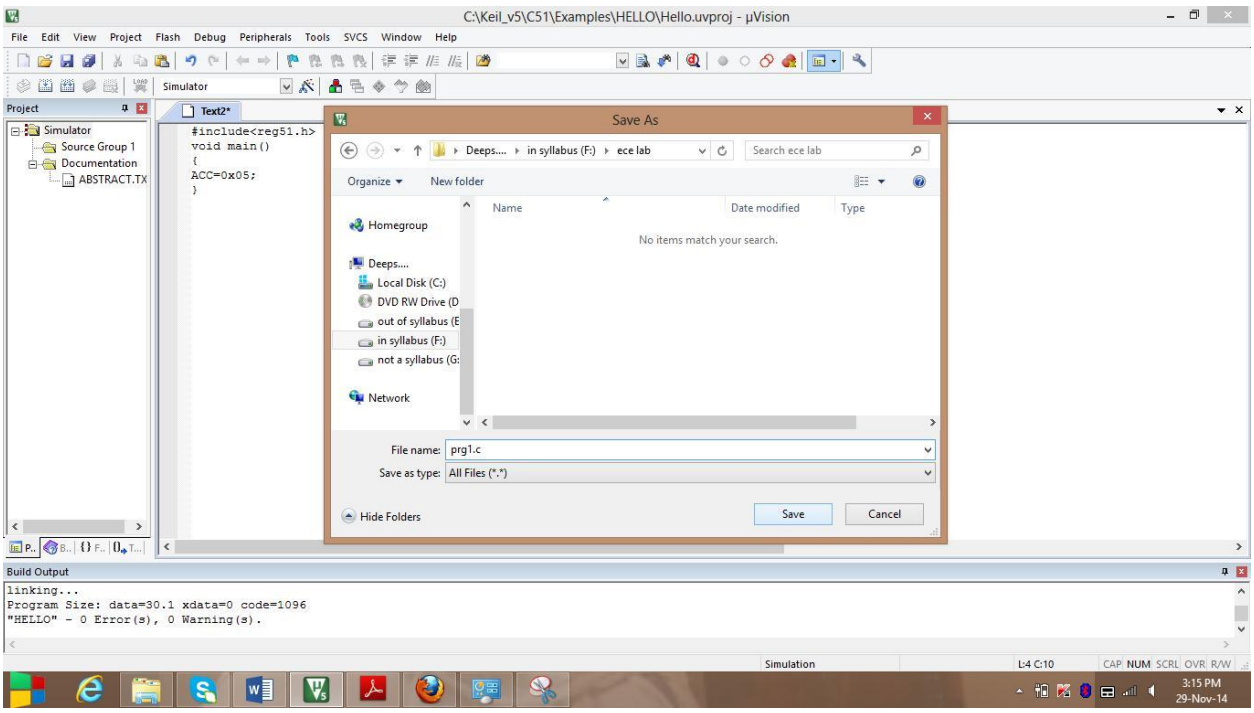
The program is written



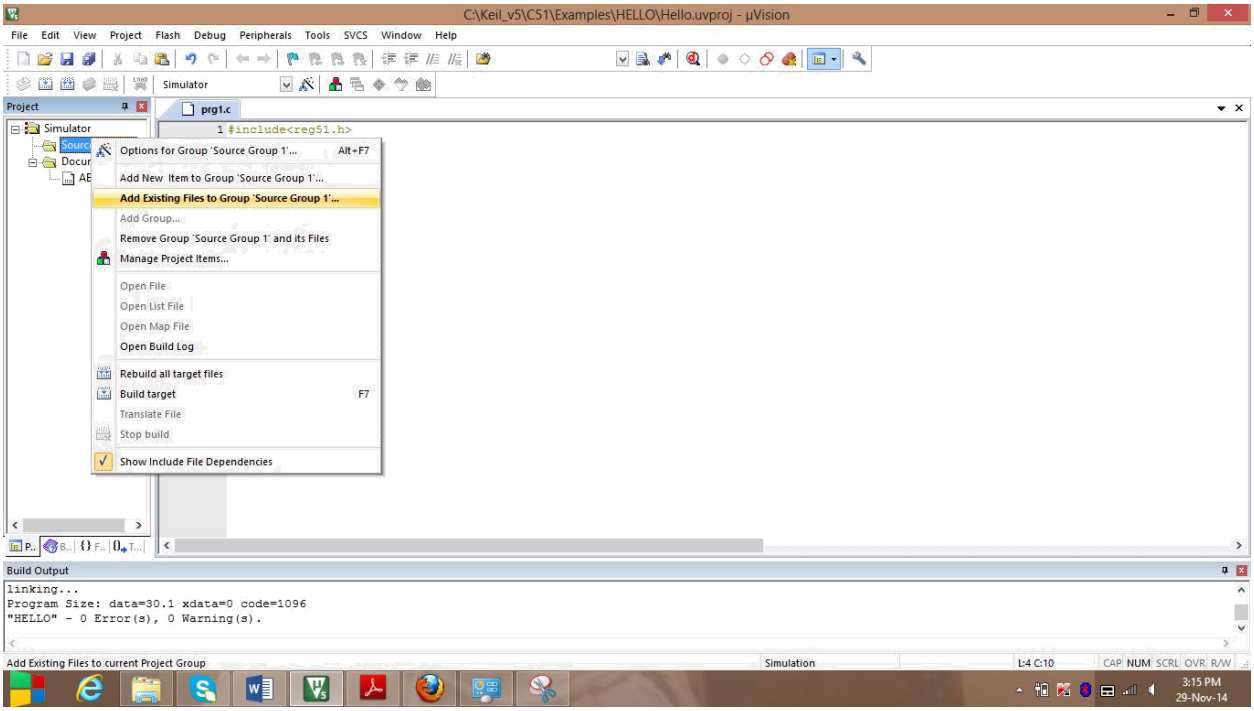
The program is to be saved



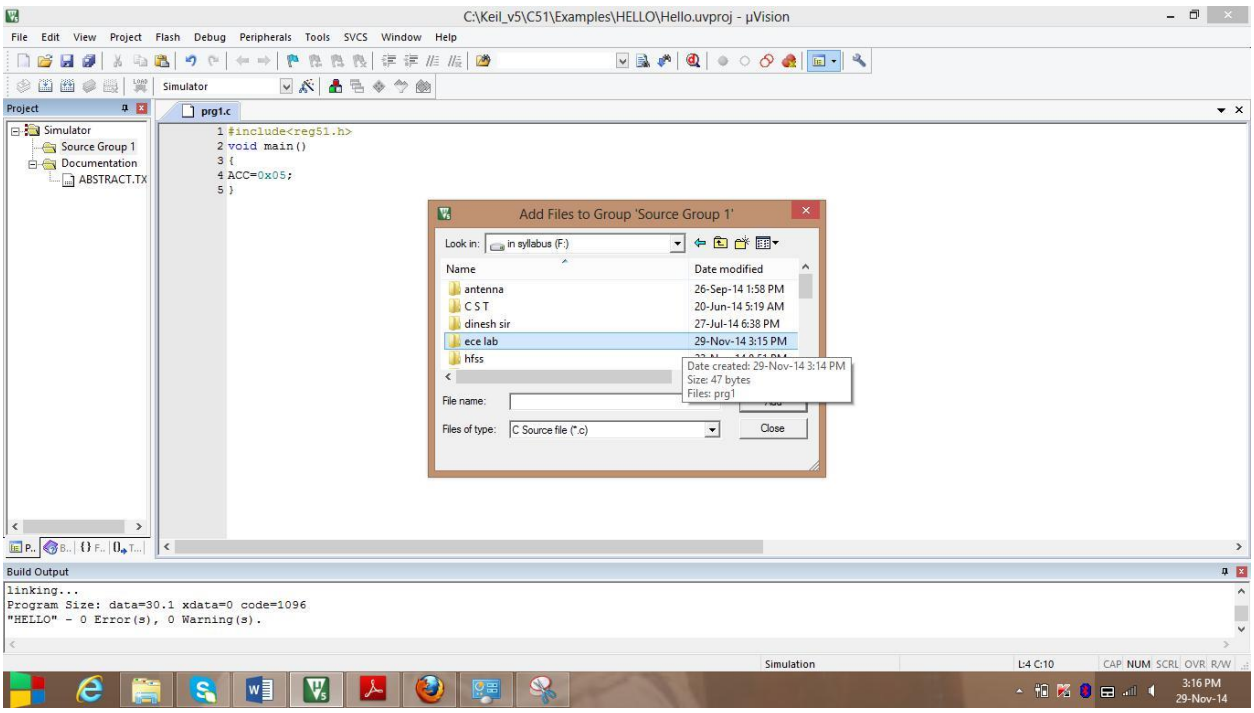
The program is saved in a new folder (created folder ece lab)



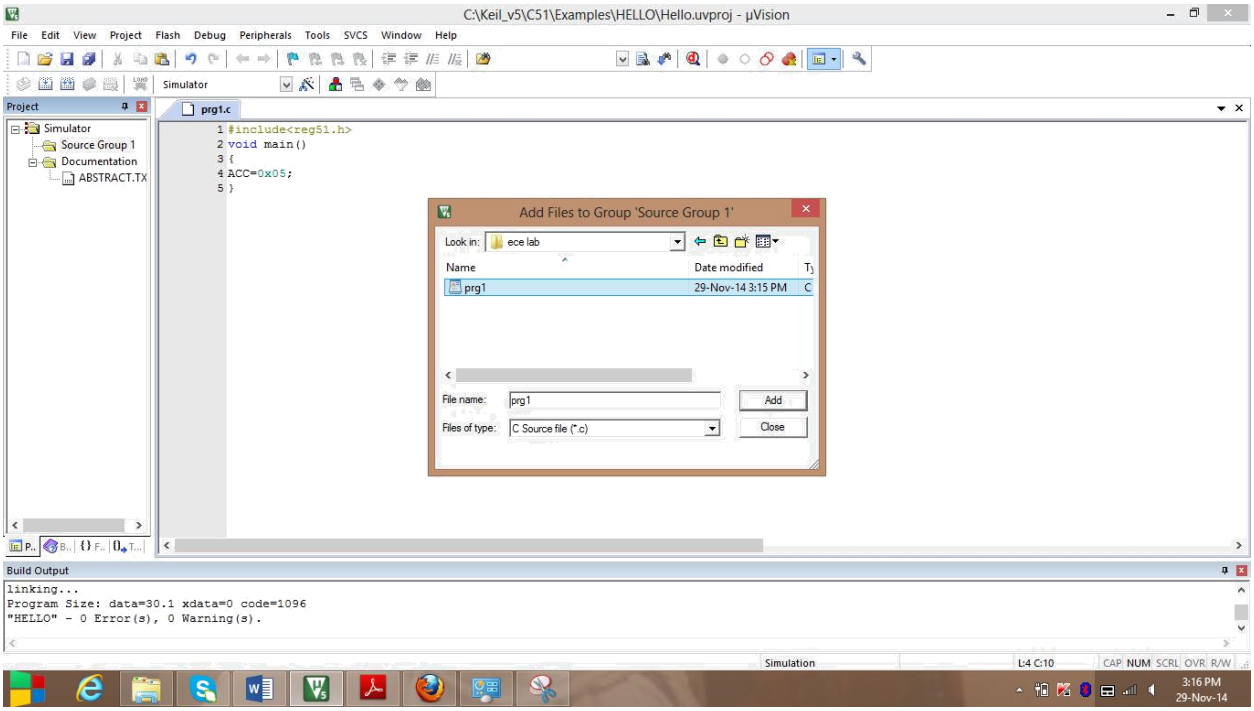
The program is to be saved with the extension .c



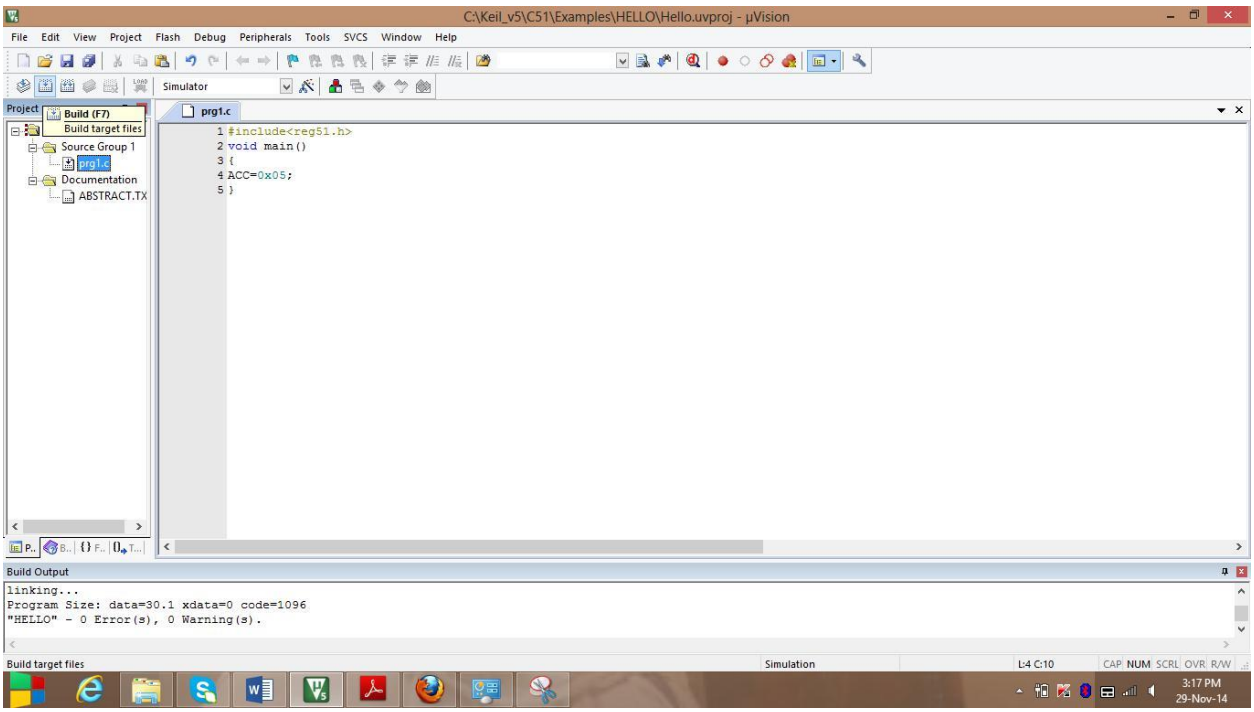
Addition of prg1.c to the Source Group1



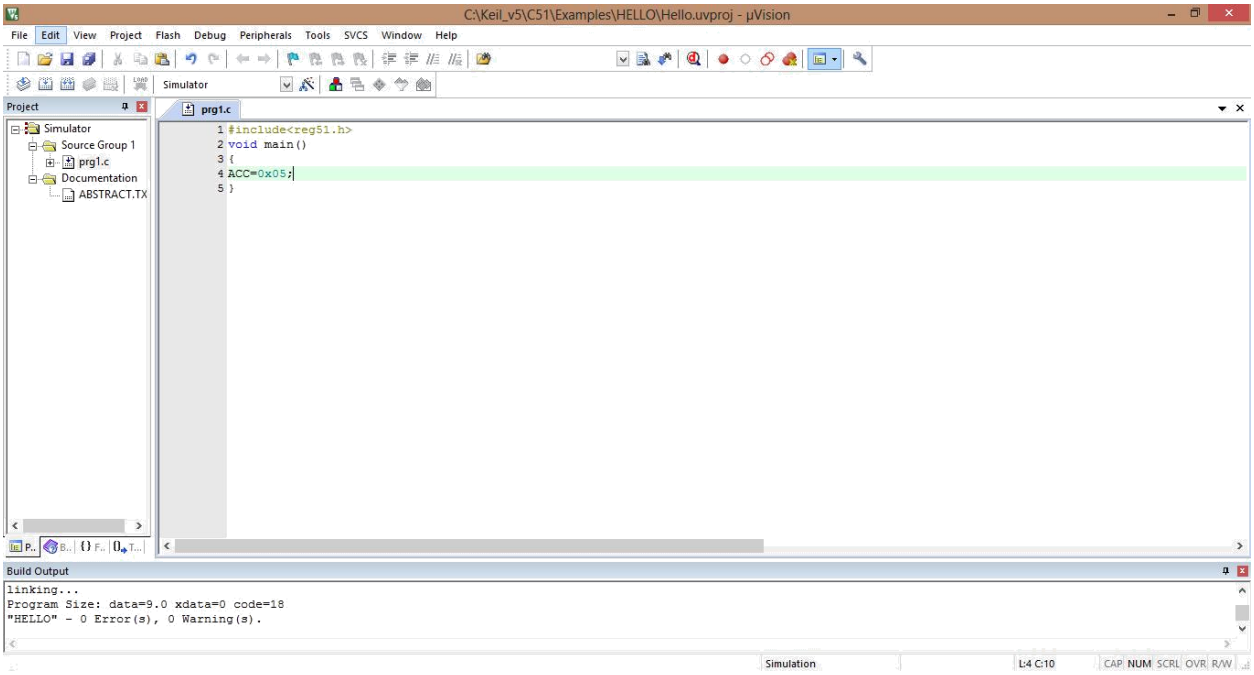
The newly created folder ece lab is selected



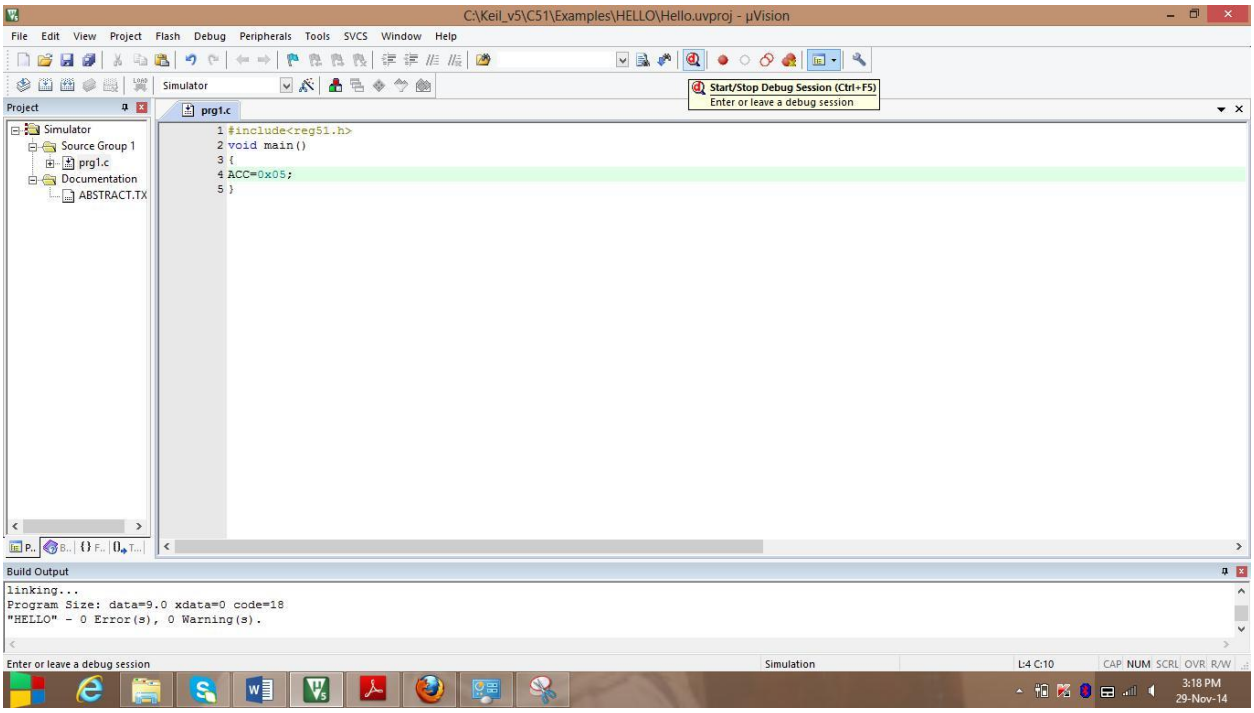
Now prg1.c is selected



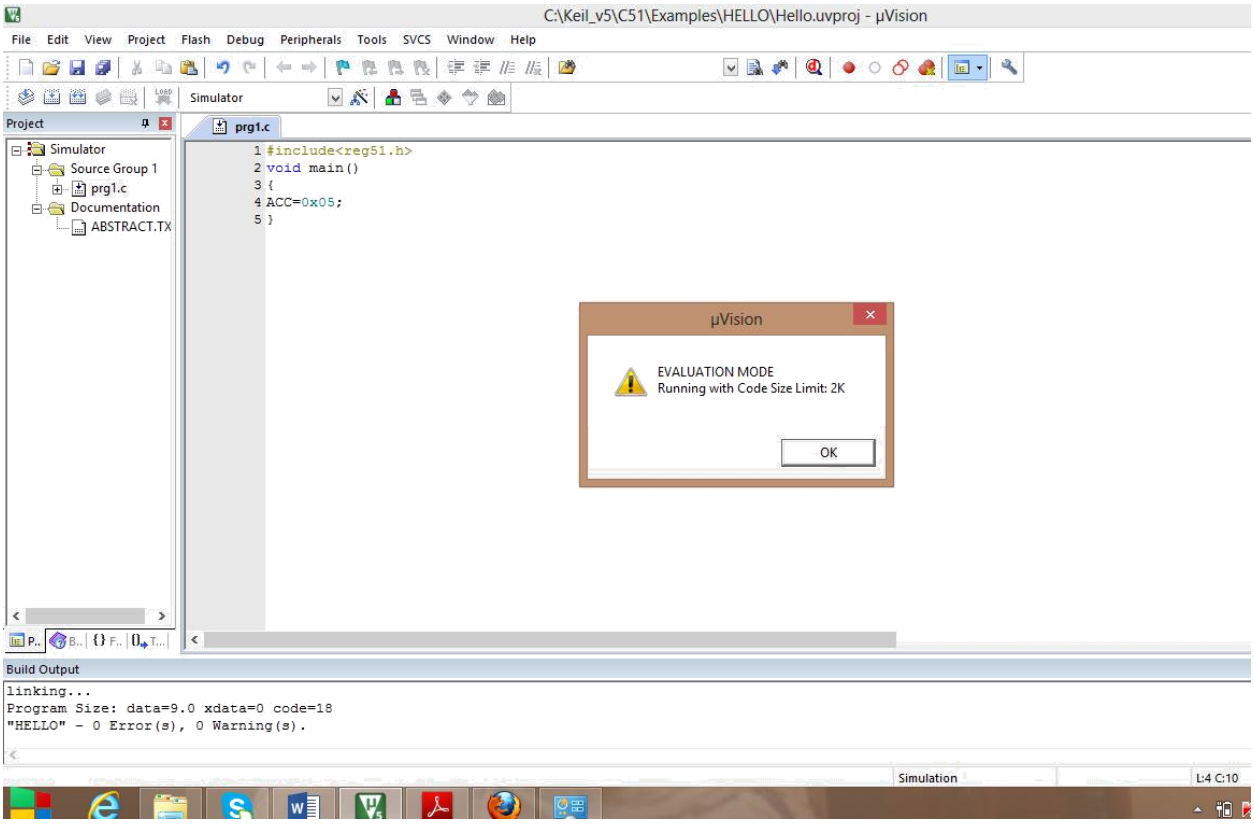
Check whether the prg1.c is added in Source Group1 or not. Now bulid(F7) the target.



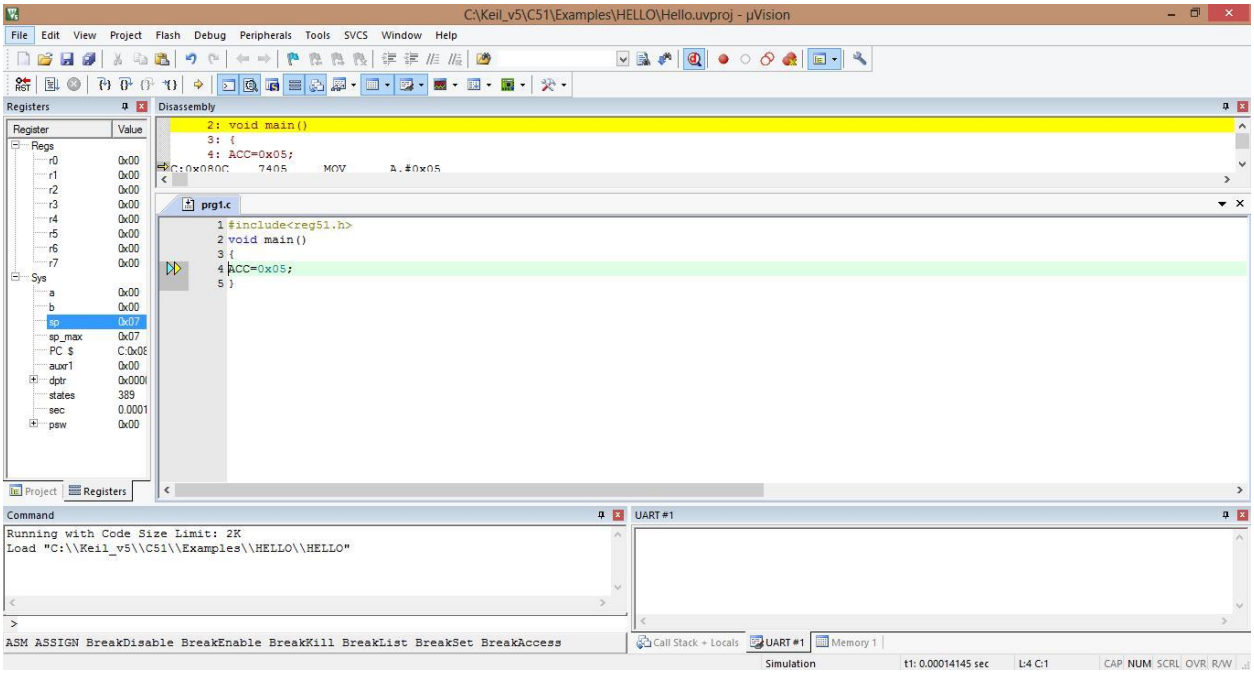
Check for any errors in program in the build output. Here displaying 0 errors and 0 warnings.

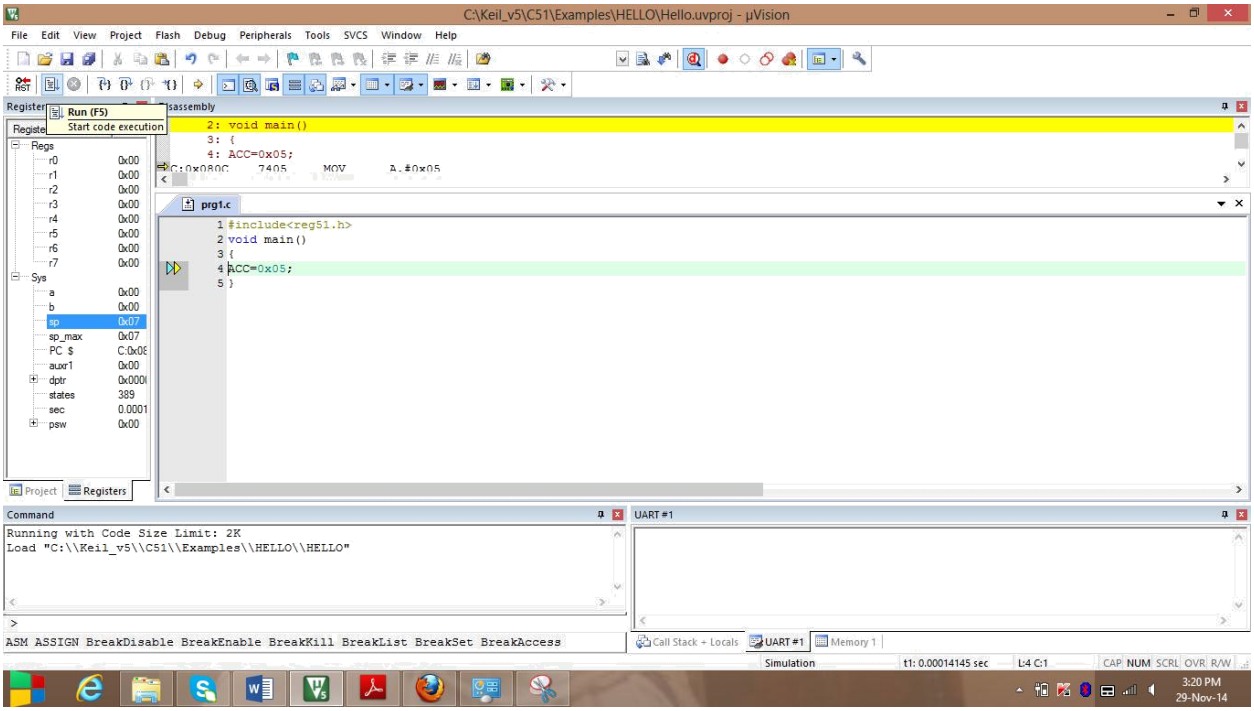


Start debugging session

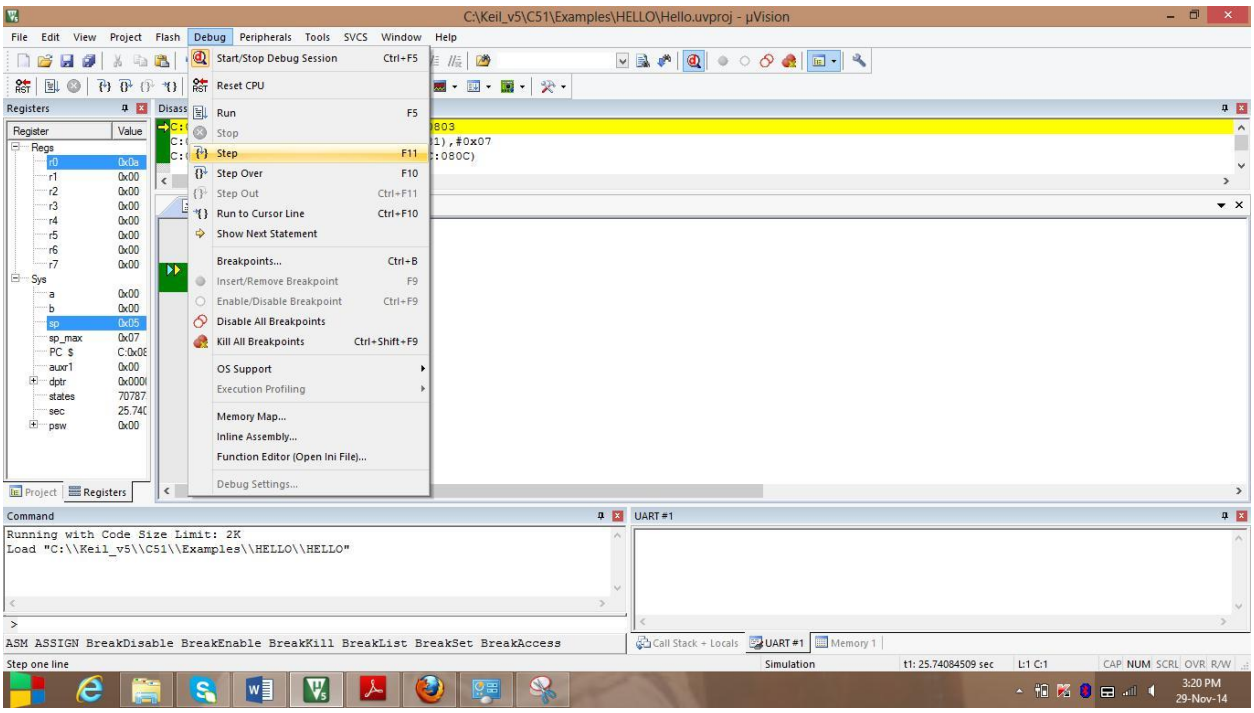


New window evaluation mode appeared. Press ok

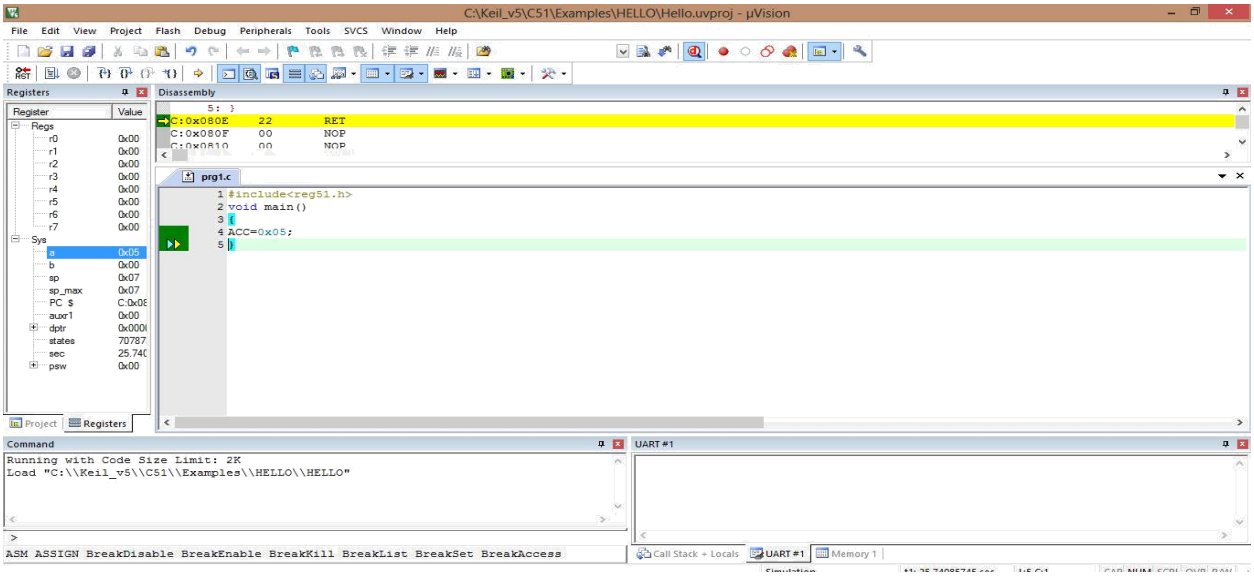




Run the program

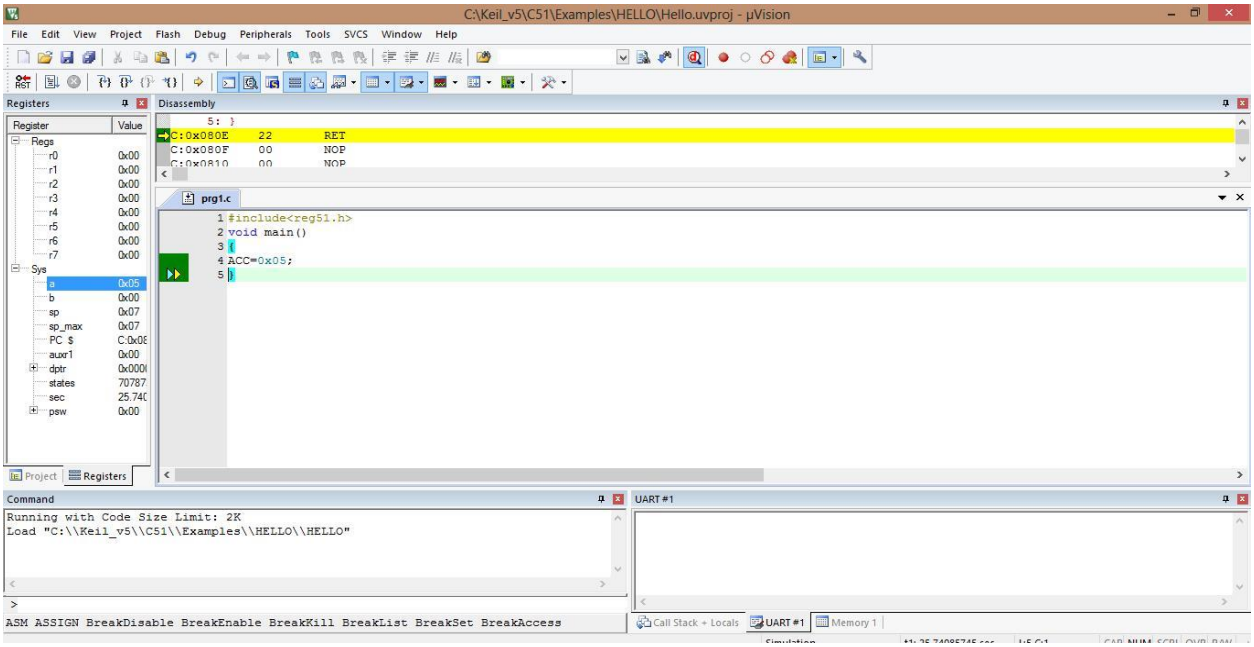


Select step from debug or press F11



Check the value of a register.

Output:-



The value of accumulator is 05

Observation:-

From the above output it is analyzed that the value 05 is stored in accumulator.

Conclusion:-

The experiment to store any data in accumulator using keil u vision successfully.

Experiment-4

Aim of the experiment:- Write a C Program to send values 00-ff to Port 1.

Software required: - Keil u vision 5

Theory: - This experiment aims to send value 00-ff using the software keil u vision 5. In this program, the data 00-ff is to be sent to Port1. Firstly the header files reg51.H is declared for the intended 8051. Then the main function starts. In the main function an unsigned character is initialized followed by a for loop where 0-255(00-ff) is assigned to Port1.

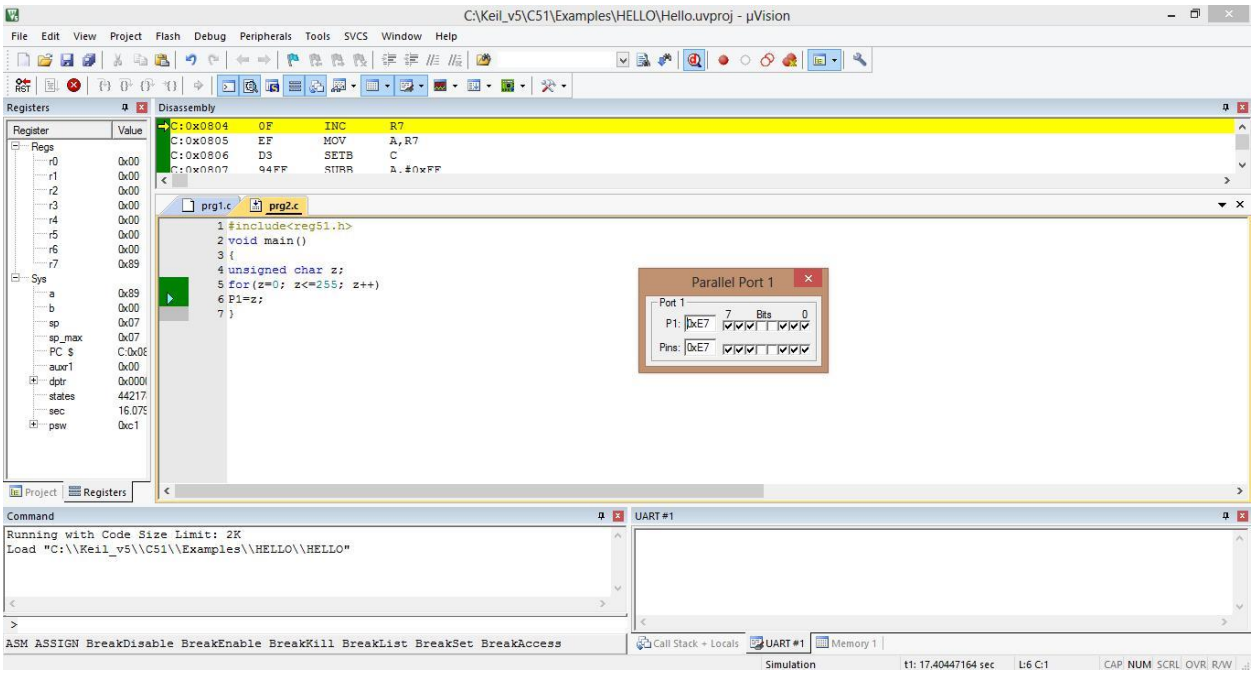
Program:-

```
#include<reg51.h>

void main()

{

unsigned char z;

for(z=0; z<=255; z++)

P1=z;

}
```

Procedure: - The procedure for the program of storing a data in accumulator is as follows.

File → new → Program(code) → Save → Add existing files to source group 1 →
Build → Start Debug → Run → Peripherals → Port1 → Debug the process

Output:-



Observation:-

From the above output it is observed that the data 00-ff is sent to Port1.

Conclusion:-

The experiment to send values 00-ff to Port 1 using keil u vision successfully

Experiment-5

Aim of experiment:- Write a C Program to send hex values for ASCII characters 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f to port 1

Software required:- Keil u vision 5

Theory :- This experiment aims to send hex values for ASCII characters 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f to port 1 using the software keil u vision 5. Firstly the header files reg51.H is declared for the intended 8051. Then the main function starts. In the main function an unsigned character array is initialized with"0123456789abcdef" and then unsigned character variable is initialized followed by a for loop. The loop runs for 16 time where each time one character from array is sent to Port1.

Program:-

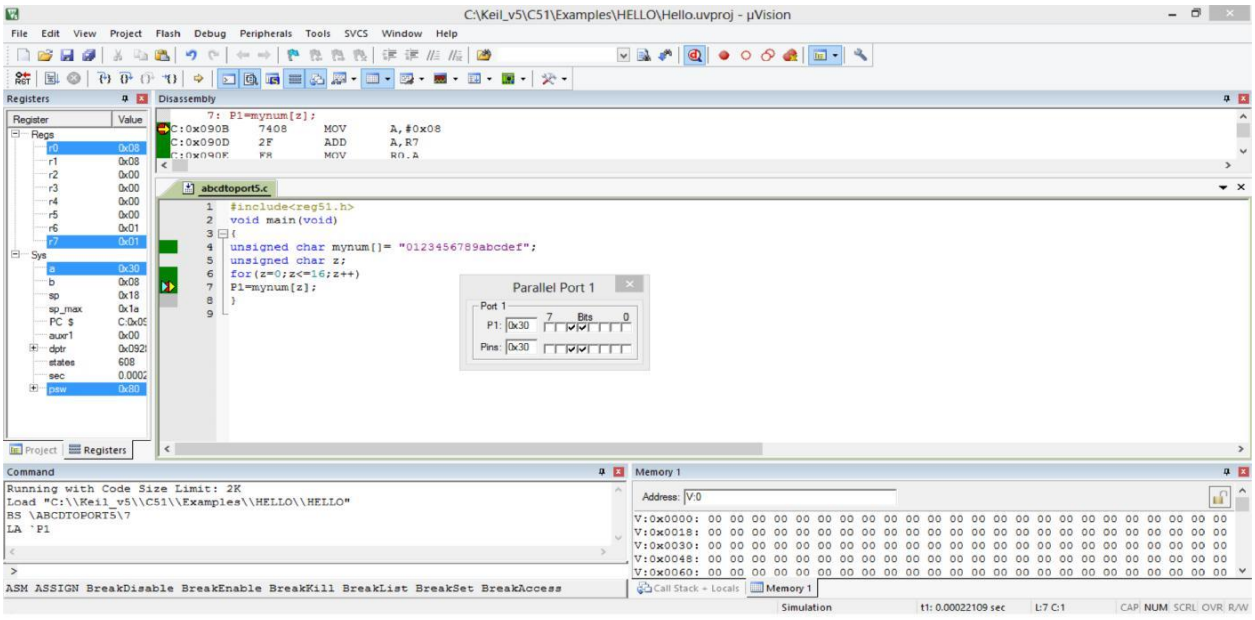
```
#include<reg51.h>

void main()

{

        unsigned char a[] = "0123456789abcdef";

        unsigned char z;

        for(z=0; z<=16; z++)

        P1=a[z];

}
```

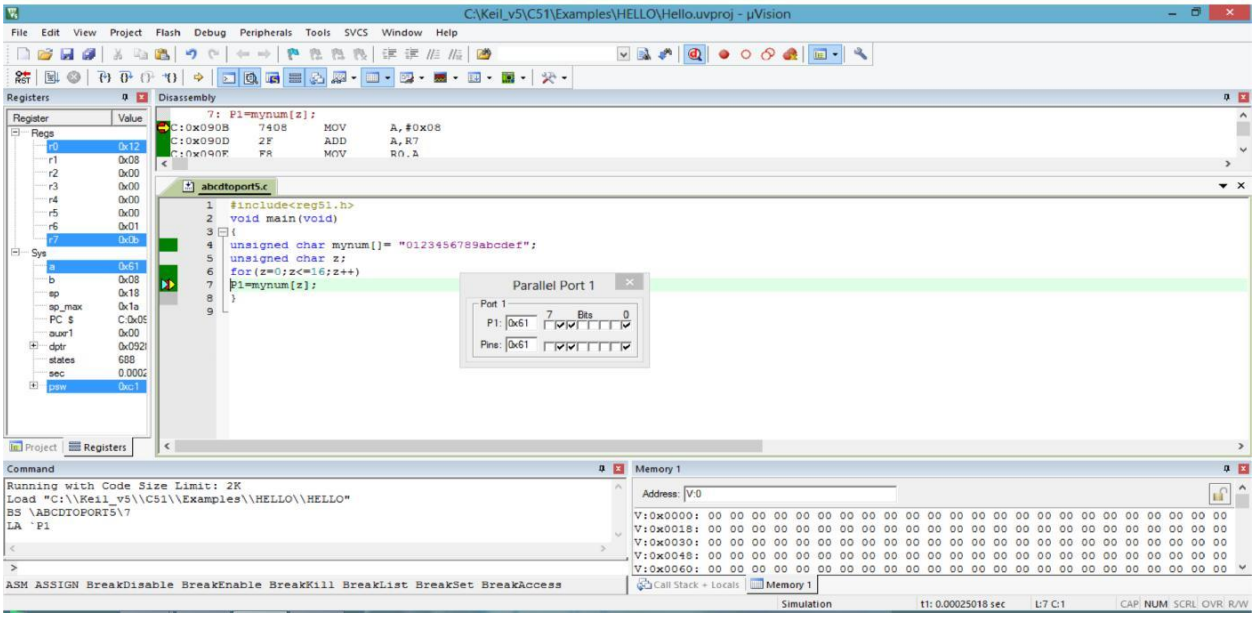
Procedure:- The procedure for the program of storing a data in accumulator is as follows.

File → new → Program(code) → Save → Add existing files to source group 1 → → Build → Start Debug → Run Peripherals → Port1 → Insert Break Point at P1 → Debug the process

Output:-

1. **For input '0'** 2. **For input 'a'**





Observation:- From the above output it is observed that the data 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f is sent to Port1.

Conclusion:- The experiment to send values 00-ff to Port 1 using keil u vision successfully.

Experiment-6

Aim of experiment:- Write a C program to toggle all the bits of P1 continuously.

Software required:- Keil u vision

Theory:- This experiment aims to toggle all the bits of Port1 continuously using the software keil u vision 5. Firstly the header files reg51.H is declared for the intended 8051. Then the main function starts. In the main function an infinite for loop starts where each time 0x55 followed by 0xAA is sent to Port1 and the loop goes on.

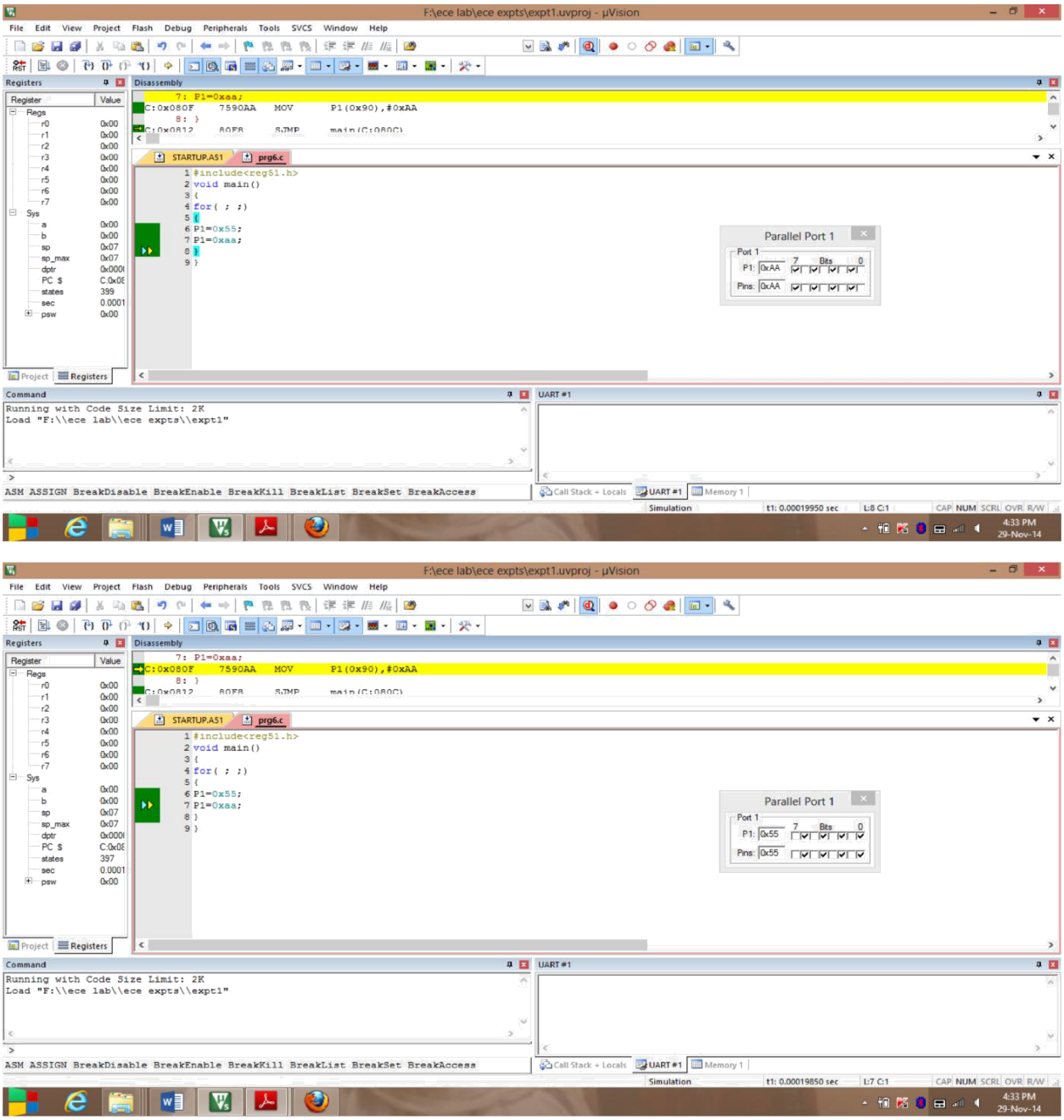
Program:-

```
#include <reg51.h>
void main()
{
for (;;)
{
p1=0x55;
p1=0xAA;
}
}
```

Procedure:- The procedure for the program of storing a data in accumulator is as follows.

File → new → Program(code) → Save → Add existing files to source group 1 → Build → Start Debug → Run → Peripherals → Port1 → Debug (step) → Debug the process

Output:-



Observation:- From the above output it is observed that all the bits of P1 are toggling continuously.

Conclusion:- The experiment to toggle all the bits of P1 continuously using keil u vision successfully

Experiment-7

Aim of experiment:- Write a C program to toggle bit D0 of port 1 50,000 times.

Software required:- keil u vision

Theory:- This experiment aims to toggle bit D0 of Port1 50,000 times using the software keil u vision 5. Firstly the header files reg51.H is declared for the intended 8051. The sbit keyword is a widely used 8051 C data type designed specifically to access single bits of SFR registers. Here we use sbit toaccess the individual bits of the Port1. Then the main function starts. In the main function a for loop runs 50000 times where each time D0 of Port1 is made 0 and then 1.

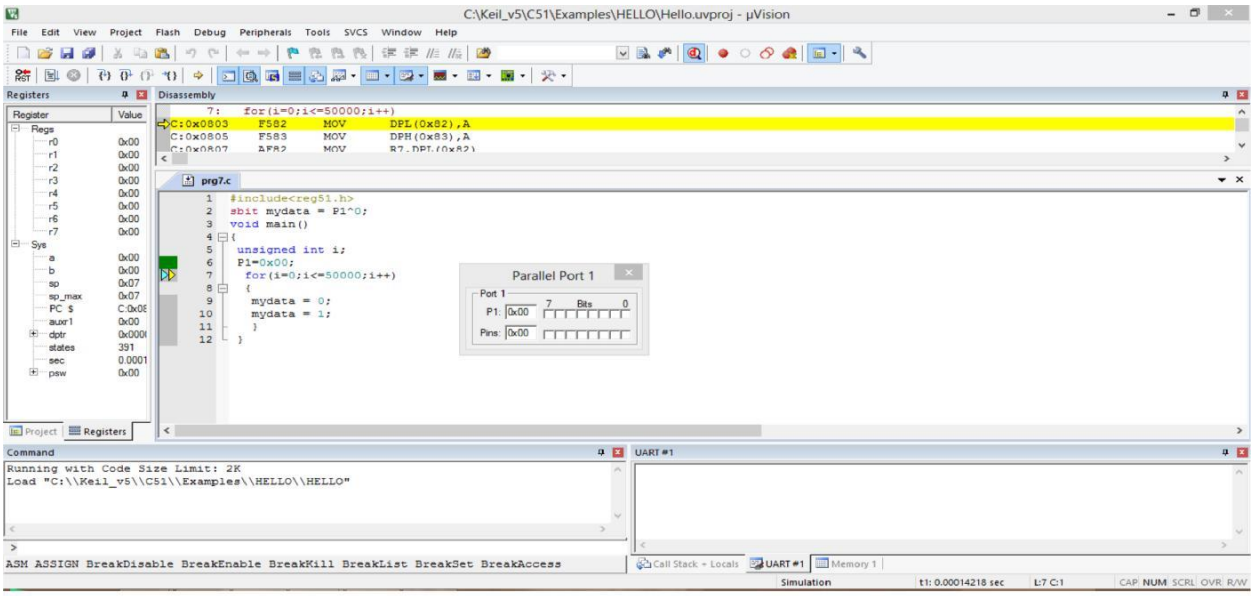
Program:-

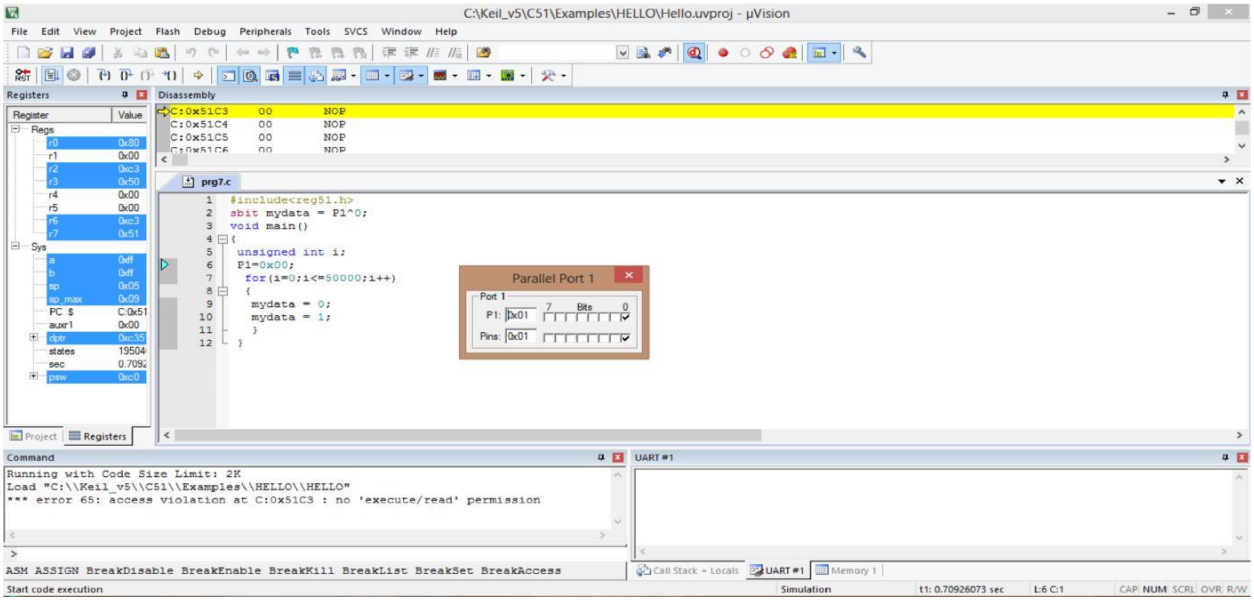
```
#include <reg51.h>
sbit MYBIT=P1^0;
void main(void)
{
unsigned int z;
for (z=0;z<=50000;z++)
{
MYBIT=0;
MYBIT=1;
}
}
```

Procedure:- The procedure for the program of storing a data in accumulator is as follows.

File → new → Program(code) → Save → Add existing files to source group 1 → Build → Start Debug → Run → Peripherals → Port1 → Debug (step) → Debug the process

Output:-





Observation:- From the above output it is observed that the bit D0 of the Port1 (P1.0) is toggled 50,000 times.

Conclusion:- The experiment to toggle bit D0 of port 1 50,000 times using keil u vision successfully

Experiment-8

Aim of experiment:- Write a C program to generate a square wave for 1ms delay.

Software required:- keil u vision

Theory:- This experiment aims to generate a square wave for 1ms delay using the software keil u vision 5. There are two ways to create a time delay in 8051 C: 1. Using a simple for loop 2. Using the 8051 timers.

Program:-

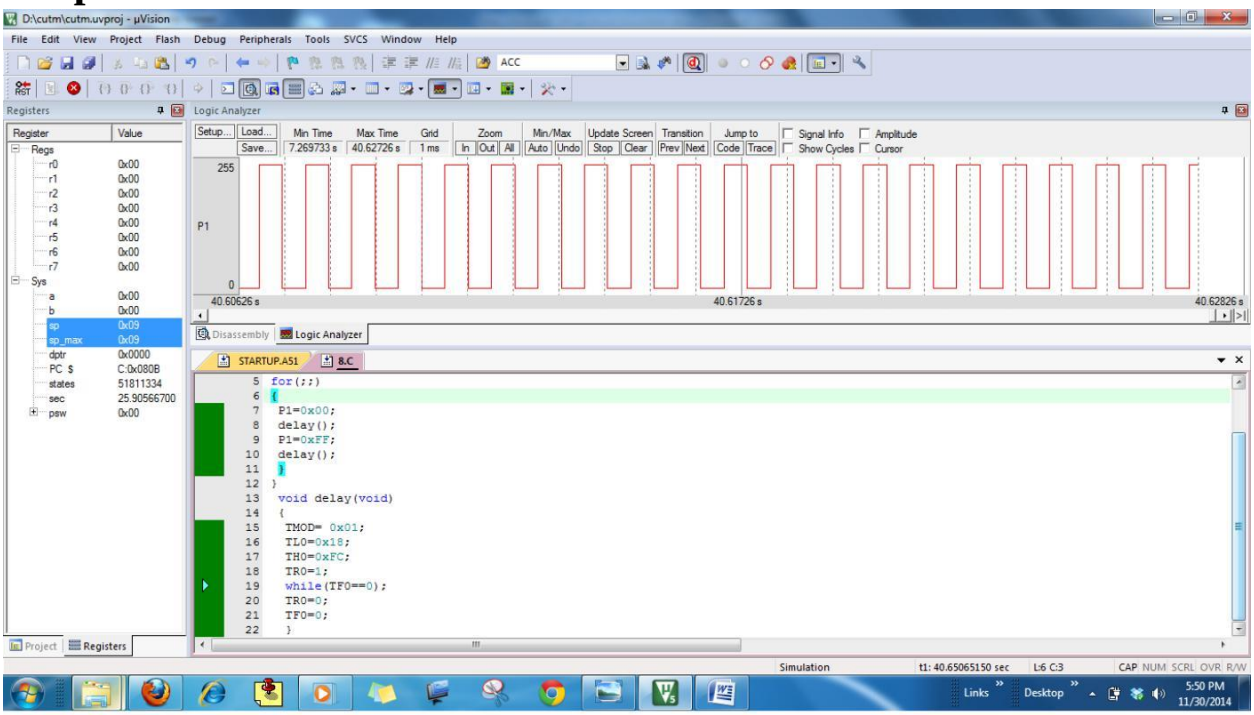
```
#include<reg51.h>
void delay( void );
void main()
{
 for(;;)
 {
  P1=0x00;
  delay();
  P1=0xFF;
  delay();
  }
}
void delay(void)
{
 TMOD=0x01;
 TL0=0X18;
 TH0=0XFC;
 TR0=1;
  while(TF0==0);
```

```
  TR0=0;
  TF0=0;
}
```

Procedure:- The procedure for the program of storing a data in accumulator is as follows.

File → new → Program(code) → Save → Add existing files to source group 1 → Build → Start Debug → Run → Analysis Window → Set up → New → Write P1 → Close → Run → Debug the process

Output:-



Observation:-

From the above output it is observed that the square wave is generated in the analysis window.

Conclusion:-

The experiment to to generate a square wave for 1ms delay using keil u vision is done successfully

Experiment-9

Aim of experiment :- Write a C program to send the data serially.

Software required:- keil u vision

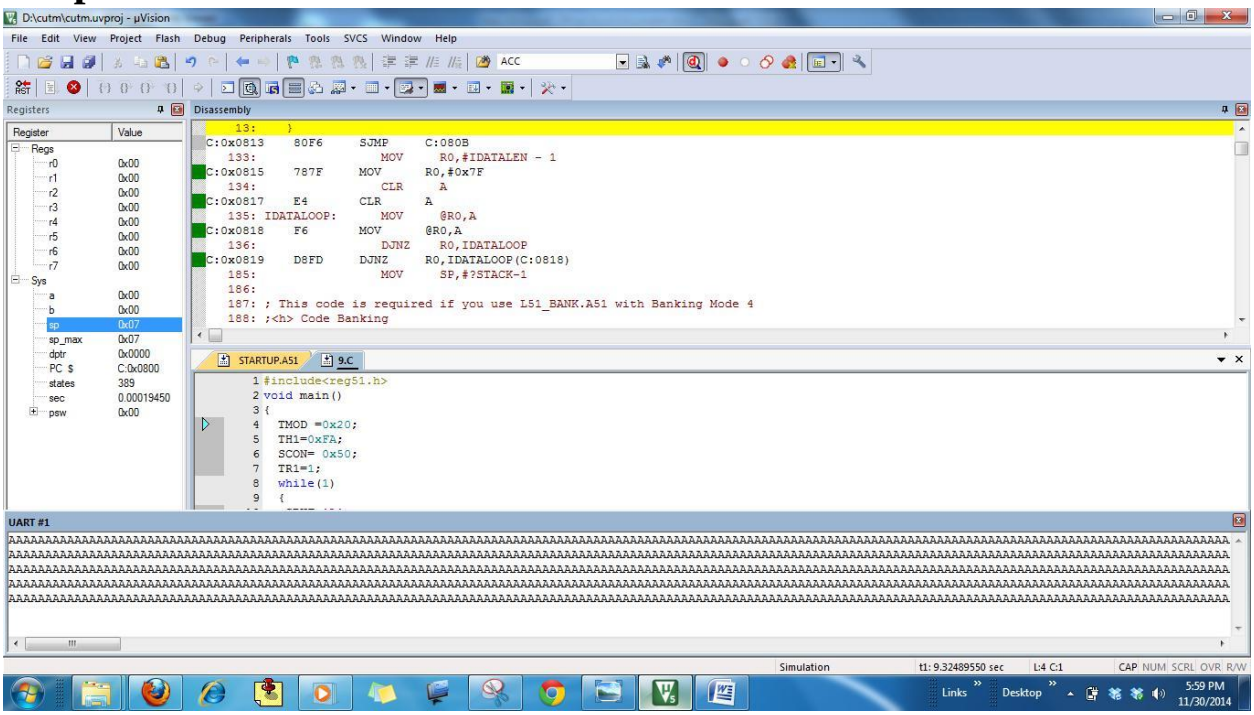
Theory:- This experiment aims to send the data serially using the software keil u vision 5.

Program:-

```
#include<reg51.h>
void main()
{
  TMOD=0X20;
  TH1=0XFA;
  SCON=0X50;
  TR1=1;
  while(1)
  {
     SBUF='A';
          while(TI==0)
          TI=0;
  }
}
```

Procedure:- Program(code) → Save → Add existing files to source group 1 → Build → Start Debug → Run → View → Serial Window → UART#1 → Check the output → Debug the process

Output:-



Observation:-

From the above output it is observed that the data is sent serially .

Conclusion:-

The experiment to to generate a square wave for 1ms delay using keil u vision is done successfully

Experiment-10

Aim of experiment:- Write a C program to receive the data serially

Software required:- keil u vision

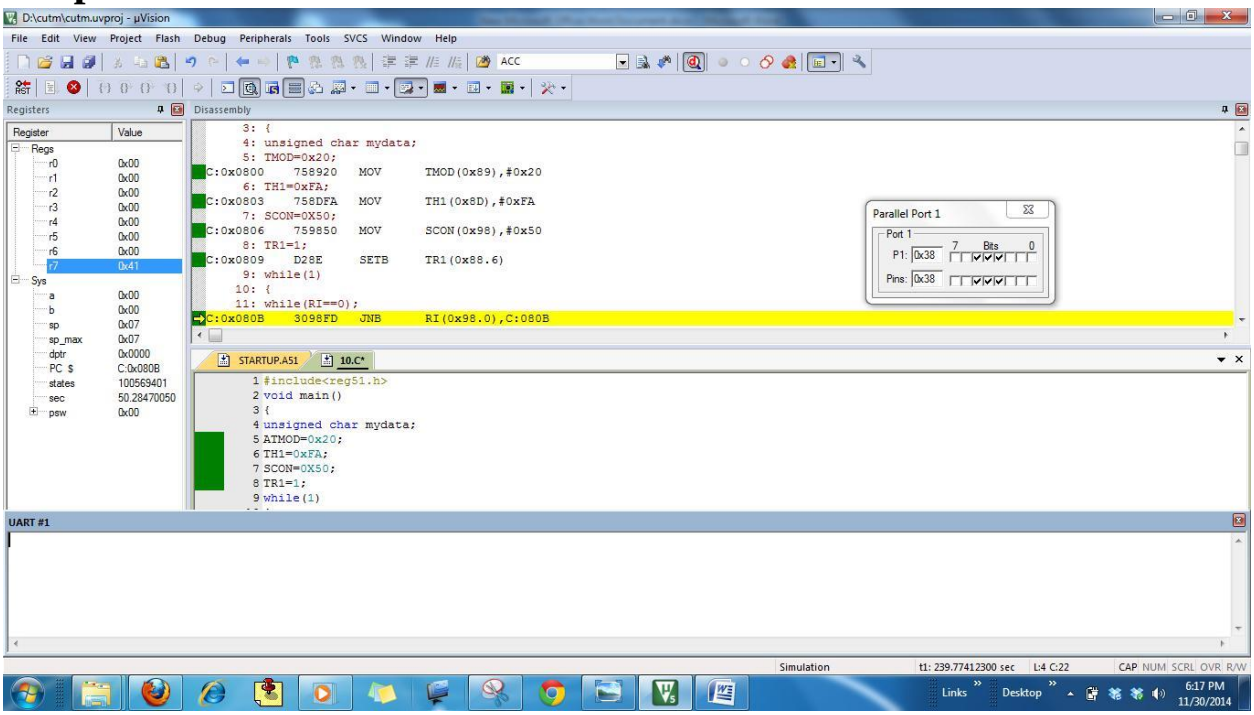
Theory:- This experiment aims to receive the data serially using the software keil u vision 5.

Program:-

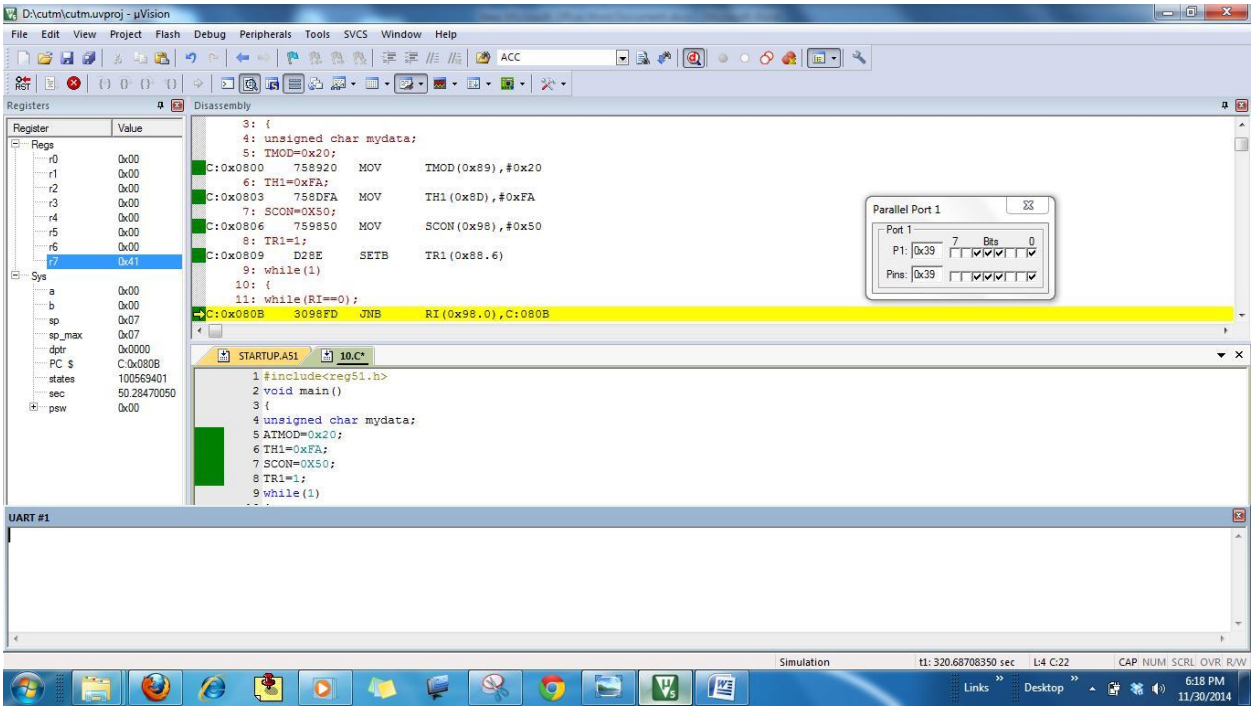
```
#include<reg51.h>
void main()
{
unsigned char mydata;
TMOD=0x20;
TH1=0xFA;
SCON=0X50;
TR1=1;
while(1)
{
while(RI==0);
mydata=SBUF;
P1=mydata;
RI=0;
}
}
```

Procedure:- File → new → Program(code) → Save → Add existing files to source group 1 → Build → Start Debug → Run → View → Serial Window → UART#1 → Check the output → Debug the process

Output:-



FOR 8 INPUT



FOR 9 INPUT

Observation:-

From the above output it is observed that the data as input 8 (in ASCII as 38) and input 9 (in ASCII as 39) is received serially

Conclusion:-

The experiment to receive the data serially using Keil u vision is done successfully

Experiment-11

Aim of the experiment:- Write a C program to convert packed BCD 0x29 to ASCII and Display the bytes on P1and P2.

Software required:- keil u vision

Theory:- This experiment aims to convert packed BCD 0x29 to ASCII. To covert packed BCD to ASCII, it must be converted to unpacked BCD is tagged with 00110000(30H).

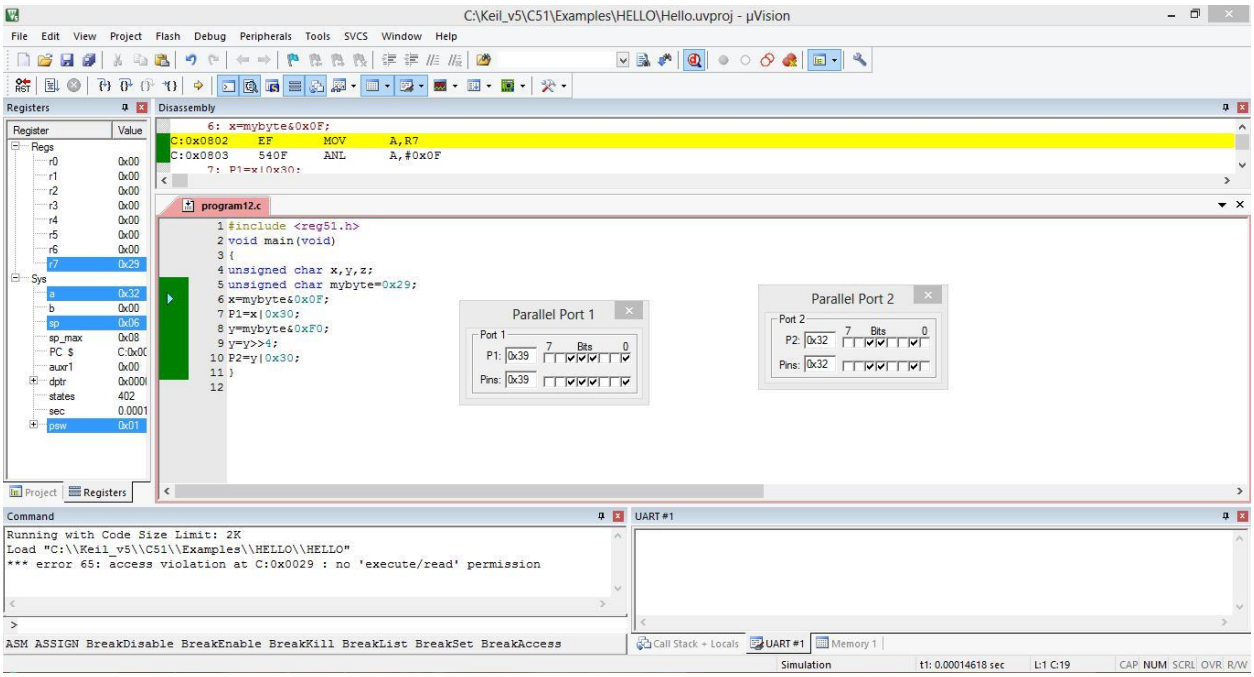
Packed BCD	Unpacked BCD	ASCII
0x29	0x02,0x09	0x32,0x39
00101001	00000010, 00001001	00110010,00111001

Program:-

```
#include <reg51.h>
void main(void)
{
unsigned char x,y,z;
unsigned char mybyte=0x29;
x=mybyte&0x0F;
P1=x|0x30;
y=mybyte&0xF0;
y=y>>4;
P2=y|0x30;
}
```

Procedure:- File → new → Program(code) → Save → Add existing files to source group 1 → Build → Start Debug → Run → Peripherals → Port1 → Peripherals → Port2 → Stop debug the process

Output:-



Observation:-

From the above output it is observed that packed BCD 0x29 is converted to ASCII and Port2 displays 0x32 (00110010) and Port1 displays 0x39 (00111001) .

Conclusion:-

The experiment to convert packed BCD 0x29 to ASCII using keil u vision successfully.

Experiment-12

Aim of the experiment:- Write a C program to convert the hex to decimal and display the digits on P0, P1, and P2.

Software required:- keil u vision

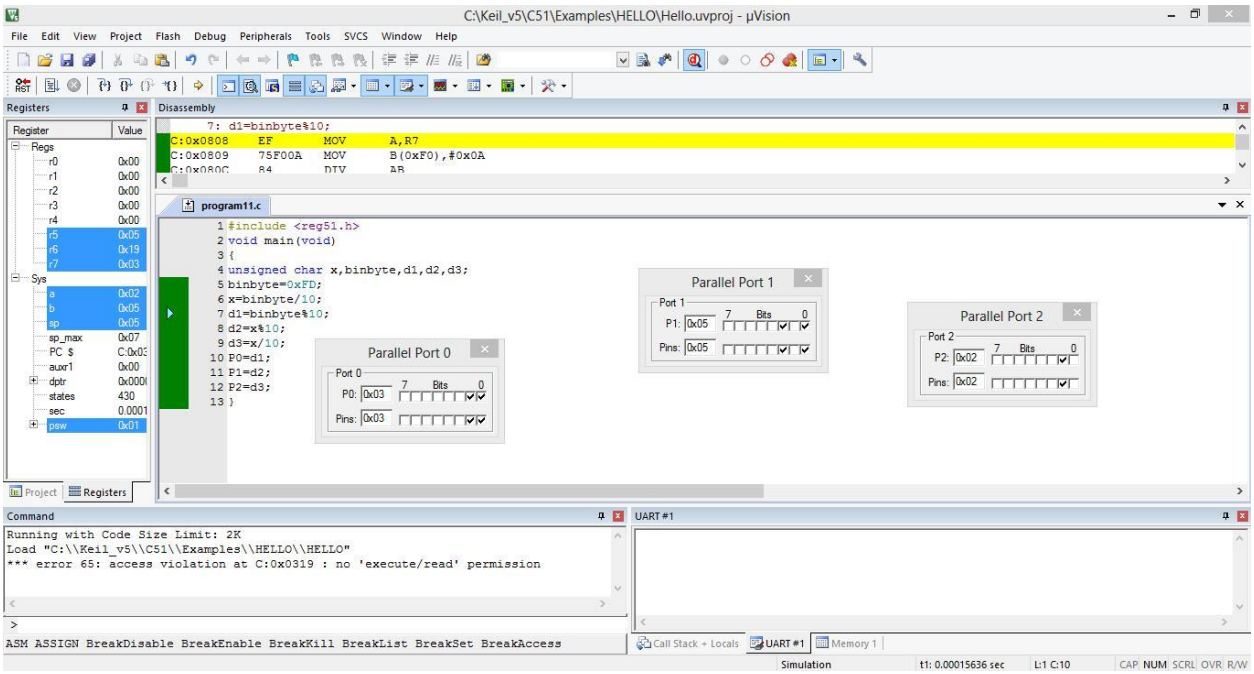
Theory:- One of the most widely used conversions is the binary to decimal coversion.in devices such as ADC(Analog to Digital Conversion) chips, the data is provided to the microcontroller in binary. In order to display binary data we need to convert it to decimal and then to ASCII. Since the hexadecimal format is a convenient way of representing binary data we refer to the binary data as hex. This experiment aims to to convert the hex to decimal using the software keil u vision 5. Firstly the header files reg51.H is declared for the intended 8051. Then the main function starts. One way to do this is to divide it by 10 and keep the remainder.

Program:-

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

Procedure:- File → new → Program(code) → Save → Add existing files to source group 1 → Build → Start Debug → Run → Peripherals → Port0 → Peripherals → Port1 → Peripherals → Port2 → Debug the process

Output:-



Observation:-

From the above output it is observed that the binary data 00-ff H is converted to decimal which will give us 000-255. In this program binary data 11111101 or FD H is converted to decimal 253 and the output is checked at Port0, Port1 and Port2.

Conclusion:-

The experiment to convert the hex to decimal and displaying the digits on P0, P1, and P2 using keil u vision successfully.