Lab Code: 20EC504/JO1A

EMBEDDED SYSTEM & DESIGN Lab Manual

Department of Electronics & Communication Engineering

Bapatla Engineering College :: Bapatla

(Autonomous)

G.B.C. Road, Mahatmajipuram, Bapatla-522102, Guntur (Dist.)
Andhra Pradesh, India.
E-Mail:bec.principal@becbapatla.ac.in
Web:www.becbapatla.ac.in

Contents

Bapatla Engineering College :: Bapatla
(Autonomous)

Vision

- To build centers of excellence, impart high quality education and instill high standards of ethics and professionalism through strategic efforts of our dedicated staff, which allows the college to effectively adapt to the ever changing aspects of education.

- To empower the faculty and students with the knowledge, skills and innovative thinking to facilitate discovery in numerous existing and yet to be discovered fields of engineering, technology and interdisciplinary endeavors.

Mission

- Our Mission is to impart the quality education at par with global standards to the students from all over India and in particular those from the local and rural areas.

- We continuously try to maintain high standards so as to make them technologically competent and ethically strong individuals who shall be able to improve the quality of life and economy of our country.

Bapatla Engineering College :: Bapatla

(Autonomous)

Department of Electronics and Communication Engineering

Vision

To produce globally competitive and socially responsible Electronics and Communication Engineering graduates to cater the ever changing needs of the society.

Mission

- To provide quality education in the domain of Electronics and Communication Engineering with advanced pedagogical methods.

- To provide self learning capabilities to enhance employability and entrepreneurial skills and to inculcate human values and ethics to make learners sensitive towards societal issues.

- To excel in the research and development activities related to Electronics and Communication Engineering.

Bapatla Engineering College :: Bapatla

(Autonomous)

Department of Electronics and Communication Engineering

Program Educational Objectives (PEO's)

PEO-I: Equip Graduates with a robust foundation in mathematics, science and Engineering Principles, enabling them to excel in research and higher education in Electronics and Communication Engineering and related fields.

PEO-II: Impart analytic and thinking skills in students to develop initiatives and innovative ideas for Start-ups, Industry and societal requirements.

PEO-III: Instill interpersonal skills, teamwork ability, communication skills, leadership, and a sense of social, ethical, and legal duties in order to promote lifelong learning and Professional growth of the students.

Program Outcomes (PO's)

Engineering Graduates will be able to:

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

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

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

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

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

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

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

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

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

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

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

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

Bapatla Engineering College :: Bapatla

(Autonomous)

Department of Electronics and Communication Engineering

Program Specific Outcomes (PSO’s)

PSO1: Develop and implement modern Electronic Technologies using analytical methods to meet current as well as future industrial and societal needs.

PSO2: Analyze and develop VLSI, IoT and Embedded Systems for desired specifications to solve real world complex problems.

PSO3: Apply machine learning and deep learning techniques in communication and signal processing.

EMBEDDED SYSTEM & DESIGN
III B.Tech. V Semester (Code:20EC504/JO1-A)

Lectures	:	2 Hours/Week	Tutorial	:	0 Hour/Week	Practical	:	2 Hour/Week
CIE Marks	:	30	SEE Marks	:	70	Credits	:	3

Pre-Requisite: Microprocessors, and Microcontrollers.

Course Objectives: Students will learn how to

➢	Learn basic design and architectural concepts of embedded systems.
➢	Understand the concepts of Real-Time Operating Systems and provide the scheduling Algorithms
➢	Familiarize with the fundamentals of prevalent IP-Core: ARM Cortex M3/M4 & Design of an embedded system using ARM Cortex Processor
➢	Be able to use the instruction set of ARM Cortex M3/M4 processor and explain the ALP's Using ARM processor

Course Outcomes: After studying this course, the students will be able to

CO1	Describe different methodologies and approaches in the design of embedded systems
CO2	Analyze the concepts of Real-Time Operating systems and scheduling Algorithms.
CO3	Illustrate the features, basic architecture and memory management unit of ARM Processors
CO4	Simulate ARM Programming models using Keil µVision for different embedded Applications.

Mapping of Course Outcomes with Program Outcomes & Program Specific Outcomes

	PO's												PSO's		
CO	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**1**	**2**	**3**
CO1	3													3	
CO2	2	3												3	
CO3	3													3	
CO4	2	3	2		3									3	
AVG	2.5	3	2		3									3	

Syllabus

UNIT-1: EMBEDDED SYSTEMS DESIGN: Introduction to Embedded System, categories of embedded system, specialties, and recent trends in Embedded System.
ARCHITECTURE OF AN EMBEDDED SYSTEM: Hardware Architecture, Software Architecture, application Software, Communication Software, Development/Testing Tools
UNIT-2 : OVERVIEW OF RTOS: Architecture of the Kernel, Tasks, Task scheduler, real-time tasks, Task scheduling, Interrupt Service Routine, Memory Management, Semaphores, Mutex, Mailboxes, Message Queues, Event Registers, Pipes.
CLASSIFICATION OF SCHEDULING ALGORITHMS: Clock driven Scheduling, Event driven Scheduling, Resource sharing, Priority inversion problem, Deadlock.
UNIT-3: EMBEDDED PROCESSORS: Introduction to ARM family, ARM Architecture-Pipeline, Registers, Operation modes, Big Endian and Little Endian. Cache Mechanism, Memory Management Unit
UNIT-4: ARM INSTRUCTIONS: ARM and Thumb Instruction Sets, Data Processing Instructions, Data Transfer Instructions, Control Flow Instructions, Basic Assembly Language Programs. Case Study: Smart Phone, Digital Camera, and Automatic Washing Machine

PRACTICAL EXERCISES

1. Exploring the features of Keil and RTX51
2. Task Creation and Deletion usingRTX51in Keil
3. TaskschedulingusingRTX51in Keil
4. Processing Critical Section using RTX51inKeil
5. Task Synchronization using RTX51semaphores in Keil
6. Task Communication using shared memory in Keil
7. Task Communication using RTX51 mailbox in Keil
8. Introduction to ARM Cortex M3Processor
9. ALP to multiplytwo16-bit binary numbers
10. ALP to find the sum of the first 10integers.
11. ALP to find the number of 0'sand 1'sin32-bit data.
12. ALP to determine whether the given16-bitnumber is ODD or EVEN.
13. ALP to write data in RAM(CO4)
14. Display Hello World message using Internal UART.
15. Interface a Stepper motor and rotate it in clock wise and anti-clock wise direction.

***Any Ten programs Compulsory.**

1. Exploring the Features of Keil and RTX51

AIM: To understand and utilize the features of Keil and RTX51 for real-time operating system development.

APPARATUS:

1. Keil uVision IDE
2. RTX51 operating system
3. Microcontroller (e.g. STM32)

THEORY:

Keil is a development environment that provides a comprehensive set of tools for creating embedded systems. It includes a compiler, assembler, linker, and debugger, as well as a robust editor and project manager. Keil supports a wide range of microcontrollers and provides a flexible and customizable development environment.

RTX51 is a real-time operating system that provides multitasking, synchronization, and communication mechanisms. It is designed to be highly efficient and flexible, making it suitable for a wide range of applications. RTX51 provides a number of features, including task creation and management, synchronization mechanisms such as semaphores and mutexes, and communication mechanisms such as mailboxes and message queues.

One of the key features of RTX51 is its support for multitasking. This allows multiple tasks to run concurrently, improving the overall efficiency and responsiveness of the system. RTX51 also provides a number of synchronization mechanisms, including semaphores and mutexes, which allow tasks to coordinate their access to shared resources.

In addition to multitasking and synchronization, RTX51 also provides a number of communication mechanisms, including mailboxes and message queues. These allow tasks to exchange data and coordinate their actions, making it easy to implement complex systems.

Overall, Keil and RTX51 provide a powerful platform for real-time operating system development. Their flexibility, efficiency, and robustness make them suitable for a wide range of applications, from small embedded systems to large and complex systems.

By leveraging the features of Keil and RTX51, developers can create high-performance and reliable real-time systems that meet the demands of today's embedded systems applications. Whether you are developing a small microcontroller-based system or a large and complex system, Keil and RTX51 provide the tools and features you need to succeed.

PROGRAM:

```
#include <rtx51.h>
// Define a task
void task1(void) {
  while (1) {
    // Task 1 code here
    os_delay(100); // Delay 100ms
  }
}
// Define another task
void task2(void) {
  while (1) {
    // Task 2 code here
    os_delay(200); // Delay 200ms
  }
}
// Main function
void main(void) {
  // Initialize RTX51
  os_init();
  // Create tasks
  os_task_create(task1, 0);
  os_task_create(task2, 0);
  // Start scheduler
  os_start_scheduler();
}
```

PROCEDURE:

1. Create a new project in Keil uVision IDE
2. Write a program that creates multiple tasks using RTX51
3. Use synchronization mechanisms to coordinate task access to shared resources
4. Use communication mechanisms to exchange data between tasks
5. Compile and run the program
6. Observe and record the results

RESULT: Observed the basic program and how to use Keil and RTX51.

2. TASK CREATION AND DELEATION USING RTX51

Aim: Write an embedded C program for 8051 microcontrollers using RTX51 Tiny to create a task and delete it. Report on the code execution statistics by identify the time-consuming module for optimization

Apparatus required: 1.PC with Windows 10 64-bit OS.

2.Keil µVision5 Software,

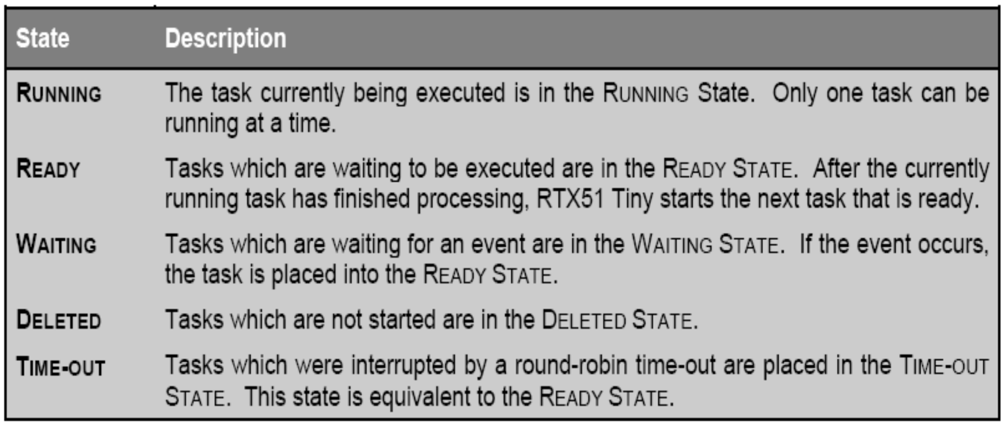
Theory: Task is a unit of work or part of a program in execution. RTX51 Tiny supports a maximum of 16 standard tasks. They use individual register bank of 8051 and have its own stack area. Standard tasks require relatively more time for task switching and lesser internal memory. They share a register bank and stack area. During standard task switch, current contents of registers and stack are stored in external RAM.

RTX51 Tiny Task Declaration Syntax:

voidfunc (void) _task_ num

where, num is a task ID number from 0 to 15.

RTX51 Tiny Task States:



State	Description
RUNNING	The task currently being executed is in the RUNNING State. Only one task can be running at a time.
READY	Tasks which are waiting to be executed are in the READY STATE. After the currently running task has finished processing, RTX51 Tiny starts the next task that is ready.
WAITING	Tasks which are waiting for an event are in the WAITING STATE. If the event occurs, the task is placed into the READY STATE.
DELETED	Tasks which are not started are in the DELETED STATE.
TIME-OUT	Tasks which were interrupted by a round-robin time-out are placed in the TIME-OUT STATE. This state is equivalent to the READY STATE.

RTX51 Tiny Task Management Functions:

Function Name	Parameter	Description
os_create_task	*unsigned char task_number* Number of task to be started (= number used for task declaration).	Creates a task and includes it in dispatching.
os_delete_task	*unsigned char task_number* Number of task to be terminated.	Terminates a task.
os_running_task_id	*(void)*	Returns the number (identification) of the running task.

PROGRAM:

cprog.c

```
#include <reg51.h>
#include <rtx51tny.h>
#include <stdio.h>
extern char putchar(char c);
extern char getkey(void);
extern void serialinit(void);
void TASK0(void) _task_ 0
{
serialinit();
printf("Task Creation/Deletion Example using RTX51 Tiny
RTOS\n");
 printf("CPU: 8xC51RD2, Memory Model: Large, Code ROM Size:
 Large, XTAL:
11.0592 MHz\n");
printf("In Options for Target -> BL51 Misc -> Overlay -> * ! getkey,
printf ! *\n");
os_create_task(1);
os_delete_task(0);
}
void TASK1(void) _task_ 1
{
char c = ' ';
signed char sc;
sc = os_running_task_id();
printf("Task with ID %d created.\n", (int)sc);
printf("\nTask 1 Deleting\n");
os_delete_task(sc);
}
```

serial.c

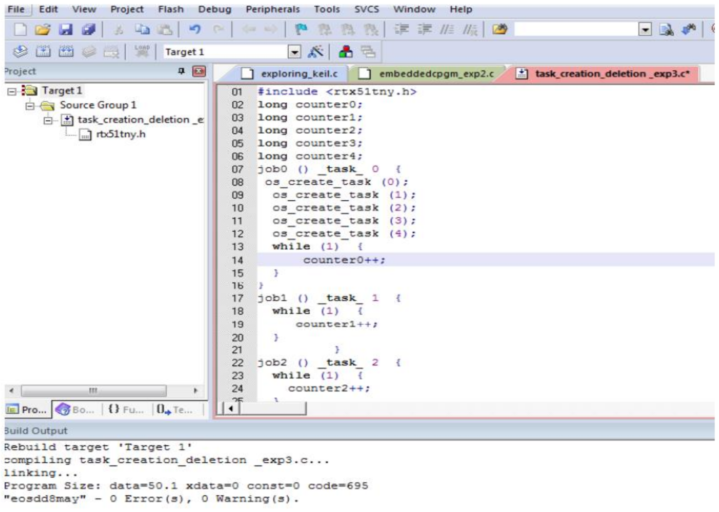
```
#include <reg51.h>
#define BAUDRATE 9600
#define CRYSTAL 11059200
#define CONST (((CRYSTAL / 12) / BAUDRATE) >> 5)
charputchar(char c);
voidserialinit(void);
chargetkey(void);
voidserialinit(void)
{
EA = 1;
ES = 0;
PCON = 0X00;
TMOD = 0X20;
TH1 = 0X0FF + 1 - CONST;
SCON = 0X50;
TR1 = 1;
}
charputchar(char c)
{
SBUF = c;
while(!TI);
TI = 0;
return(c);
}
chargetkey(void)
{
char c;
while(!RI);
c = SBUF;
RI = 0;
return(c);
}
```

PROCEDURE:

1. Open Keil µVision5 Software in a PC.
2. Create a new Project by clicking on Project menu followed by New Project submenu and give it a name.
3. Select the target device as 8xC51RD2.
4. Configure the Options for Target for BL51 appropriately.
5. Create two new source files by clicking on File menu followed by New submenu.
6. Type the C code and save it.

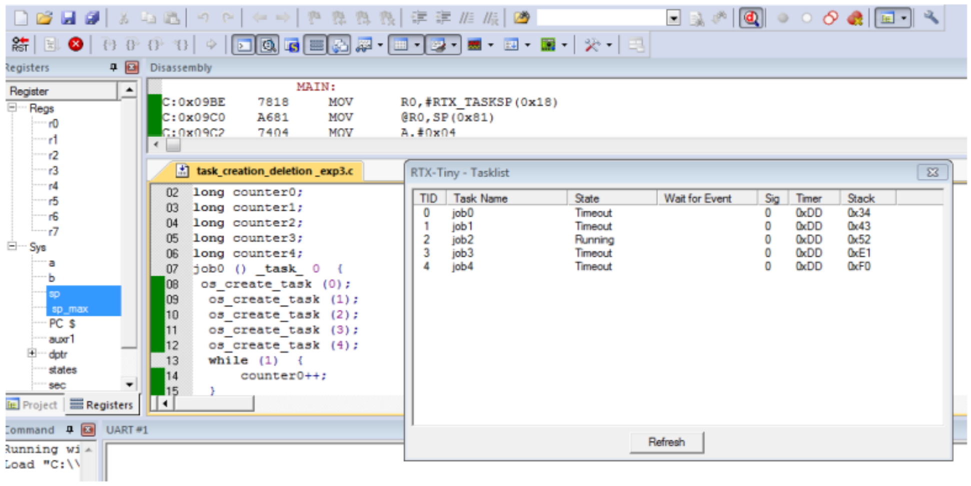
7. Add the required source files into the project by right mouse clicking on Source Group 1 in Project Window and select Add Files to Group ‘Source Group 1’.
8. Build the target file by clicking on Project menu followed by Build target submenu.
9. Correct the errors if any and debug the project by clicking on Debug menu followed by Start/Stop Debug Session submenu.
10. Open the required peripherals and debug the project.

OUTPUT:



```
#include <rtx51tny.h>
long counter0;
long counter1;
long counter2;
long counter3;
long counter4;
job0 () _task_ 0  {
 os_create_task (0);
  os_create_task (1);
  os_create_task (2);
  os_create_task (3);
  os_create_task (4);
  while (1)  {
        counter0++;
  }
}
job1 () _task_ 1  {
  while (1)  {
      counter1++;
  }
          }
job2 () _task_ 2  {
  while (1)  {
     counter2++;
```

```
Rebuild target 'Target 1'
compiling task_creation_deletion _exp3.c...
linking...
Program Size: data=50.1 xdata=0 const=0 code=695
"eosdd8may" - 0 Error(s), 0 Warning(s).
```



```
                    MAIN:
C:0x09BE    7818      MOV       R0,#RTX_TASKSP(0x18)
C:0x09C0    A681      MOV       @R0,SP(0x81)
C:0x09C2    7404      MOV       A,#0x04
```

```
long counter0;
long counter1;
long counter2;
long counter3;
long counter4;
job0 () _task_ 0  {
 os_create_task (0);
  os_create_task (1);
  os_create_task (2);
  os_create_task (3);
  os_create_task (4);
  while (1)  {
        counter0++;
  }
```

RTX-Tiny - Tasklist

TID	Task Name	State	Wait for Event	Sig	Timer	Stack
0	job0	Timeout		0	0xDD	0x34
1	job1	Timeout		0	0xDD	0x43
2	job2	Running		0	0xDD	0x52
3	job3	Timeout		0	0xDD	0xE1
4	job4	Timeout		0	0xDD	0xF0

```
Running wi
Load "C:\\
```

3. TASK SCHEDULING USING RTX51 TINY

AIM: Write an embedded C program for an 8051 microcontroller using RTX51 to create two tasks and schedule them using round robin scheduling algorithm with a time slice of 1000 units. Report on the code execution statistics by identifying the time-consuming module for optimization.

APPARATUS: 1.PC with Windows 10 64-bit OS.
2.Keil µVision5 Software,

THEORY: A task is a unit of work or part of a program in execution. RTX51 Tiny supports a maximum of 16 standard tasks. They use an individual register bank of 8051 and have their own stack area. Standard tasks require relatively more time for task switching and lesser internal memory. They share a register bank and stack area. During the standard task switch, the current contents of registers and stack are stored in external RAM.

PROGRAM:

cprog.c

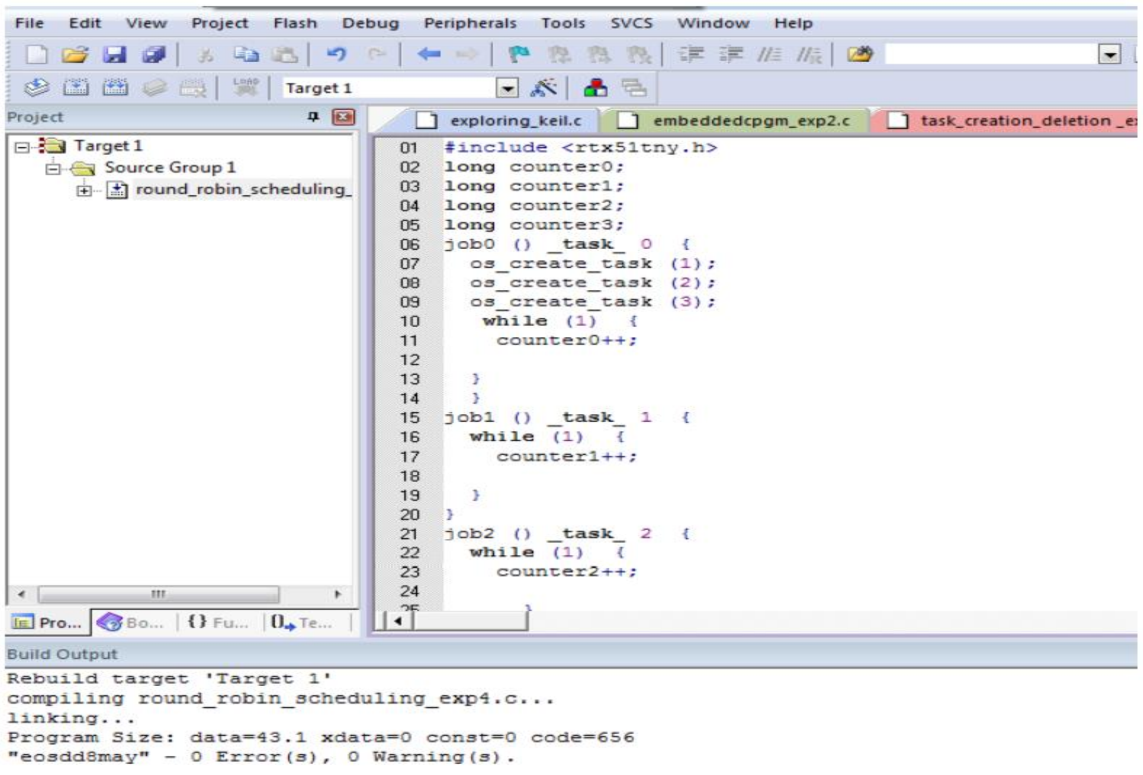
```
#include <reg51.h>
#include <rtx51tny.h>
#include <stdio.h>
extern char putchar(char c);
extern char getkey(void);
extern void serialinit(void);
void TASK0(void) _task_ 0
{
serialinit();
printf("CPU: 8xC51RD2, Memory Model: Large, Code ROM Size:
Large, XTAL:
11.0592 MHz\n");
printf("In Options for Target -> BL51 Misc -> Overlay -> * ! getkey,
printf ! *\n");
os_create_task(1);
os_create_task(2);
os_delete_task(0);
}
```

```
void task1(void) _task_ 1
{
while(1)
{
printf("Task 1 Running and Task 2 Ready.\n");
P0++;
}
}
void task2(void) _task_ 2
{
while(1)
{
printf("Task 2 Running and Task 1 Ready.\n");
P1++;
}
}
```

PROCEDURE:

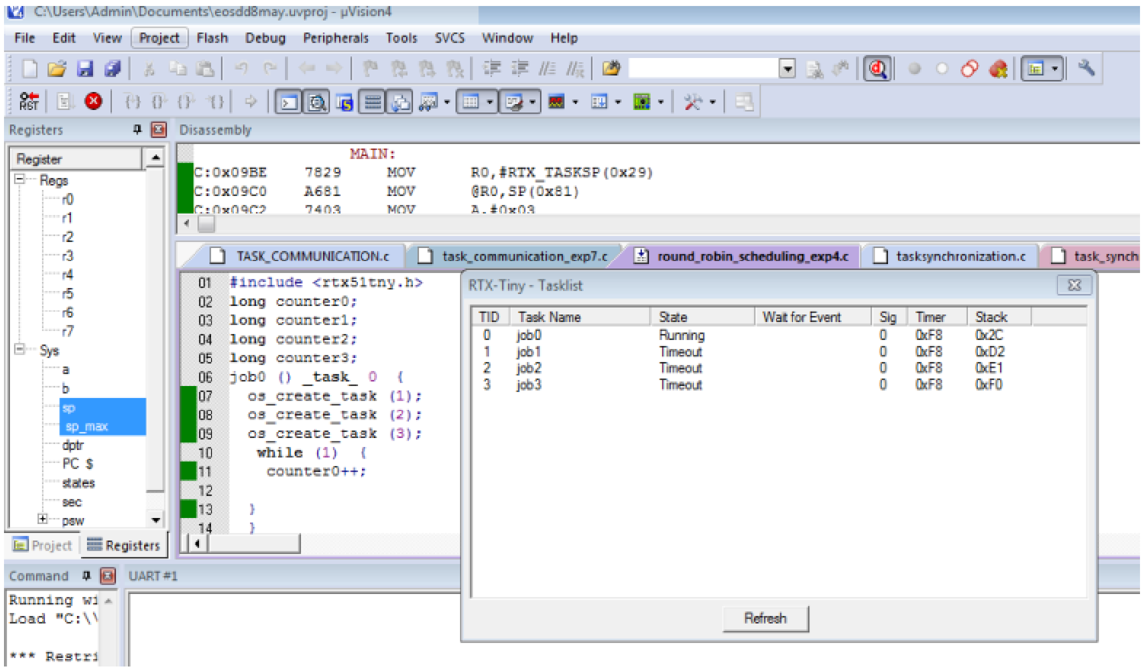
1. Open Keil µVision5 Software in a PC.
2. Create a new Project by clicking on Project menu followed by New Project submenu and give it a name.
3. Select the target device as 8xC51RD2.
4. Configure the Options for Target for BL51 appropriately.
5. Create two new source files by clicking on File menu followed by New submenu.
6. Type the C code and save it.
7. Add the required source files into the project by right mouse clicking on Source Group 1 in Project Window and select Add Files to Group 'Source Group 1'.
8. Build the target file by clicking on Project menu followed by Build target submenu.
9. Correct the errors if any and debug the project by clicking on Debug menu followed by Start/Stop Debug Session submenu.
10. Open the required peripherals and debug the project.

OUTPUT:



```
#include <rtx51tny.h>
long counter0;
long counter1;
long counter2;
long counter3;
job0  ()  _task_  0   {
  os_create_task  (1);
  os_create_task  (2);
  os_create_task  (3);
   while  (1)   {
     counter0++;

  }
  }
job1  ()  _task_  1   {
  while  (1)   {
     counter1++;

  }
}
job2  ()  _task_  2   {
  while  (1)   {
     counter2++;
```

```
Rebuild target 'Target 1'
compiling round_robin_scheduling_exp4.c...
linking...
Program Size: data=43.1 xdata=0 const=0 code=656
"eosdd8may" - 0 Error(s), 0 Warning(s).
```



```
                    MAIN:
C:0x09BE      7829      MOV        R0,#RTX_TASKSP(0x29)
C:0x09C0      A681      MOV        @R0,SP(0x81)
C:0x09C2      7403      MOV        A,#0x03
```

```
#include <rtx51tny.h>
long counter0;
long counter1;
long counter2;
long counter3;
job0  ()  _task_  0   {
  os_create_task  (1);
  os_create_task  (2);
  os_create_task  (3);
   while  (1)   {
     counter0++;

  }
  }
```

TID	Task Name	State	Wait for Event	Sig	Timer	Stack
0	job0	Running		0	0xF8	0x2C
1	job1	Timeout		0	0xF8	0xD2
2	job2	Timeout		0	0xF8	0xE1
3	job3	Timeout		0	0xF8	0xF0

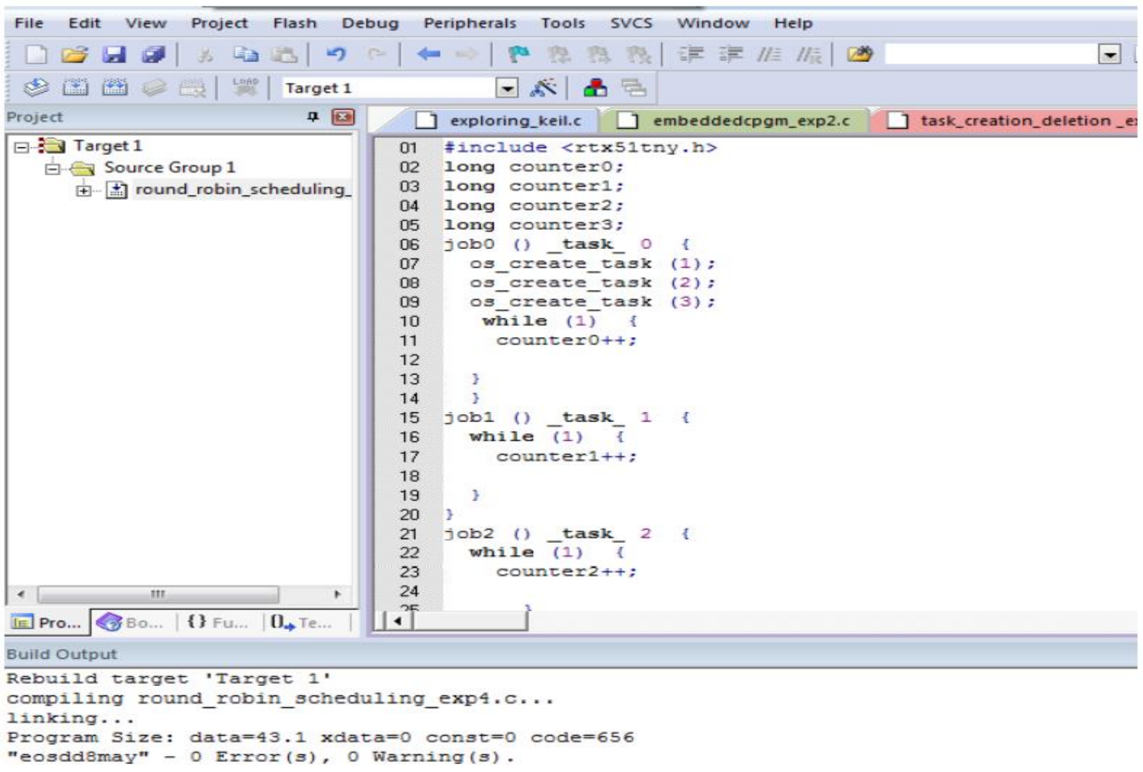
```
Running wi
Load "C:\\

*** Restri
```

PROGRAM:

cprog.c

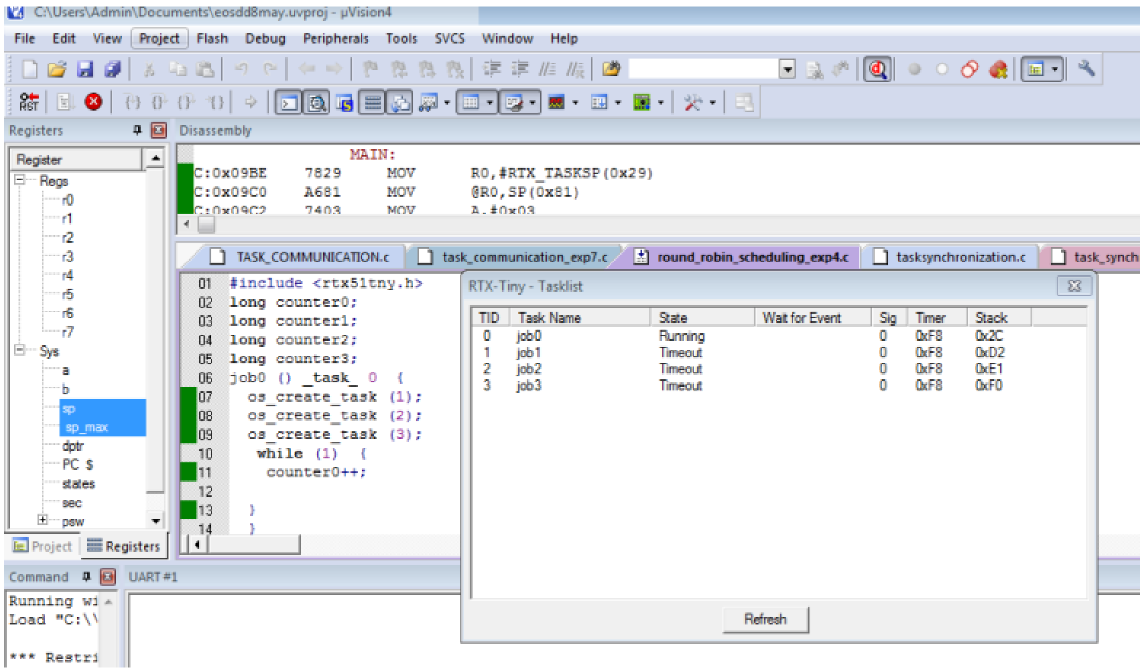
```
#include <reg51.h>
#include <rtx51tny.h>
#include <stdio.h>
extern char putchar(char c);
extern char getkey(void);
extern void serialinit(void);
void TASK0(void) _task_ 0
{
serialinit();
printf("CPU: 8xC51RD2, Memory Model: Large, Code ROM Size:
Large, XTAL:
11.0592 MHz\n");
printf("In Options for Target -> BL51 Misc -> Overlay -> * ! getkey,
printf ! *\n");
os_create_task(1);
os_create_task(2);
os_delete_task(0);
}
void task1(void) _task_ 1
{
while(1)
{
printf("Task 1 Running and Task 2 Ready.\n");
P0++;
}
}
void task2(void) _task_ 2
{
while(1)
{
printf("Task 2 Running and Task 1 Ready.\n");
P1++;
}
}
```

OUTPUT:



```
#include <rtx51tny.h>
long counter0;
long counter1;
long counter2;
long counter3;
job0 () _task_ 0 {
  os_create_task (1);
  os_create_task (2);
  os_create_task (3);
   while (1) {
     counter0++;

  }
 }
job1 () _task_ 1 {
  while (1) {
     counter1++;

  }
}
job2 () _task_ 2 {
  while (1) {
     counter2++;

        }
```

```
Rebuild target 'Target 1'
compiling round_robin_scheduling_exp4.c...
linking...
Program Size: data=43.1 xdata=0 const=0 code=656
"eosdd8may" - 0 Error(s), 0 Warning(s).
```



```
             MAIN:
C:0x09BE    7829     MOV      R0,#RTX_TASKSP(0x29)
C:0x09C0    A681     MOV      @R0,SP(0x81)
C:0x09C2    7403     MOV      A,#0x03
```

```
#include <rtx51tny.h>
long counter0;
long counter1;
long counter2;
long counter3;
job0 () _task_ 0 {
  os_create_task (1);
  os_create_task (2);
  os_create_task (3);
   while (1) {
     counter0++;

  }
 }
```

TID	Task Name	State	Wait for Event	Sig	Timer	Stack
0	job0	Running		0	0xF8	0x2C
1	job1	Timeout		0	0xF8	0xD2
2	job2	Timeout		0	0xF8	0xE1
3	job3	Timeout		0	0xF8	0xF0

```
Running wi
Load "C:\\

*** Restri
```

4. Processing Critical Section using RTX51inKeil

AIM: To demonstrate the use of RTX51 in Keil to process a critical section of code.

APPARATUS:

1. Keil software
2. RTX51 library

THEORY:

In embedded systems, a critical section is a region of code that requires exclusive access to a shared resource. The RTX51 operating system provides a mechanism for processing critical sections using the os_critical_section_enter() and os_critical_section_exit() functions.

When a task enters a critical section, it must first lock the critical section to prevent other tasks from entering it simultaneously. This is done by calling os_critical_section_enter(), which sets a flag indicating that the critical section is occupied. If another task attempts to enter the critical section while it is occupied, it will be blocked until the occupying task exits the critical section.

Once a task has entered a critical section, it can access the shared resource without fear of interference from other tasks. When the task has finished accessing the shared resource, it must exit the critical section by calling os_critical_section_exit(), which clears the flag and allows other tasks to enter the critical section.

The use of critical sections ensures that shared resources are accessed in a mutually exclusive manner, preventing data corruption and other concurrency-related issues. RTX51 provides a efficient and easy-to-use mechanism for processing critical sections, making it a popular choice for embedded systems developers.

In addition to critical sections, RTX51 also provides other synchronization mechanisms such as semaphores, mutexes, and events, which can be used to coordinate access to shared resources. By using these mechanisms, developers can write robust and reliable code that can handle the demands of real-time embedded systems.

PROGRAM:

```
#include <rtx51.h>

// Define a critical section
os_critical_section_t my_critical_section;

// Initialize the critical section
os_critical_section_init(&my_critical_section);
// Task 1
void task1(void) {
  // Enter the critical section
  os_critical_section_enter(&my_critical_section);
  // Critical section code
  // ...
  // Exit the critical section
  os_critical_section_exit(&my_critical_section);
}

// Task 2
void task2(void) {
  // Enter the critical section
  os_critical_section_enter(&my_critical_section);
  // Critical section code
  // ...
  // Exit the critical section
  os_critical_section_exit(&my_critical_section);
}
```

PROCEDURE:

1. Create a new project in Keil and include the RTX51 library.
2. Write the program code as shown above.
3. Compile and run the program.
4. Observe the behavior of the tasks and the critical section.

RESULT: The tasks will run concurrently, but the critical section will be processed exclusively by one task at a time. The output will depend on the specific implementation and the tasks being run.

5. Task Synchronization using RTX51 Semaphores in Keil

AIM: To demonstrate the use of semaphores in Keil for task synchronization.

APPARATUS:

1. Keil software
2. RTX51 library

THEORY:

Task synchronization is a crucial aspect of real-time operating systems, ensuring that tasks access shared resources without conflicts. RTX51 provides semaphores as a mechanism for task synchronization. A semaphore is a variable that controls access to a shared resource by multiple tasks.

In RTX51, semaphores are initialized using the os_semaphore_init() function, which sets the semaphore's initial value. The os_semaphore_wait() function is used to decrement the semaphore's value, blocking the task if the value is zero. The os_semaphore_signal() function increments the semaphore's value, waking up a blocked task if necessary.

When a task wants to access a shared resource, it first waits on the semaphore using os_semaphore_wait(). If the semaphore's value is zero, the task is blocked until another task signals the semaphore using os_semaphore_signal(). Once the task gains access to the shared resource, it executes its critical section code and then signals the semaphore to release the resource.

RTX51 semaphores ensure that only one task can access a shared resource at a time, preventing data corruption and other concurrency issues. By using semaphores, developers can write robust and reliable code for real-time embedded systems.

In addition to semaphores, RTX51 provides other synchronization mechanisms like mutexes and events, which can be used to coordinate access to shared resources. By leveraging these mechanisms, developers can create efficient and reliable real-time systems.

In summary, RTX51 semaphores provide a powerful mechanism for task synchronization, ensuring exclusive access to shared resources and preventing concurrency issues. By understanding and utilizing semaphores effectively, developers can create robust and reliable real-time embedded systems.

PROGRAM:

```
#include <rtx51.h>

// Create a semaphore

os_semaphore_t my_semaphore;

// Initialize the semaphore

os_semaphore_init(&my_semaphore, 1);

// Task 1

void task1(void) {

  // Wait on the semaphore

  os_semaphore_wait(&my_semaphore);

  // Critical section

  // ...

  // Signal the semaphore

  os_semaphore_signal(&my_semaphore);

}

// Task 2

void task2(void) {

  // Wait on the semaphore

  os_semaphore_wait(&my_semaphore);
```

```
          // Critical section
          // ...
          // Signal the semaphore
          os_semaphore_signal(&my_semaphore);
        }
        // Delete the semaphore
        os_semaphore_delete(&my_semaphore);
```

PROCEDURE:

1. Create a new project in Keil and include the RTX51 library.
2. Write the program code as shown above.
3. Compile and run the program.
4. Observe the behavior of the tasks and the semaphore.

RESULT: The tasks will run concurrently, but the semaphore will ensure that only one task can access the shared resource at a time. The output will depend on the specific implementation and the tasks being run.

6. Task Communication using shared memory in Keil

AIM: To demonstrate the use of shared memory for task communication in Keil.

APPARATUS:

1. Keil software
2. RTX51 library

THEORY:

Task communication is a vital aspect of real-time operating systems, enabling tasks to exchange data and coordinate actions. Shared memory is a mechanism that allows tasks to communicate by accessing a common memory region. In RTX51, shared memory is implemented using the os_shared_memory_create() function, which creates a shared memory block.

Once created, tasks can access the shared memory block using the os_shared_memory_attach() function, which maps the shared memory block to the task's address space. Tasks can then read and write data to the shared memory block using standard memory access operations.

Shared memory communication is fast and efficient, as tasks do not need to use operating system services to exchange data. However, it requires careful synchronization to prevent data corruption and ensure consistency.

RTX51 provides synchronization mechanisms like semaphores and mutexes to coordinate access to shared memory. Tasks must use these mechanisms to ensure exclusive access to the shared memory region, preventing simultaneous writes and ensuring data integrity.

Shared memory communication is suitable for tasks that need to exchange large amounts of data or require low-latency communication. By using shared memory, developers can create efficient and reliable real-time systems.

PROGRAM:

```
#include <rtx51.h>

// Define a shared memory variable

os_shared_memory_t my_shared_memory;

// Initialize the shared memory

os_shared_memory_init(&my_shared_memory, sizeof(int));

// Task 1

void task1(void) {

  int data = 10;

  // Write to shared memory

  os_shared_memory_write(&my_shared_memory, &data, sizeof(int));

}

// Task 2

void task2(void) {

  int data;

  // Read from shared memory

  os_shared_memory_read(&my_shared_memory, &data, sizeof(int));

  // Use the data

  printf("Received data: %d\n", data);

}
```

PROCEDURE:

1. Create a new project in Keil and include the RTX51 library.
2. Write the program code as shown above.
3. Compile and run the program.

4. Observe the behavior of the tasks and the shared memory.

RESULT: The tasks will communicate with each other using the shared memory, and the output will depend on the specific implementation and the tasks being run.

7. Task Communication using RTX51 Mailbox in Keil

AIM: To demonstrate the use of RTX51 mailbox for task communication in Keil.

APPARATUS:

1. Keil software
2. RTX51 library

THEORY:

Task communication is a crucial aspect of real-time operating systems, enabling tasks to exchange data and coordinate actions. RTX51 provides a mailbox mechanism for task communication, allowing tasks to send and receive messages. A mailbox is a buffer that stores messages, and tasks can access it using the os_mailbox_create() function.

To send a message, a task uses the os_mailbox_post() function, which copies the message to the mailbox. The message is then stored in the mailbox until a task receives it using the os_mailbox_pend() function. If a task attempts to send a message to a full mailbox, it will be blocked until space becomes available.

Similarly, if a task attempts to receive a message from an empty mailbox, it will be blocked until a message is available. RTX51 provides synchronization mechanisms like semaphores and mutexes to coordinate access to mailboxes, ensuring exclusive access and preventing data corruption.

Mailboxes are suitable for tasks that need to exchange small amounts of data or require asynchronous communication. By using mailboxes, developers can create efficient and reliable real-time systems that meet the demands of embedded applications.

In addition to mailboxes, RTX51 provides other communication mechanisms like shared memory and message queues, which can be used to exchange data between tasks. By leveraging these mechanisms, developers can create robust and reliable real-time systems that meet the demands of embedded applications.

The use of mailboxes in RTX51 provides a flexible and efficient way for tasks to communicate, enabling real-time systems to respond to events and coordinate actions. By understanding and utilizing mailboxes effectively, developers can create high-performance and reliable real-time systems.

PROGRAM:

```
#include <rtx51.h>

// Define a mailbox

os_mailbox_t my_mailbox;

// Initialize the mailbox

os_mailbox_init(&my_mailbox, sizeof(int));

// Task 1

void task1(void) {

  int data = 10;

  // Send a message to the mailbox

  os_mailbox_send(&my_mailbox, &data, sizeof(int));

}

// Task 2

void task2(void) {

  int data;

  // Receive a message from the mailbox

  os_mailbox_receive(&my_mailbox, &data, sizeof(int));

  // Use the data

  printf("Received data: %d\n", data);
```

```
        }
```

PROCEDURE:

1. Create a new project in Keil and include the RTX51 library.
2. Write the program code as shown above.
3. Compile and run the program.
4. Observe the behavior of the tasks and the mailbox.

RESULT: The tasks will communicate with each other using the mailbox, and the output will depend on the specific implementation and the tasks being run.

8. Introduction to ARM Cortex M3 Processor

AIM: To provide an overview of the ARM Cortex M3 processor and its features.

APPARATUS:

1. ARM Cortex M3 processor
2. Keil software

THEORY:

The ARM Cortex M3 processor is a 32-bit microprocessor that is widely used in embedded systems and microcontrollers. It is based on the ARMv7-M architecture and is designed for high performance, low power consumption, and small size.

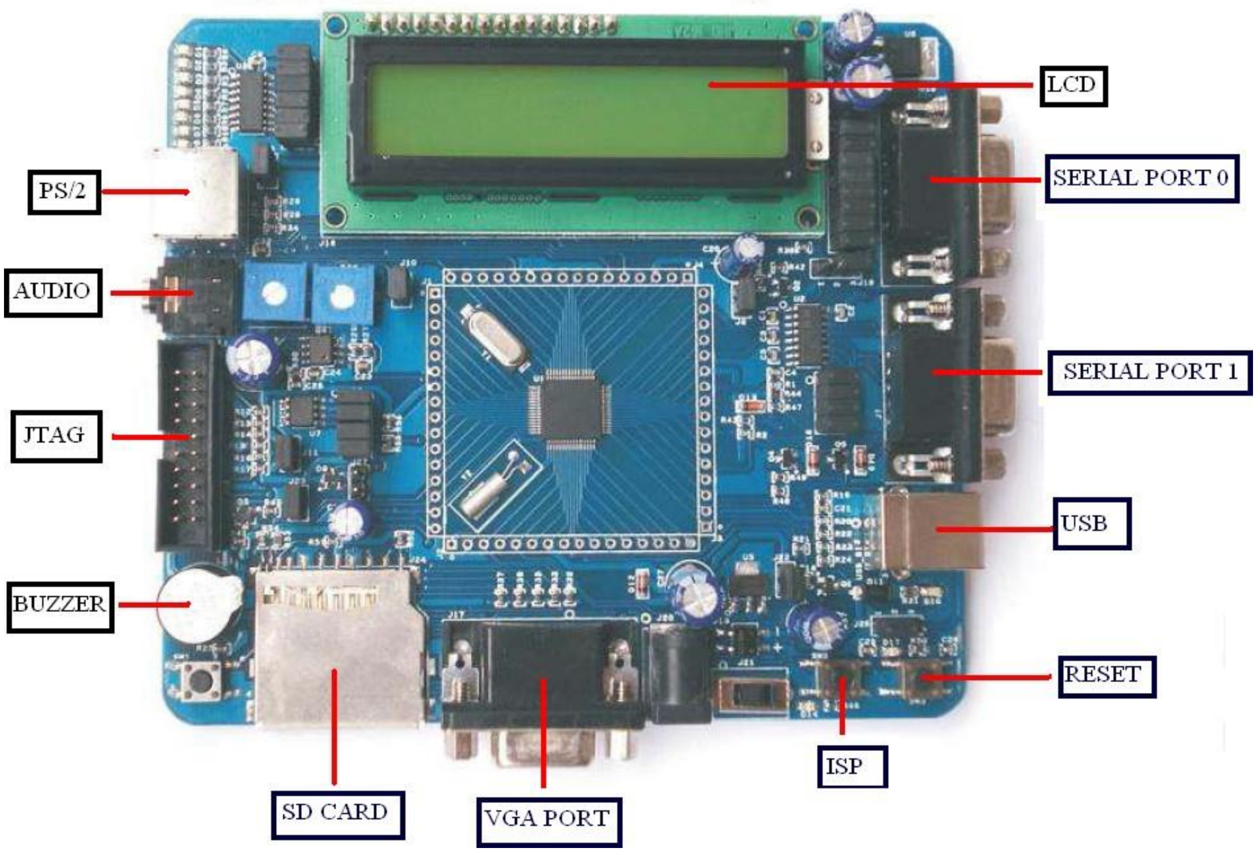
One of the key features of the ARM Cortex M3 processor is its pipelined architecture, which allows it to execute instructions in a series of stages. This improves performance by allowing the processor to execute multiple instructions simultaneously. Additionally, the processor uses the Thumb instruction set, which is a compressed version of the ARM instruction set. This reduces code size and improves performance.

The processor also has a number of built-in features, including a Nested Vectored Interrupt Controller (NVIC) and a System Tick Timer (SysTick). The NVIC allows for efficient handling of interrupts, while the SysTick provides a regular tick interrupt. Furthermore, the processor has a memory protection unit (MPU), which allows for memory access control and protection.

The ARM Cortex M3 processor is widely used in a variety of applications, including microcontrollers, embedded systems, automotive systems, industrial control systems, and consumer electronics. Its high performance, low power consumption, and small size make it a popular choice for many applications.

However, the processor also has some limitations. For example, it has a limited instruction set and a limited address space. Despite these limitations, the ARM Cortex M3 processor is a powerful and efficient processor that is widely used in many different applications.

LPC2148 (ARM) MICROCONTROLLER:



Procedure:

1. Introduction to the ARM Cortex M3 processor architecture:

   The ARM Cortex M3 processor is a 32-bit microprocessor that is based on the ARMv7-M architecture. It is designed for embedded systems and microcontrollers. The processor core has the following features:

   a. 32-bit instruction set
   b. 32-bit data bus
   c. 16-bit Thumb instruction set (for efficient code density)
   d. Harvard bus architecture (separate data and instruction buses)
   e. Pipelined architecture (for improved performance)

2. Overview of the processor's features, including:

   The ARM Cortex M3 processor has several features that make it suitable for embedded systems:

   a. Nested Vectored Interrupt Controller (NVIC): The NVIC is a built-in interrupt controller that allows for efficient handling of interrupts. It supports up to 240 interrupts and has a flexible priority scheme.
   b. System Tick Timer (SysTick): The SysTick is a built-in timer that provides a regular tick interrupt. It can be used for tasks such as scheduling, timing, and synchronization.
   c. Memory protection: The processor has a memory protection unit (MPU) that allows for memory access control and protection.
   d. Thumb instruction set: The Thumb instruction set is a compressed version of the ARM instruction set. It provides efficient code density and improved performance.

3. Discussion of the processor's applications, including:
   The ARM Cortex M3 processor is widely used in various applications, including:
      a. Microcontrollers: The processor is used in many microcontrollers, such as the STM32 and LPC175x series.
      b. Embedded systems: The processor is used in various embedded systems, such as industrial control systems, automotive systems, and consumer electronics.
      c. Automotive systems: The processor is used in various automotive systems, such as infotainment systems, navigation systems, and safety systems.
      d. Industrial control systems: The processor is used in various industrial control systems, such as motor control systems, power supply systems, and automation systems.

Result: Understanding of the ARM Cortex M3 processor and its features, as well as its applications in various fields.

9. MULTPLICATION OF TWO 16 BIT NUMBERS USING ARM CORTEX-3

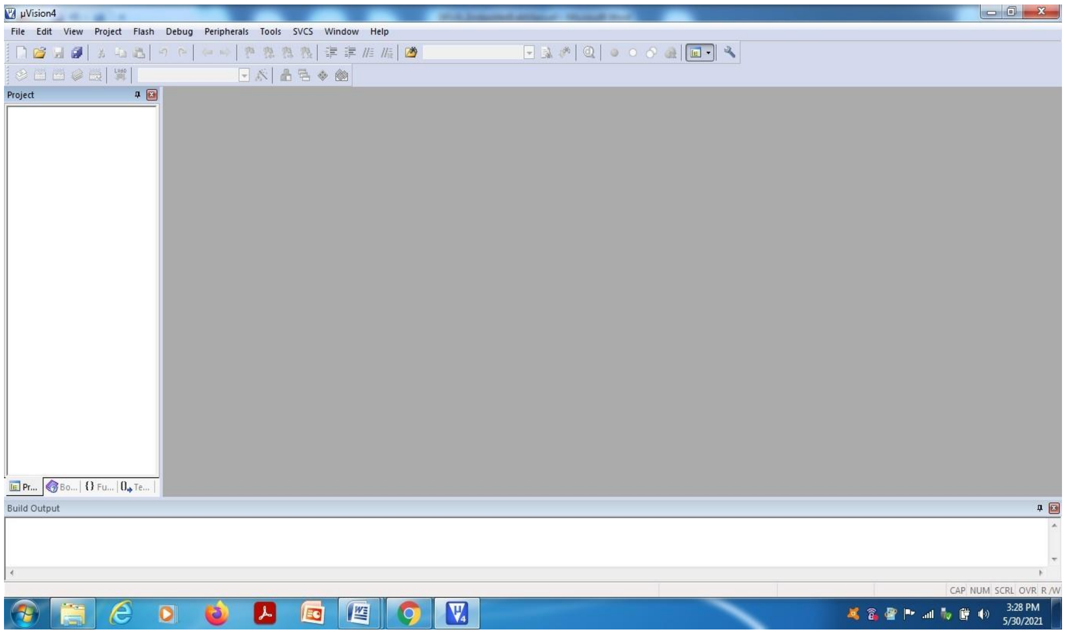
AIM: To study and verify the multiplication of 2 16-bit numbers using ARM cortex-3 ALP program.

APPARATUS: 1. PC with Windows 10, 64-bit OS.

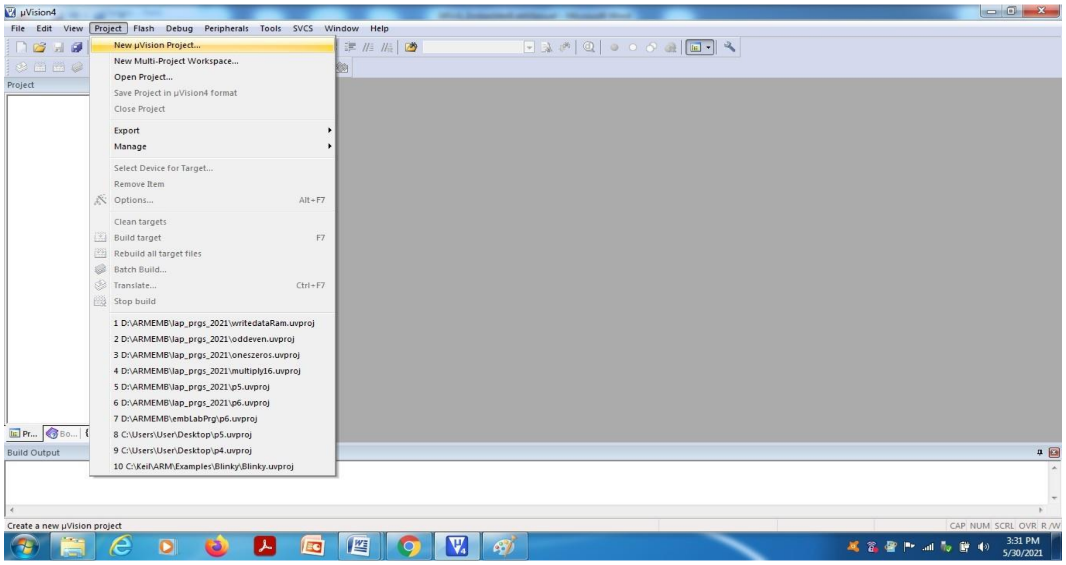
2. Keil µVision4 Software.

PROCEDURE:

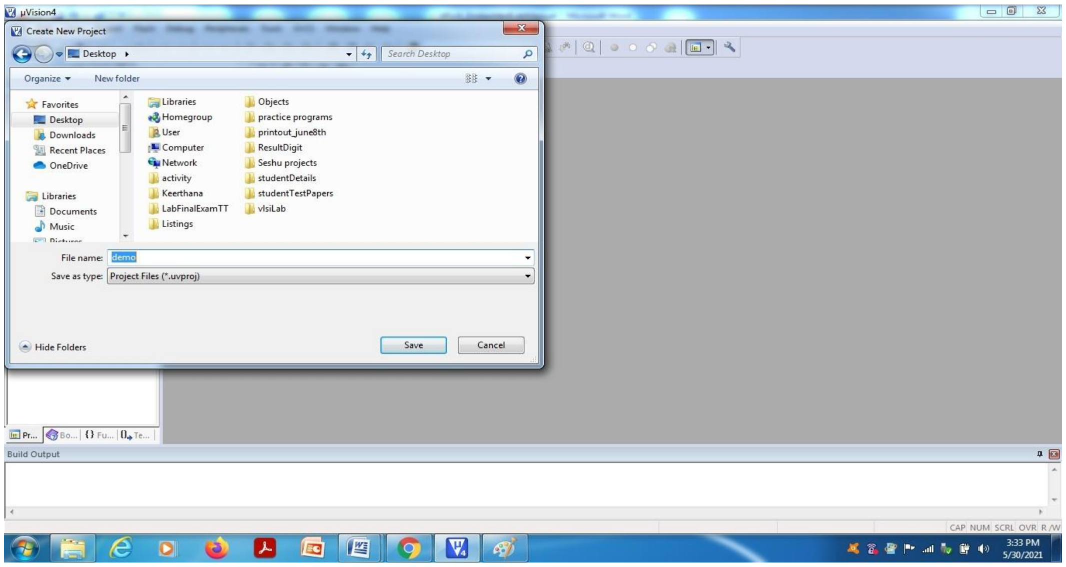
1. Double click on µvision 4 icon in the desktop.


µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Project
Build Output

2. Select "New µvision Project" from project in the menubar.


µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
New µVision Project...
New Multi-Project Workspace...
Open Project...
Save Project in µVision4 format
Close Project
Export
Manage
Select Device for Target...
Remove Item
Options... Alt+F7
Clean targets
Build target F7
Rebuild all target files
Batch Build...
Translate... Ctrl+F7
Stop build
1 D:\ARMEMB\lap_prgs_2021\writedataRam.uvproj
2 D:\ARMEMB\lap_prgs_2021\oddeven.uvproj
3 D:\ARMEMB\lap_prgs_2021\oneszeros.uvproj
4 D:\ARMEMB\lap_prgs_2021\multiply16.uvproj
5 D:\ARMEMB\lap_prgs_2021\p5.uvproj
6 D:\ARMEMB\lap_prgs_2021\p6.uvproj
7 D:\ARMEMB\embLabPrg\p6.uvproj
8 C:\Users\User\Desktop\p5.uvproj
9 C:\Users\User\Desktop\p4.uvproj
10 C:\Keil\ARM\Examples\Blinky\Blinky.uvproj
Project
Build Output
Create a new µVision project

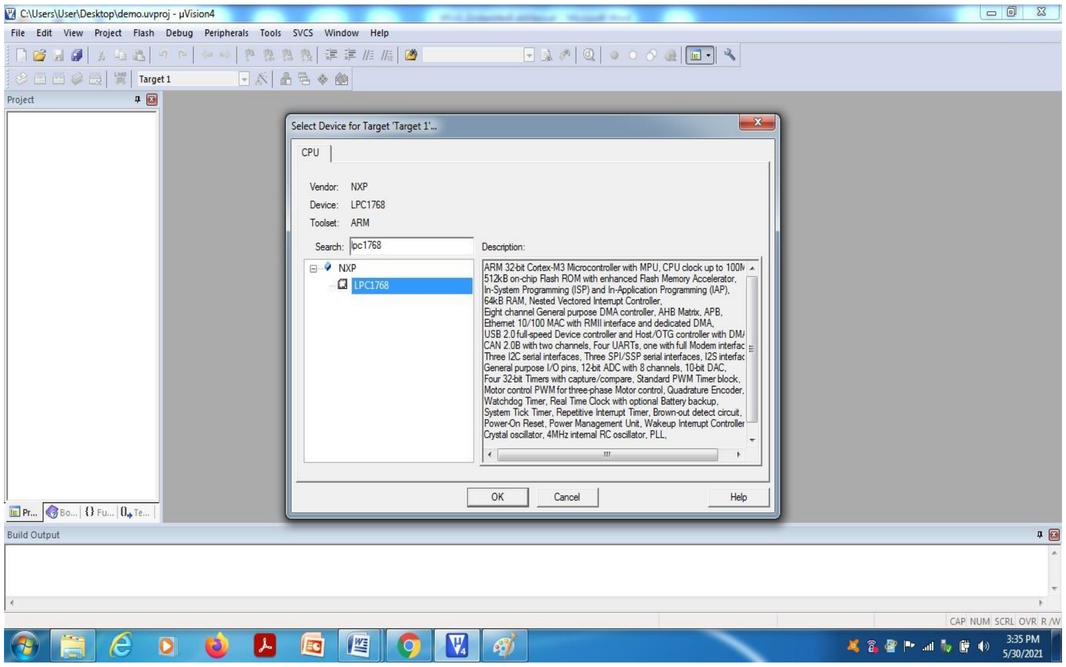
3. Browse and create a new project in the required location.



File name: demo

Save as type: Project Files (*.uvproj)

4. Select the target device (here, LPC1768 from NXP) from the list or type the exact name of the device. Press OK.

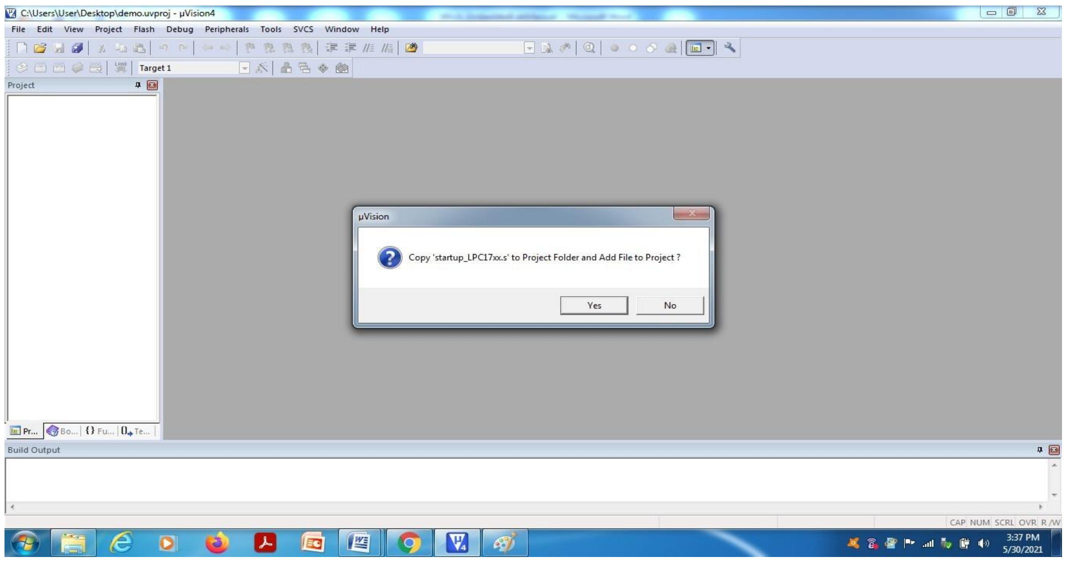


Vendor: NXP
Device: LPC1768
Toolset: ARM
Search: lpc1768

Description:

ARM 32-bit Cortex-M3 Microcontroller with MPU, CPU clock up to 100M
512kB on-chip Flash ROM with enhanced Flash Memory Accelerator,
In-System Programming (ISP) and In-Application Programming (IAP),
64kB RAM, Nested Vectored Interrupt Controller,
Eight channel General purpose DMA controller, AHB Matrix, APB,
Ethernet 10/100 MAC with RMII interface and dedicated DMA,
USB 2.0 full-speed Device controller and Host/OTG controller with DM
CAN 2.0B with two channels, Four UARTs, one with full Modem interfac
Three I2C serial interfaces, Three SPI/SSP serial interfaces, I2S interfac
General purpose I/O pins, 12-bit ADC with 8 channels, 10-bit DAC,
Four 32-bit Timers with capture/compare, Standard PWM Timer block,
Motor control PWM for three-phase Motor control, Quadrature Encoder,
Watchdog Timer, Real Time Clock with optional Battery backup,
System Tick Timer, Repetitive Interrupt Timer, Brown-out detect circuit,
Power-On Reset, Power Management Unit, Wakeup Interrupt Controller
Crystal oscillator, 4MHz internal RC oscillator, PLL,

5. Copy start up to Project folder and add to project file"?- Press NO.

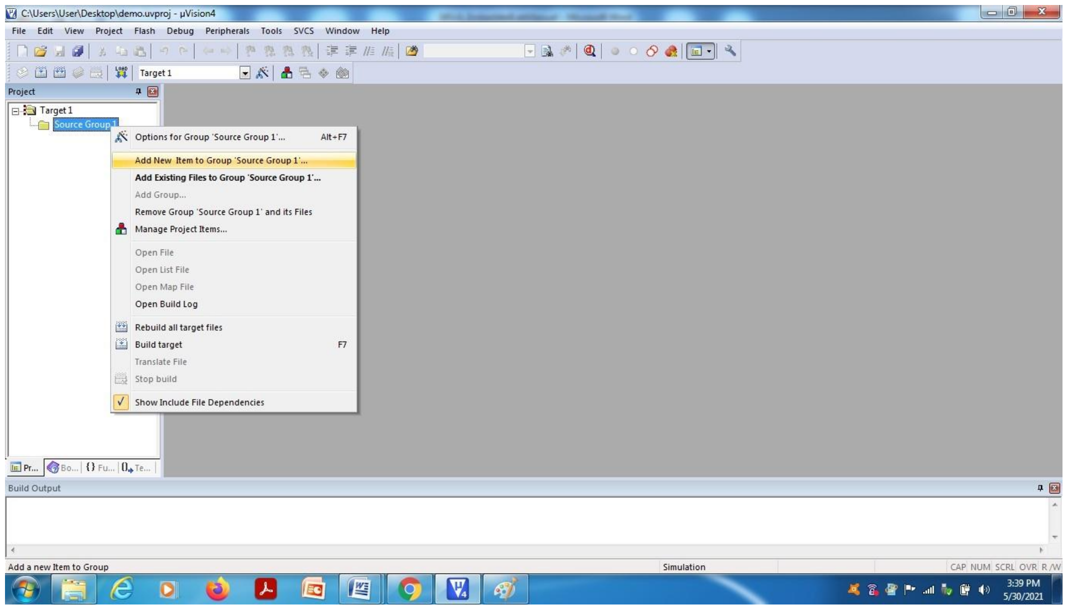

C:\Users\User\Desktop\demo.uvproj - µVision4

µVision

Copy 'startup_LPC17xx.s' to Project Folder and Add File to Project ?

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Yes
No
Build Output
CAP NUM SCRL OVR R/W
3:37 PM
5/30/2021

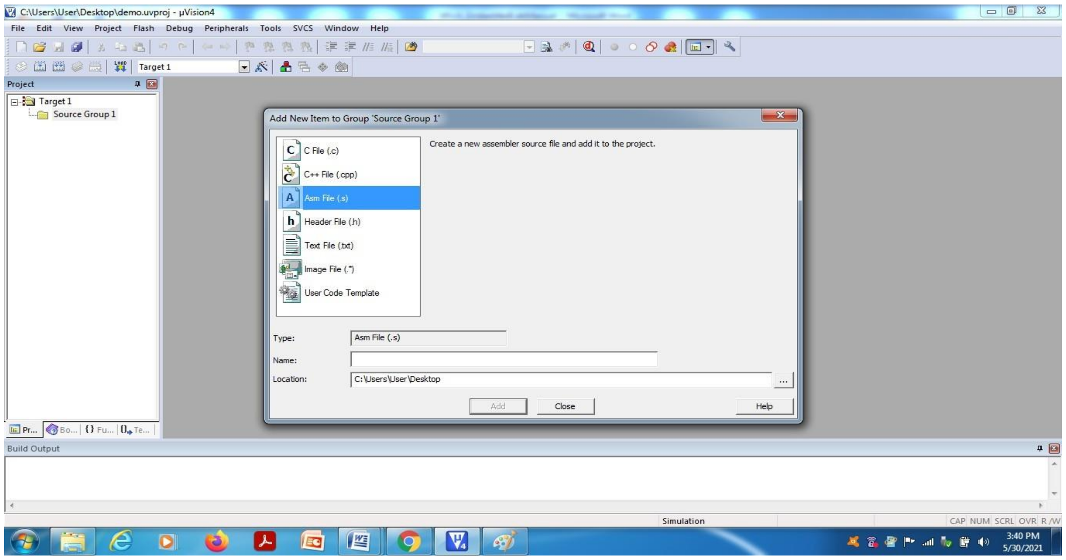
6. In the project window, right click on source and select Add new item to group "source group1".


C:\Users\User\Desktop\demo.uvproj - µVision4

Add a new Item to Group

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Target 1
Source Group 1
Options for Group 'Source Group 1'... Alt+F7
Add New Item to Group 'Source Group 1'...
Add Existing Files to Group 'Source Group 1'...
Add Group...
Remove Group 'Source Group 1' and its Files
Manage Project Items...
Open File
Open List File
Open Map File
Open Build Log
Rebuild all target files
Build target F7
Translate File
Stop build
Show Include File Dependencies
Build Output
Simulation
CAP NUM SCRL OVR R/W
3:39 PM
5/30/2021

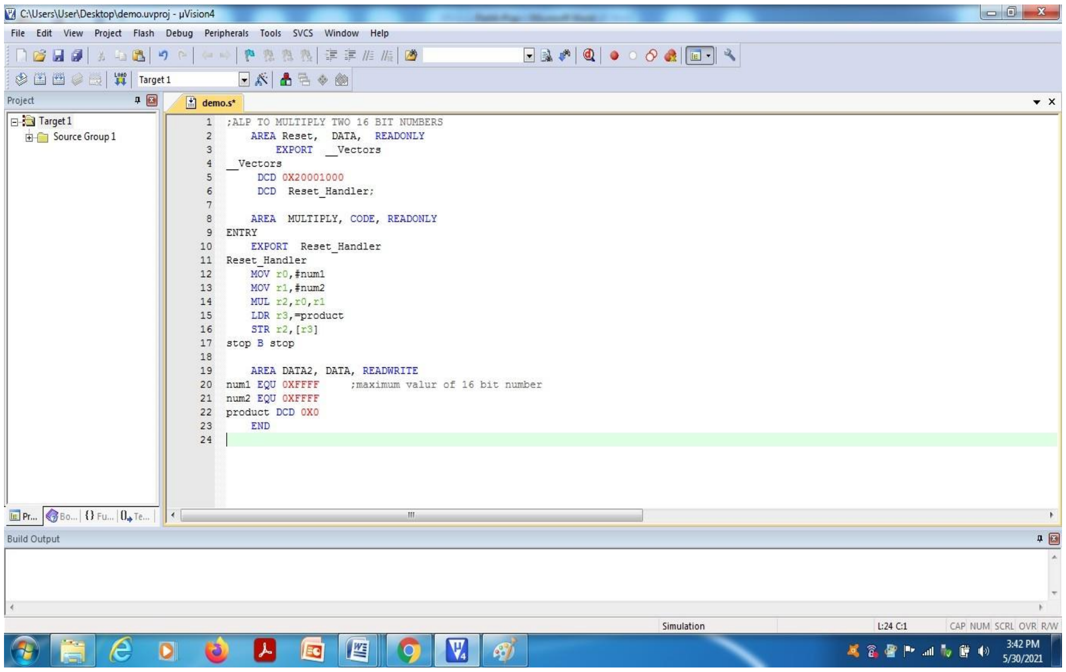
7. Select ASM file and give name of the file with .s extension and press ADD.



Create a new assembler source file and add it to the project.

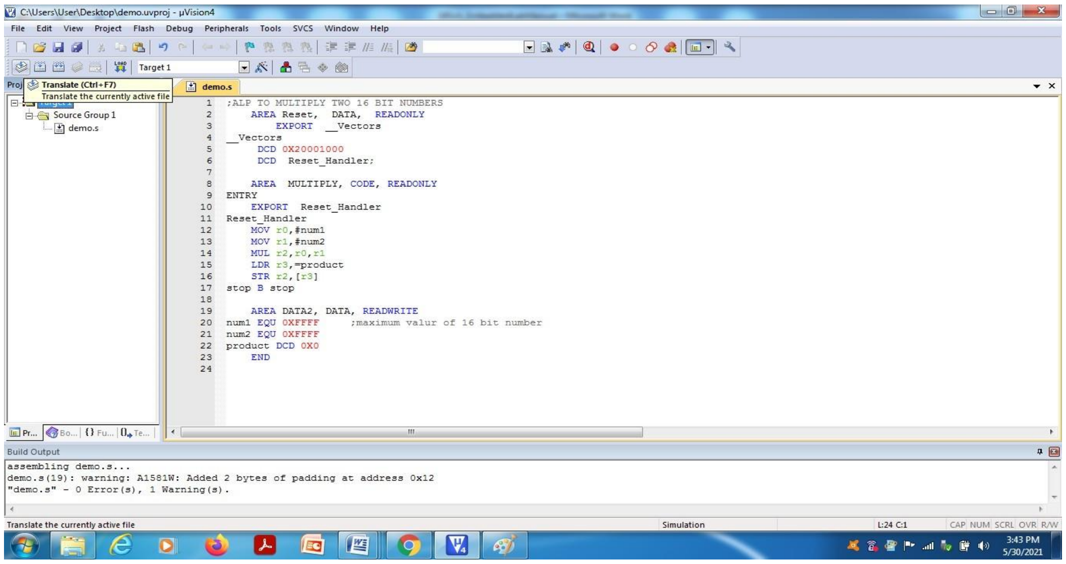
Type: Asm File (.s)
Name:
Location: C:\Users\User\Desktop

8. Type the program in the editor space and save


C:\Users\User\Desktop\demo.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Target 1
Source Group 1
demo.s*
Build Output
Simulation
L:24 C:1

```
;ALP TO MULTIPLY TWO 16 BIT NUMBERS
    AREA Reset,  DATA,  READONLY
        EXPORT  __Vectors
__Vectors
     DCD 0X20001000
     DCD  Reset_Handler;

    AREA  MULTIPLY, CODE, READONLY
ENTRY
    EXPORT  Reset_Handler
Reset_Handler
    MOV r0,#num1
    MOV r1,#num2
    MUL r2,r0,r1
    LDR r3,=product
    STR r2,[r3]
stop B stop

    AREA DATA2, DATA, READWRITE
num1 EQU 0XFFFF     ;maximum valur of 16 bit number
num2 EQU 0XFFFF
product DCD 0X0
    END
```

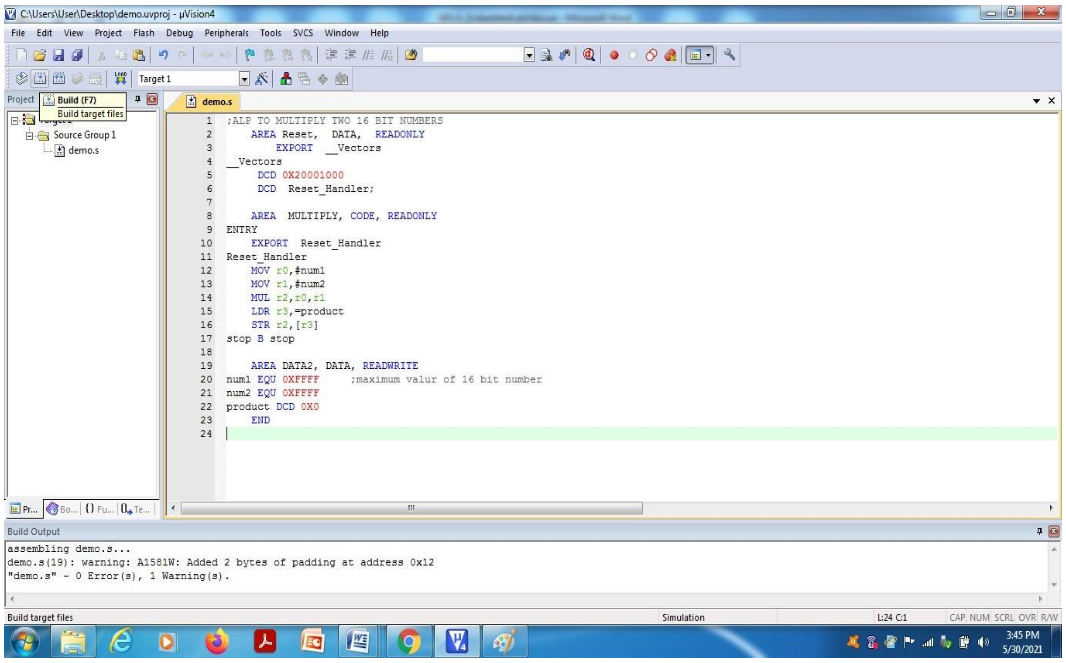
9. Translate the program by select the icon from tool bar or from menu bar, also Check for errors and warnings in the bottom window.


C:\Users\User\Desktop\demo.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Translate (Ctrl+F7)
Translate the currently active file
Source Group 1
demo.s
Build Output
Simulation
L:24 C:1

```
;ALP TO MULTIPLY TWO 16 BIT NUMBERS
    AREA Reset,  DATA,  READONLY
        EXPORT  __Vectors
__Vectors
     DCD 0X20001000
     DCD  Reset_Handler;

    AREA  MULTIPLY, CODE, READONLY
ENTRY
    EXPORT  Reset_Handler
Reset_Handler
    MOV r0,#num1
    MOV r1,#num2
    MUL r2,r0,r1
    LDR r3,=product
    STR r2,[r3]
stop B stop

    AREA DATA2, DATA, READWRITE
num1 EQU 0XFFFF     ;maximum valur of 16 bit number
num2 EQU 0XFFFF
product DCD 0X0
    END
```

```
assembling demo.s...
demo.s(19): warning: A1581W: Added 2 bytes of padding at address 0x12
"demo.s" - 0 Error(s), 1 Warning(s).
```

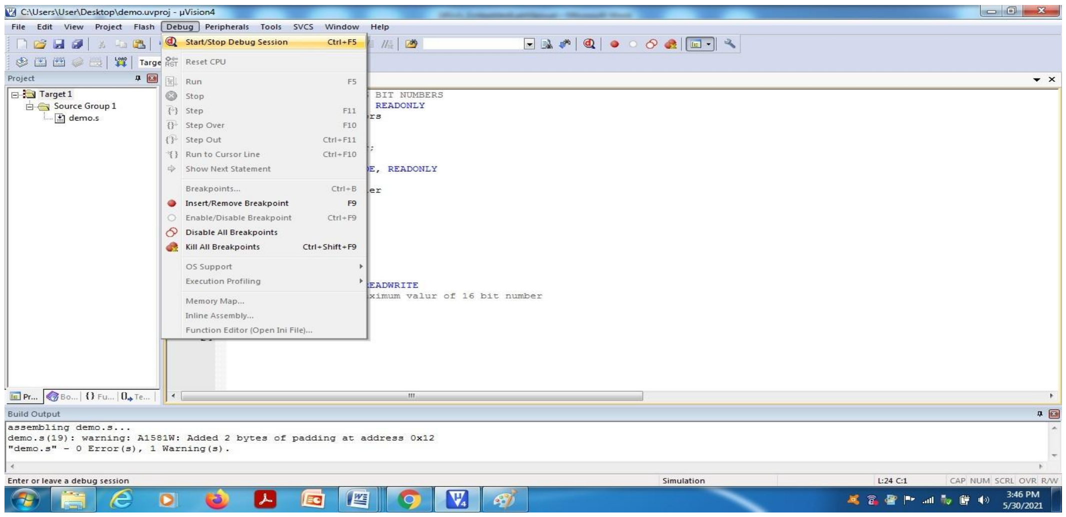
10. If no error, Select "Build" icon from tool bar or from menu bar.


C:\Users\User\Desktop\demo.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Build (F7)
Build target files
Source Group 1
demo.s
Build Output
Simulation
L:24 C:1

```
;ALP TO MULTIPLY TWO 16 BIT NUMBERS
    AREA Reset,  DATA,  READONLY
        EXPORT  __Vectors
__Vectors
     DCD 0X20001000
     DCD  Reset_Handler;

    AREA  MULTIPLY, CODE, READONLY
ENTRY
    EXPORT  Reset_Handler
Reset_Handler
    MOV r0,#num1
    MOV r1,#num2
    MUL r2,r0,r1
    LDR r3,=product
    STR r2,[r3]
stop B stop

    AREA DATA2, DATA, READWRITE
num1 EQU 0XFFFF     ;maximum valur of 16 bit number
num2 EQU 0XFFFF
product DCD 0X0
    END
```

```
assembling demo.s...
demo.s(19): warning: A1581W: Added 2 bytes of padding at address 0x12
"demo.s" - 0 Error(s), 1 Warning(s).
```

11. Start the debug session from Menu bar.


C:\Users\User\Desktop\demo.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Start/Stop Debug Session Ctrl+F5
Reset CPU
Run F5
Stop
Step F11
Step Over F10
Step Out Ctrl+F11
Run to Cursor Line Ctrl+F10
Show Next Statement
Breakpoints... Ctrl+B
Insert/Remove Breakpoint F9
Enable/Disable Breakpoint Ctrl+F9
Disable All Breakpoints
Kill All Breakpoints Ctrl+Shift+F9
OS Support
Execution Profiling
Memory Map...
Inline Assembly...
Function Editor (Open Ini File)...
Project
Target 1
Source Group 1
demo.s
Build Output
Enter or leave a debug session
Simulation
L:24 C:1
3:46 PM
5/30/2021

```
assembling demo.s...
demo.s(19): warning: A1581W: Added 2 bytes of padding at address 0x12
"demo.s" - 0 Error(s), 1 Warning(s).
```

12. Press OK

13. Press function key F11 or select "step" option under Debug menu for single step execution and verify the outputin register window/Memorywindow/xPSR.

PROGRAMME:

ALP TO MULTIPLY TWO 16-BIT NUMBERS

```
AREA Reset, DATA, READONLY

EXPORT_Vectors

_Vectors

DCD 0X20001000

DCD Reset_Handler;

AREA MULTIPLY, CODE, READONLY ENTRY

EXPORT Reset_Handler

Reset_Handler

MOV r0,#num1

MOV r1,#num2

MUL r2,r0,r1
```

LDR r3,=product

STR r2,[r3]

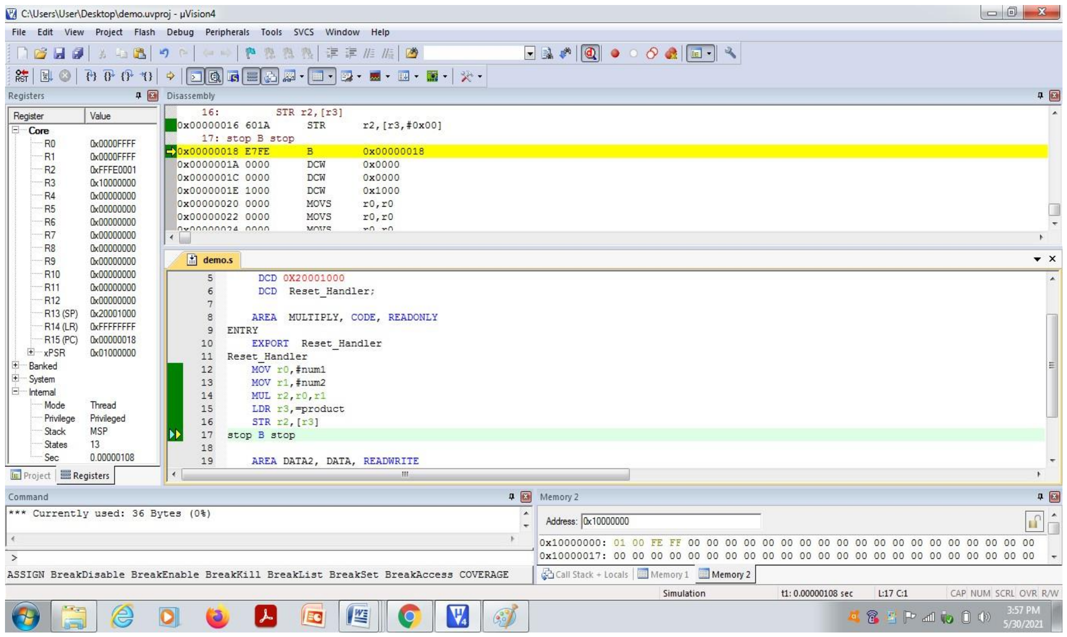
stop B stop

AREA DATA2, DATA, READWRITE

num1 EQU 0XFFFF;maximum value of 16 bit number num2 EQU 0XFFFFj

product DCD 0X0 END

OUTPUT:



C:\Users\User\Desktop\demo.uvproj - µVision4

Registers

Register	Value
Core	
R0	0x0000FFFF
R1	0x0000FFFF
R2	0xFFFE0001
R3	0x10000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20001000
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x00000018
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	13
Sec	0.00000108

Disassembly

```
    16:         STR r2,[r3]
0x00000016 601A      STR      r2,[r3,#0x00]
    17: stop B stop
0x00000018 E7FE      B        0x00000018
0x0000001A 0000      DCW      0x0000
0x0000001C 0000      DCW      0x0000
0x0000001E 1000      DCW      0x1000
0x00000020 0000      MOVS     r0,r0
0x00000022 0000      MOVS     r0,r0
```

demo.s

```
     DCD 0X20001000
     DCD  Reset_Handler;

    AREA  MULTIPLY, CODE, READONLY
ENTRY
    EXPORT  Reset_Handler
Reset_Handler
    MOV r0,#num1
    MOV r1,#num2
    MUL r2,r0,r1
    LDR r3,=product
    STR r2,[r3]
stop B stop

    AREA DATA2, DATA, READWRITE
```

Command

```
*** Currently used: 36 Bytes (0%)
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE
```

Memory 2

Address: 0x10000000

```
0x10000000: 01 00 FE FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x10000017: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Simulation t1: 0.00000108 sec L:17 C:1

RESULT:(0xFFFF) x(0xFFFF) =0xFFFE0001 in the product memory location.

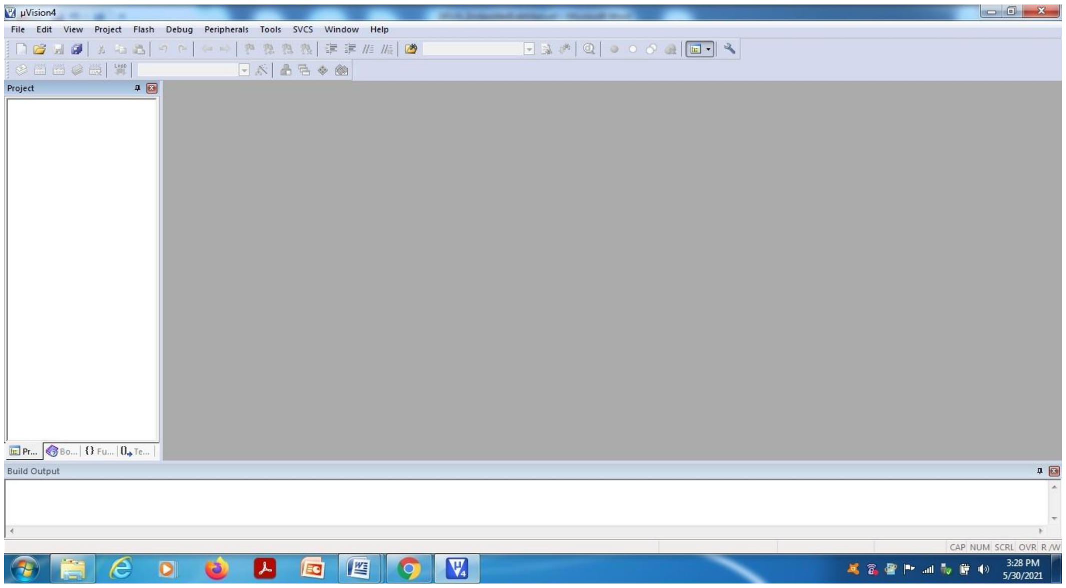
10. SUM OF FIRST TEN INTEGER NUMBERS USING ARM CORTEX-3

AIM: To find sum of first ten integer numbers using ARM cortex-3 ALP program.

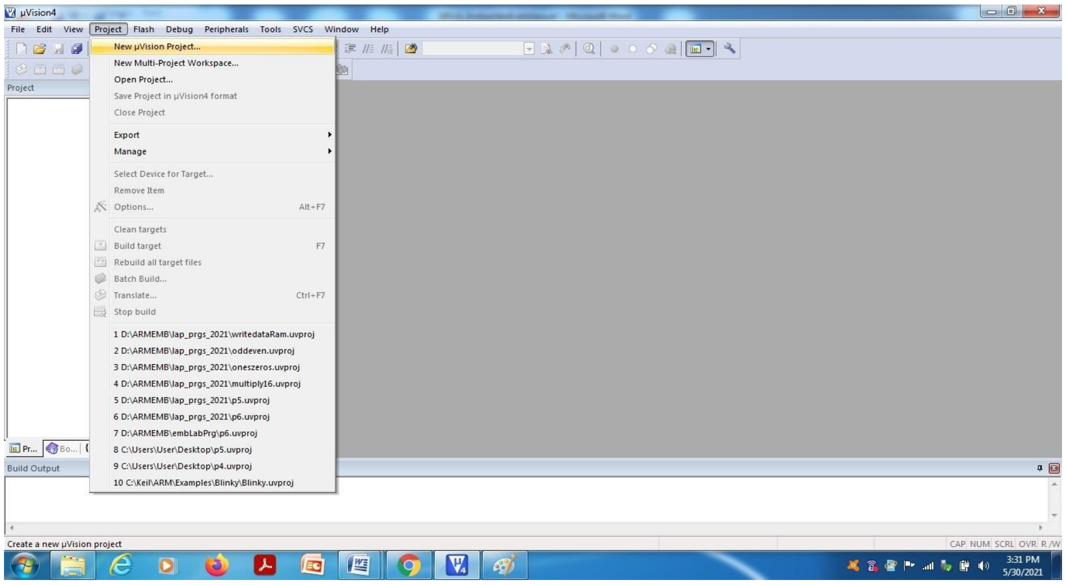
APPARATUS: 1. PC with Windows 10, 64-bit OS.
2. Keil µVision4 Software.

PROCEDURE:

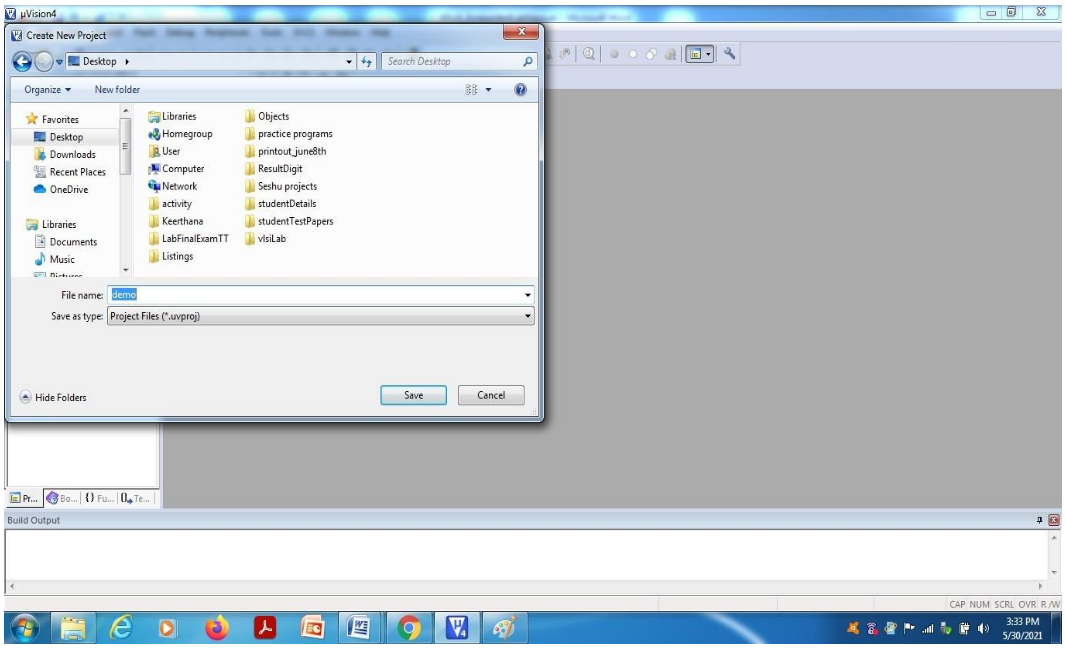
1. Double click on µvision 4 icon in the desktop.


µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Project
Build Output
CAP NUM SCRL OVR R/W
3:28 PM
5/30/2021

2. Select "New µvision Project" from project in the menu bar.


µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
New µVision Project...
New Multi-Project Workspace...
Open Project...
Save Project in µVision4 format
Close Project
Export
Manage
Select Device for Target...
Remove Item
Options... Alt+F7
Clean targets
Build target F7
Rebuild all target files
Batch Build...
Translate... Ctrl+F7
Stop build
1 D:\ARMEMB\lap_prgs_2021\writedataRam.uvproj
2 D:\ARMEMB\lap_prgs_2021\oddeven.uvproj
3 D:\ARMEMB\lap_prgs_2021\oneszeros.uvproj
4 D:\ARMEMB\lap_prgs_2021\multiply16.uvproj
5 D:\ARMEMB\lap_prgs_2021\p5.uvproj
6 D:\ARMEMB\lap_prgs_2021\p6.uvproj
7 D:\ARMEMB\embLabPrg\p6.uvproj
8 C:\Users\User\Desktop\p5.uvproj
9 C:\Users\User\Desktop\p4.uvproj
10 C:\Keil\ARM\Examples\Blinky\Blinky.uvproj
Project
Build Output
Create a new µVision project
CAP NUM SCRL OVR R/W
3:31 PM
5/30/2021

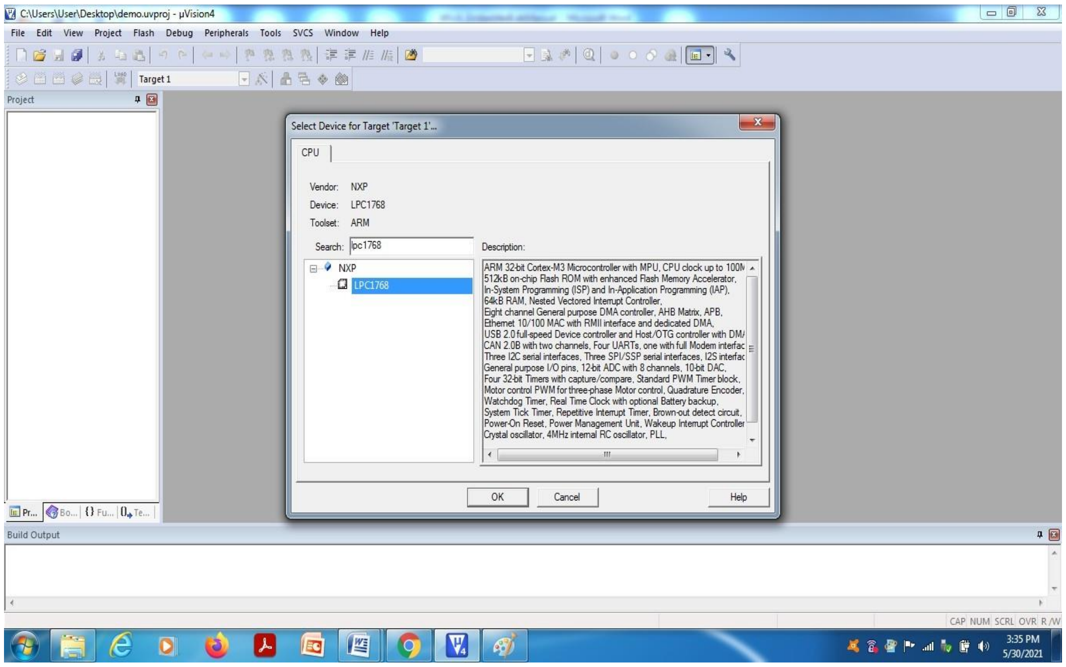
3. Browse and create a new project in the required location.



File name: demo

Save as type: Project Files (*.uvproj)

4. Select the target device (here, LPC1768 from NXP) from the list or type the exact name of the device. Press OK.

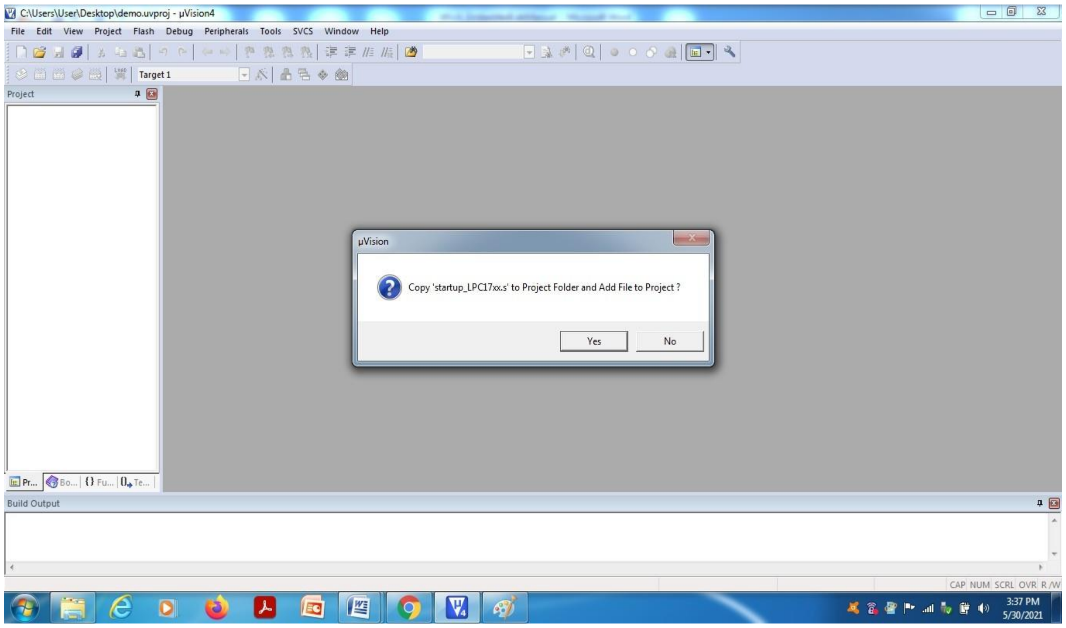


Vendor: NXP
Device: LPC1768
Toolset: ARM
Search: lpc1768

Description:

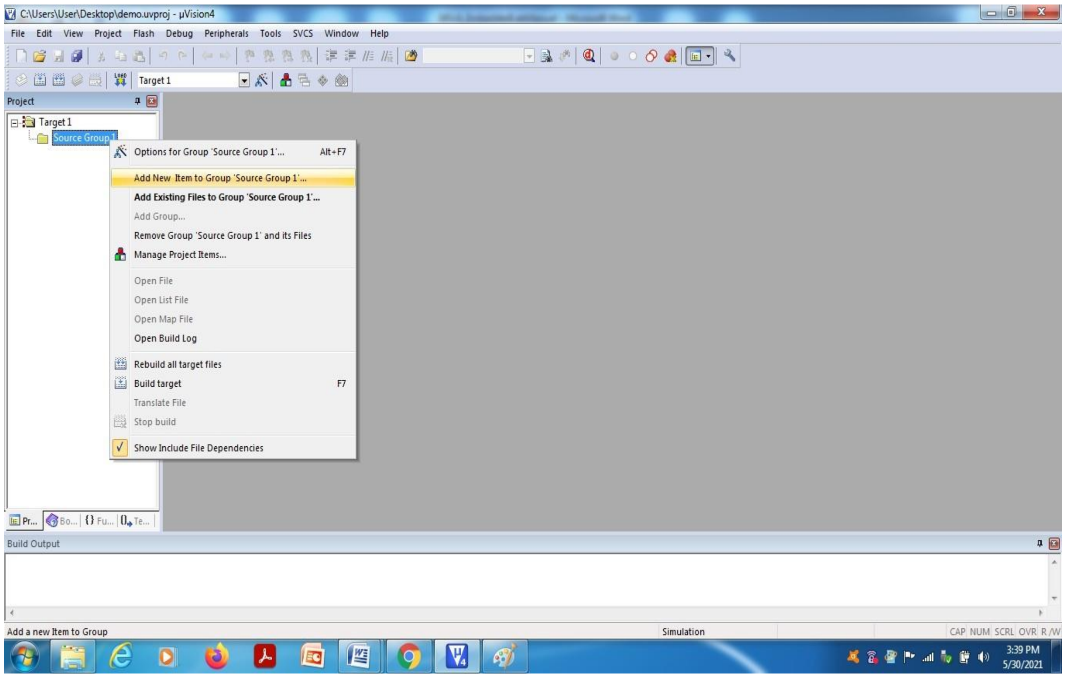
ARM 32-bit Cortex-M3 Microcontroller with MPU, CPU clock up to 100M
512kB on-chip Flash ROM with enhanced Flash Memory Accelerator,
In-System Programming (ISP) and In-Application Programming (IAP),
64kB RAM, Nested Vectored Interrupt Controller,
Eight channel General purpose DMA controller, AHB Matrix, APB,
Ethernet 10/100 MAC with RMII interface and dedicated DMA,
USB 2.0 full-speed Device controller and Host/OTG controller with DM
CAN 2.0B with two channels, Four UARTs, one with full Modem interfac
Three I2C serial interfaces, Three SPI/SSP serial interfaces, I2S interfac
General purpose I/O pins, 12-bit ADC with 8 channels, 10-bit DAC,
Four 32-bit Timers with capture/compare, Standard PWM Timer block,
Motor control PWM for three-phase Motor control, Quadrature Encoder,
Watchdog Timer, Real Time Clock with optional Battery backup,
System Tick Timer, Repetitive Interrupt Timer, Brown-out detect circuit,
Power-On Reset, Power Management Unit, Wakeup Interrupt Controller
Crystal oscillator, 4MHz internal RC oscillator, PLL,

5. Copy start up to Project folder and add to project file"?- Press NO.

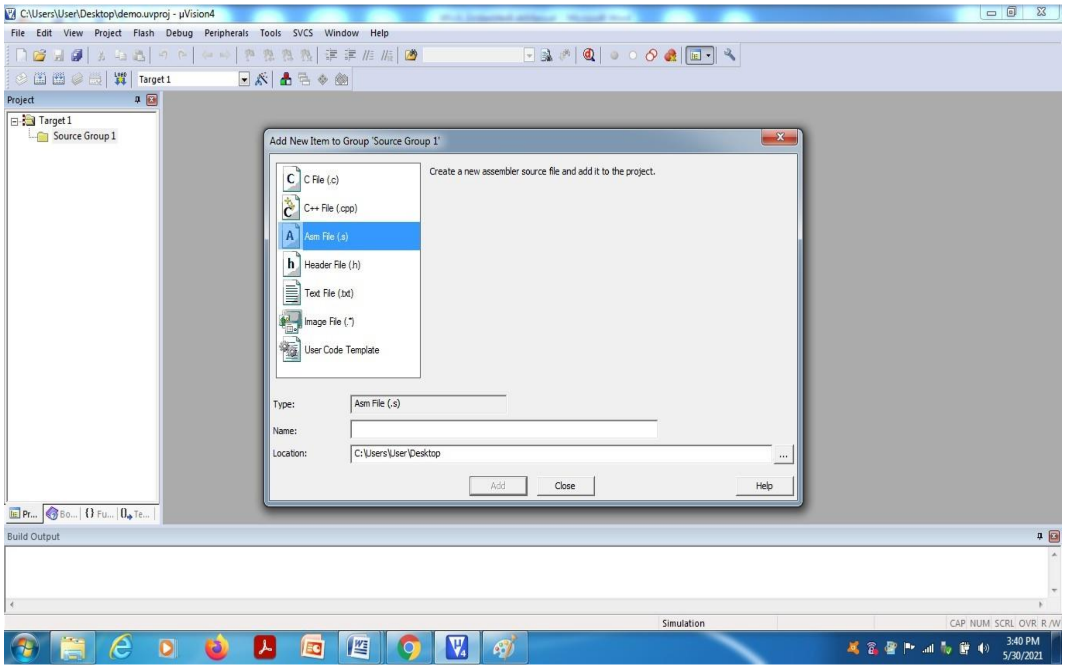

C:\Users\User\Desktop\demo.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Build Output
µVision
Yes
No

Copy 'startup_LPC17xx.s' to Project Folder and Add File to Project ?

6. In the project window, right click on source and select Add new item to group "source group1".


C:\Users\User\Desktop\demo.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Target 1
Source Group 1
Options for Group 'Source Group 1'... Alt+F7
Add New Item to Group 'Source Group 1'...
Add Existing Files to Group 'Source Group 1'...
Add Group...
Remove Group 'Source Group 1' and its Files
Manage Project Items...
Open File
Open List File
Open Map File
Open Build Log
Rebuild all target files
Build target F7
Translate File
Stop build
Show Include File Dependencies
Build Output
Add a new Item to Group
Simulation

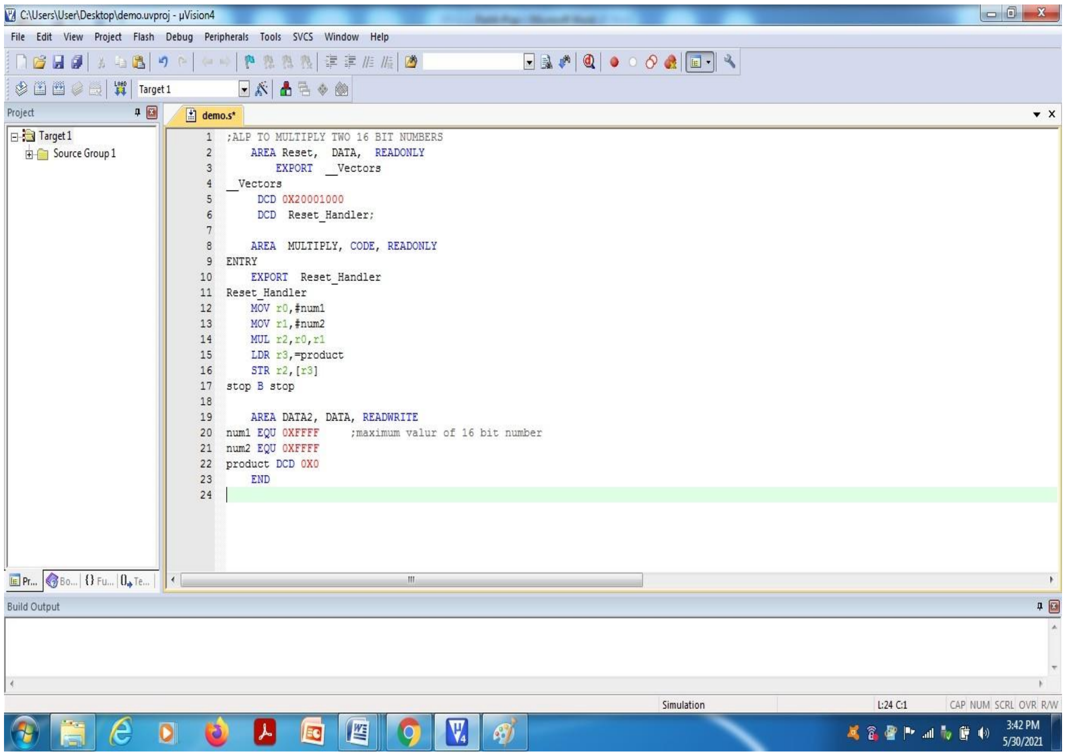
7. Select ASM file and give name of the file with .s extension and press ADD.



Create a new assembler source file and add it to the project.

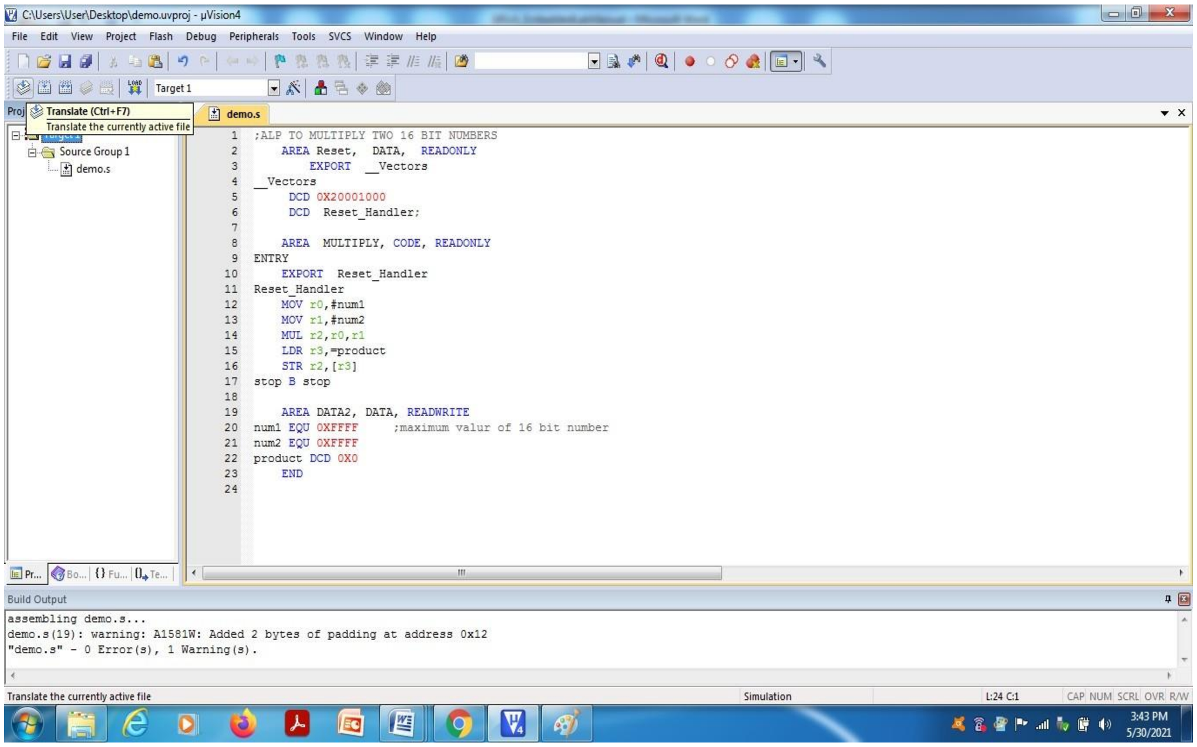
Type: Asm File (.s)
Name:
Location: C:\Users\User\Desktop

8. Type the program in the editor space and save.


C:\Users\User\Desktop\demo.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Target 1
Source Group 1
demo.s*
Build Output
Simulation
L:24 C:1

```
;ALP TO MULTIPLY TWO 16 BIT NUMBERS
    AREA Reset,  DATA,  READONLY
        EXPORT  __Vectors
__Vectors
     DCD 0X20001000
     DCD  Reset_Handler;

    AREA  MULTIPLY, CODE, READONLY
ENTRY
    EXPORT  Reset_Handler
Reset_Handler
    MOV r0,#num1
    MOV r1,#num2
    MUL r2,r0,r1
    LDR r3,=product
    STR r2,[r3]
stop B stop

    AREA DATA2, DATA, READWRITE
num1 EQU 0XFFFF     ;maximum valur of 16 bit number
num2 EQU 0XFFFF
product DCD 0X0
    END
```

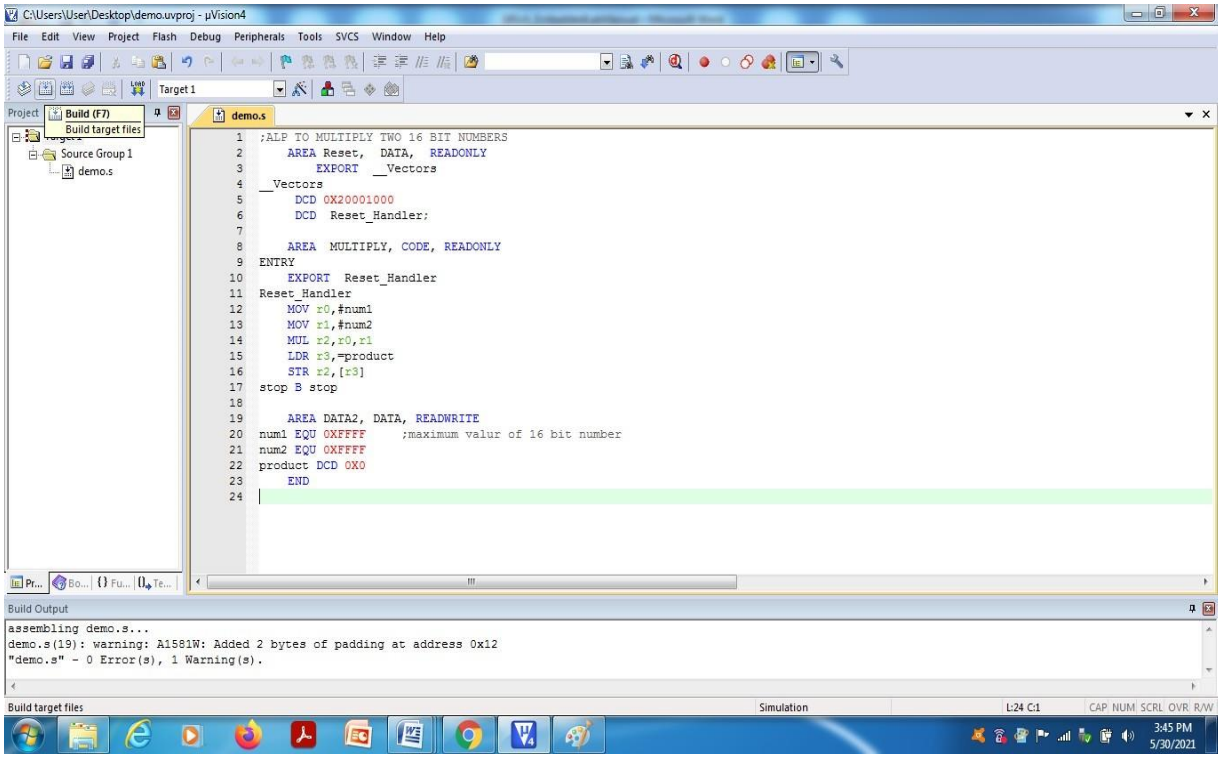
9. Translate the program by select the icon from tool bar or from menubar, also Check for errors and warnings in the bottom window.


C:\Users\User\Desktop\demo.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Translate (Ctrl+F7)
Translate the currently active file
Source Group 1
demo.s
Build Output
Translate the currently active file
Simulation
L:24 C:1
CAP NUM SCRL OVR R/W
3:43 PM
5/30/2021

```
;ALP TO MULTIPLY TWO 16 BIT NUMBERS
    AREA Reset,  DATA,  READONLY
        EXPORT  __Vectors
__Vectors
     DCD 0X20001000
     DCD  Reset_Handler;

    AREA  MULTIPLY, CODE, READONLY
ENTRY
    EXPORT  Reset_Handler
Reset_Handler
    MOV r0,#num1
    MOV r1,#num2
    MUL r2,r0,r1
    LDR r3,=product
    STR r2,[r3]
stop B stop

    AREA DATA2, DATA, READWRITE
num1 EQU 0XFFFF     ;maximum valur of 16 bit number
num2 EQU 0XFFFF
product DCD 0X0
    END
```

```
assembling demo.s...
demo.s(19): warning: A1581W: Added 2 bytes of padding at address 0x12
"demo.s" - 0 Error(s), 1 Warning(s).
```

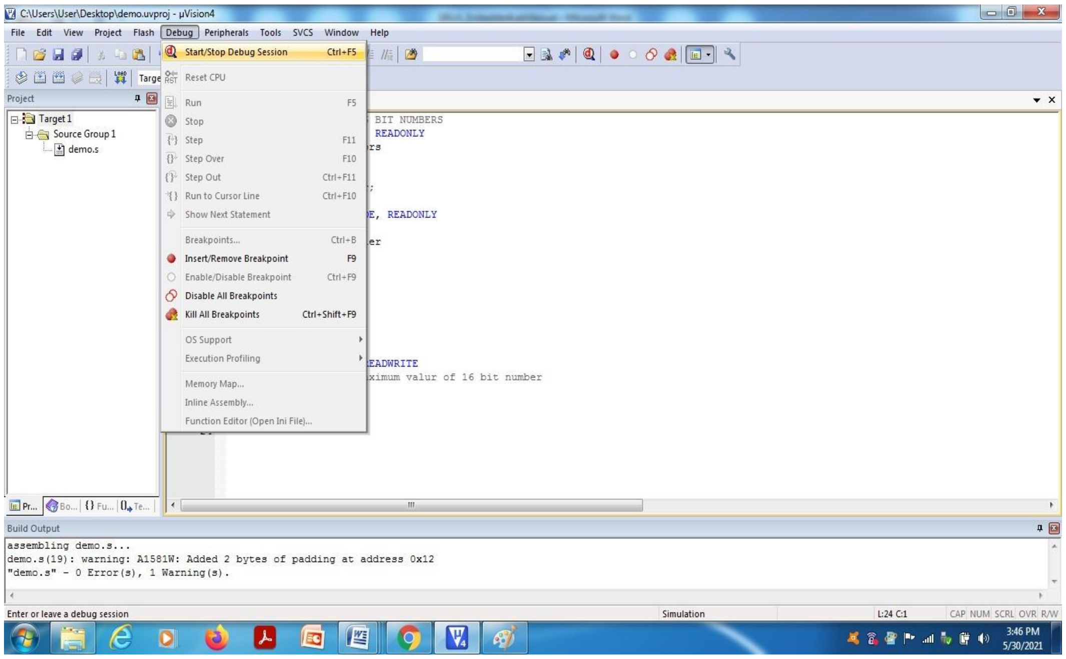
10. If no error, Select "Build" icon from tool bar or from menubar.



```
;ALP TO MULTIPLY TWO 16 BIT NUMBERS
    AREA Reset,  DATA,  READONLY
        EXPORT  __Vectors
__Vectors
     DCD 0X20001000
     DCD  Reset_Handler;

    AREA  MULTIPLY, CODE, READONLY
ENTRY
    EXPORT  Reset_Handler
Reset_Handler
    MOV r0,#num1
    MOV r1,#num2
    MUL r2,r0,r1
    LDR r3,=product
    STR r2,[r3]
stop B stop

    AREA DATA2, DATA, READWRITE
num1 EQU 0XFFFF     ;maximum valur of 16 bit number
num2 EQU 0XFFFF
product DCD 0X0
    END
```

```
assembling demo.s...
demo.s(19): warning: A1581W: Added 2 bytes of padding at address 0x12
"demo.s" - 0 Error(s), 1 Warning(s).
```

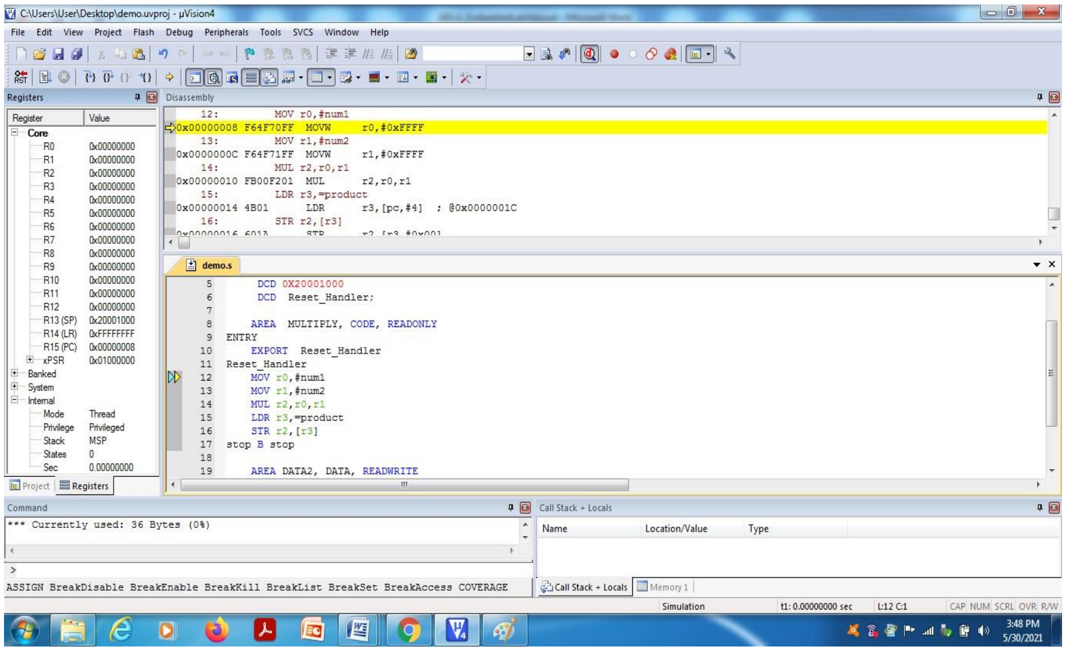
11. Start the debug session from Menubar.


C:\Users\User\Desktop\demo.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Start/Stop Debug Session Ctrl+F5
Reset CPU
Run F5
Stop
Step F11
Step Over F10
Step Out Ctrl+F11
Run to Cursor Line Ctrl+F10
Show Next Statement
Breakpoints... Ctrl+B
Insert/Remove Breakpoint F9
Enable/Disable Breakpoint Ctrl+F9
Disable All Breakpoints
Kill All Breakpoints Ctrl+Shift+F9
OS Support
Execution Profiling
Memory Map...
Inline Assembly...
Function Editor (Open Ini File)...
Project
Target 1
Source Group 1
demo.s
Enter or leave a debug session
Simulation
L:24 C:1
3:46 PM
5/30/2021

Build Output

```
assembling demo.s...
demo.s(19): warning: A1581W: Added 2 bytes of padding at address 0x12
"demo.s" - 0 Error(s), 1 Warning(s).
```

12. PressOK



Register	Value
Core	
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20001000
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x00000008
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	0
Sec	0.00000000

```
    12:         MOV r0,#num1
0x00000008 F64F70FF  MOVW      r0,#0xFFFF
    13:         MOV r1,#num2
0x0000000C F64F71FF  MOVW      r1,#0xFFFF
    14:         MUL r2,r0,r1
0x00000010 FB00F201  MUL       r2,r0,r1
    15:         LDR r3,=product
0x00000014 4B01      LDR       r3,[pc,#4]  ; @0x0000001C
    16:         STR r2,[r3]
```

```
     DCD 0X20001000
     DCD  Reset_Handler;

    AREA  MULTIPLY, CODE, READONLY
ENTRY
    EXPORT  Reset_Handler
Reset_Handler
    MOV r0,#num1
    MOV r1,#num2
    MUL r2,r0,r1
    LDR r3,=product
    STR r2,[r3]
stop B stop

    AREA DATA2, DATA, READWRITE
```

```
*** Currently used: 36 Bytes (0%)
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE
```

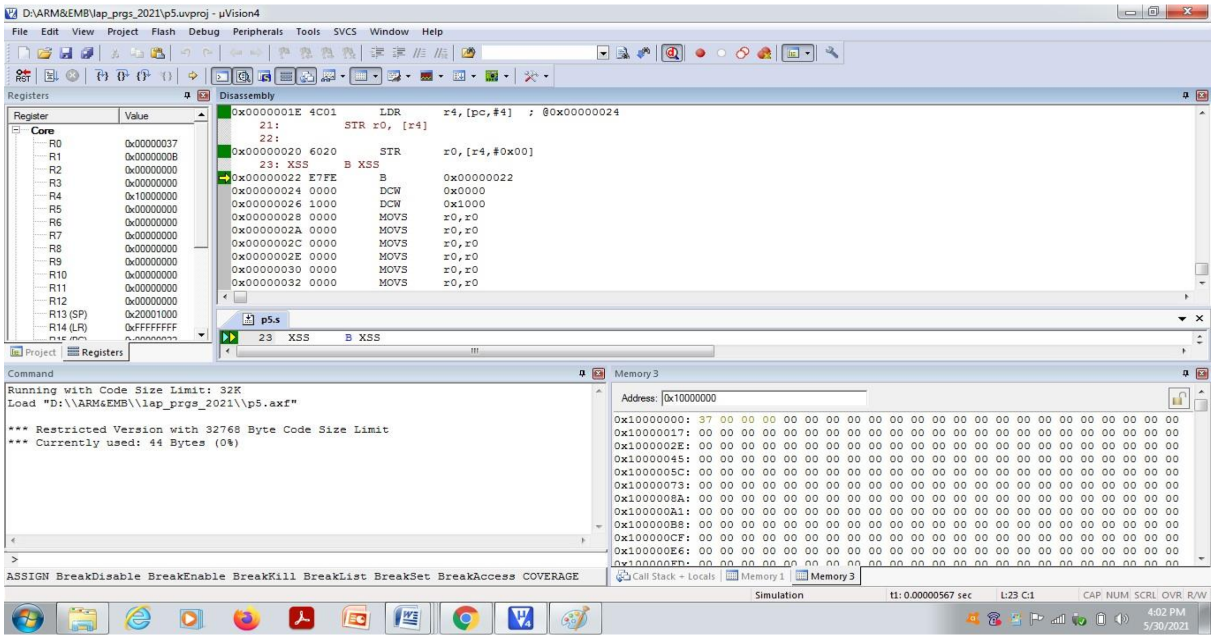
13. Press function key F11 or select “step” option under Debug menu for single step execution and verify the outputin register window/Memorywindow/xPSR.

PROGRAMME:

ALP TO FIND THE SUM OF FIRST 10INTEGERS

```
AREA Reset, DATA,READONLY
EXPORTVectors
Vectors
DCD 0X20001000
DCDReset_Handler;

AREA SUM, CODE, READONLY ENTRY
EXPORT Reset_HandlerReset_Handler
MOV r3,#10 MOV r0,#0 MOV r1,#1
l1            ADD r0,r0,r1 ADD r1,r1,#1 SUBS r3,#1 BNEl1
LDR r4, =RESULT STR r0, [r4]

XSS   BXSS

AREA DATA2, DATA, READWRITE

RESULT DCD 0X0
END   ;Mark theend
```

Result:
1+2+3+……+10=55d=37H. (At RESULT Memory Location)



Register	Value
Core	
R0	0x00000037
R1	0x0000000B
R2	0x00000000
R3	0x00000000
R4	0x10000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20001000
R14 (LR)	0xFFFFFFFF

```
0x0000001E 4C01      LDR      r4,[pc,#4]  ; @0x00000024
    21:          STR r0, [r4]
    22:
0x00000020 6020      STR      r0,[r4,#0x00]
    23: XSS      B XSS
0x00000022 E7FE      B        0x00000022
0x00000024 0000      DCW      0x0000
0x00000026 1000      DCW      0x1000
0x00000028 0000      MOVS     r0,r0
0x0000002A 0000      MOVS     r0,r0
0x0000002C 0000      MOVS     r0,r0
0x0000002E 0000      MOVS     r0,r0
0x00000030 0000      MOVS     r0,r0
0x00000032 0000      MOVS     r0,r0
```

p5.s

```
23  XSS      B XSS
```

```
Running with Code Size Limit: 32K
Load "D:\\ARM&EMB\\lap_prgs_2021\\p5.axf"

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 44 Bytes (0%)

>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE
```

Address: 0x10000000

```
0x10000000: 37 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x10000017: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x1000002E: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x10000045: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x1000005C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x10000073: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x1000008A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x100000A1: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x100000B8: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x100000CF: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x100000E6: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

11. TO FIND THE 1'S AND 0' IN THE GIVEN 32-BIT DATA USING ARM CORTEX-3

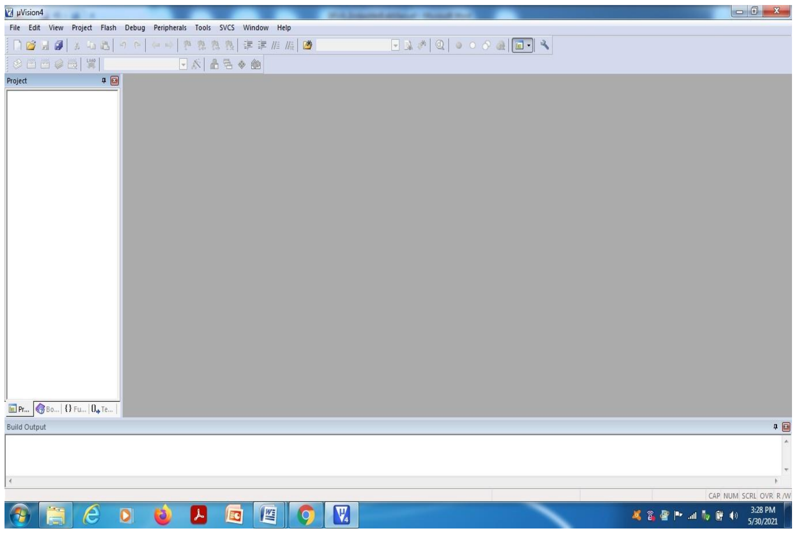
AIM: To find the 1's and 0's in the given 32-bit data using ARM cortex-3 ALP program.

APPARATUS: 1. PC with Windows 10, 64-bit OS.

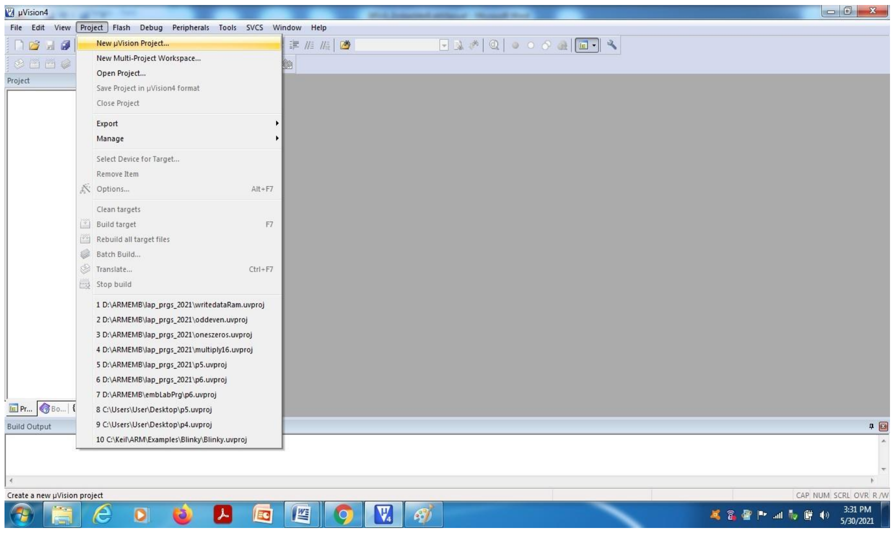
2. Keil µVision4 Software.

PROCEDURE:

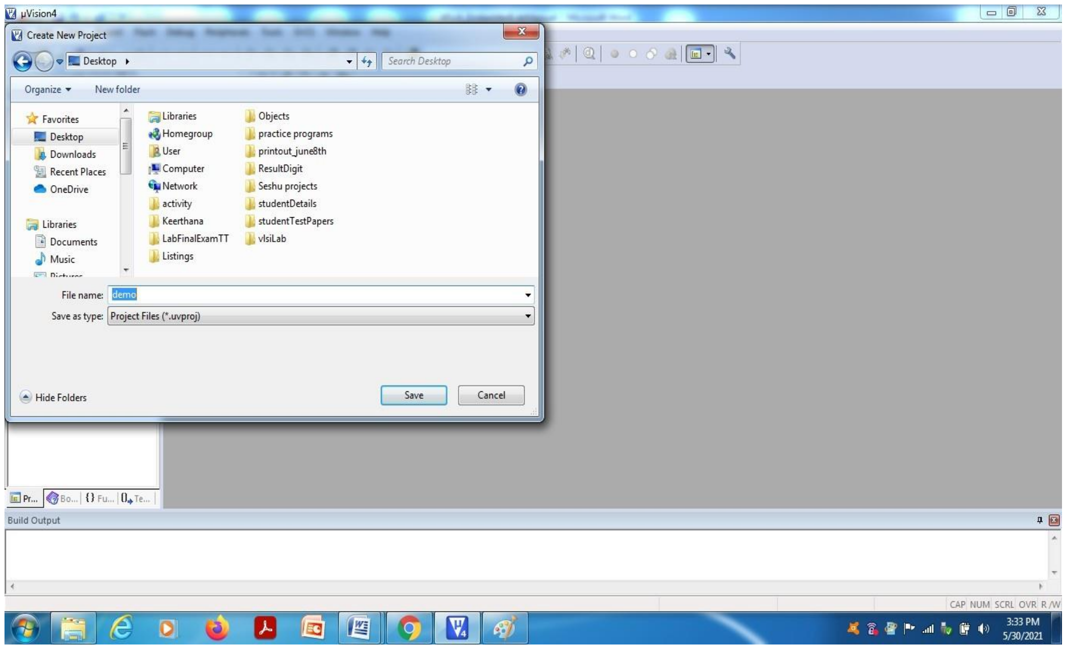
1. Double click on µvision 4 icon in thedesktop.


µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Project
Build Output
CAP NUM SCRL OVR R/W
3:28 PM
5/30/2021

2. Select "New µvision Project" from project in the menu bar.


µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
New µVision Project...
New Multi-Project Workspace...
Open Project...
Save Project in µVision4 format
Close Project
Export
Manage
Select Device for Target...
Remove Item
Options... Alt+F7
Clean targets
Build target F7
Rebuild all target files
Batch Build...
Translate... Ctrl+F7
Stop build
1 D:\ARMEMB\lap_prgs_2021\writedataRam.uvproj
2 D:\ARMEMB\lap_prgs_2021\oddeven.uvproj
3 D:\ARMEMB\lap_prgs_2021\oneszeros.uvproj
4 D:\ARMEMB\lap_prgs_2021\multiply16.uvproj
5 D:\ARMEMB\lap_prgs_2021\p5.uvproj
6 D:\ARMEMB\lap_prgs_2021\p6.uvproj
7 D:\ARMEMB\embLabPrg\p6.uvproj
8 C:\Users\User\Desktop\p5.uvproj
9 C:\Users\User\Desktop\p4.uvproj
10 C:\Keil\ARM\Examples\Blinky\Blinky.uvproj
Project
Build Output
Create a new µVision project
CAP NUM SCRL OVR R/W
3:31 PM
5/30/2021

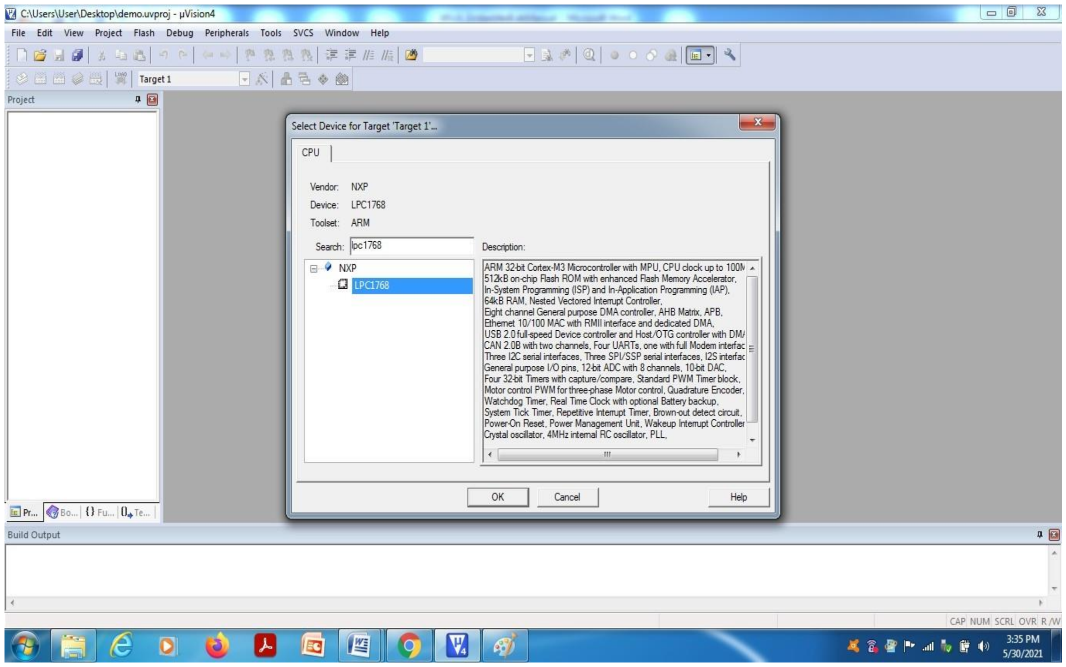
3. Browse and create a new project in the required location.



File name: demo

Save as type: Project Files (*.uvproj)

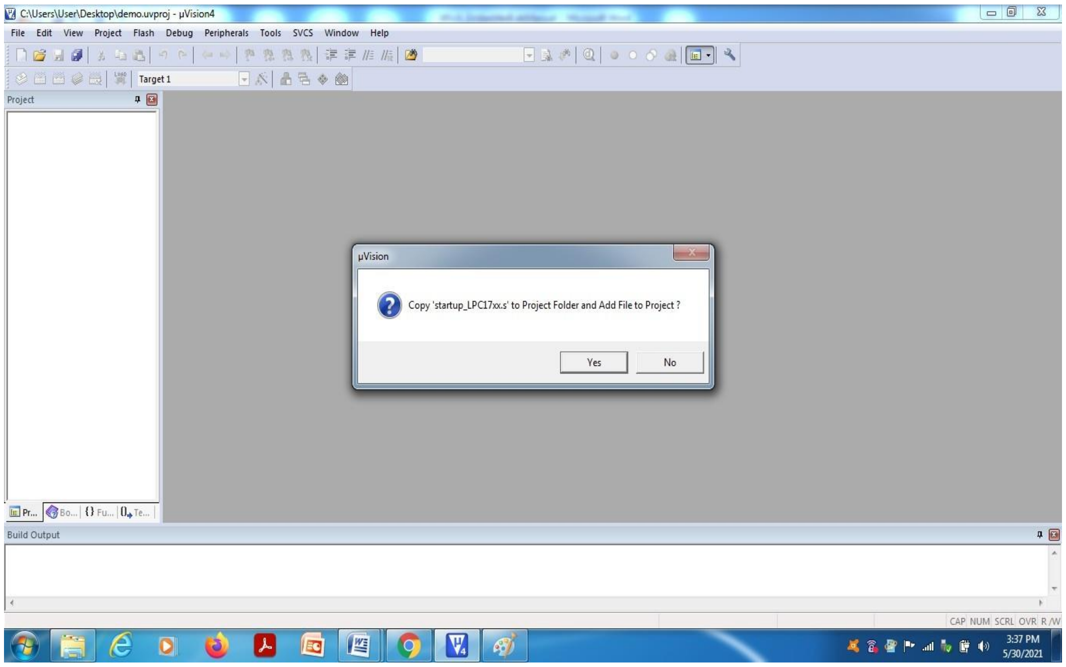
4. Select the target device (here, LPC1768 from NXP) from the list or type the exact name of the device. PressOK.


C:\Users\User\Desktop\demo.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Select Device for Target 'Target 1'...
CPU
NXP
LPC1768
OK
Cancel
Help
Build Output

Vendor: NXP
Device: LPC1768
Toolset: ARM
Search: lpc1768

Description:
ARM 32-bit Cortex-M3 Microcontroller with MPU, CPU clock up to 100M
512kB on-chip Flash ROM with enhanced Flash Memory Accelerator,
In-System Programming (ISP) and In-Application Programming (IAP),
64kB RAM, Nested Vectored Interrupt Controller,
Eight channel General purpose DMA controller, AHB Matrix, APB,
Ethernet 10/100 MAC with RMII interface and dedicated DMA,
USB 2.0 full-speed Device controller and Host/OTG controller with DM/
CAN 2.0B with two channels, Four UARTs, one with full Modem interfac
Three I2C serial interfaces, Three SPI/SSP serial interfaces, I2S interfac
General purpose I/O pins, 12-bit ADC with 8 channels, 10-bit DAC,
Four 32-bit Timers with capture/compare, Standard PWM Timer block,
Motor control PWM for three-phase Motor control, Quadrature Encoder,
Watchdog Timer, Real Time Clock with optional Battery backup,
System Tick Timer, Repetitive Interrupt Timer, Brown-out detect circuit,
Power-On Reset, Power Management Unit, Wakeup Interrupt Controller
Crystal oscillator, 4MHz internal RC oscillator, PLL,

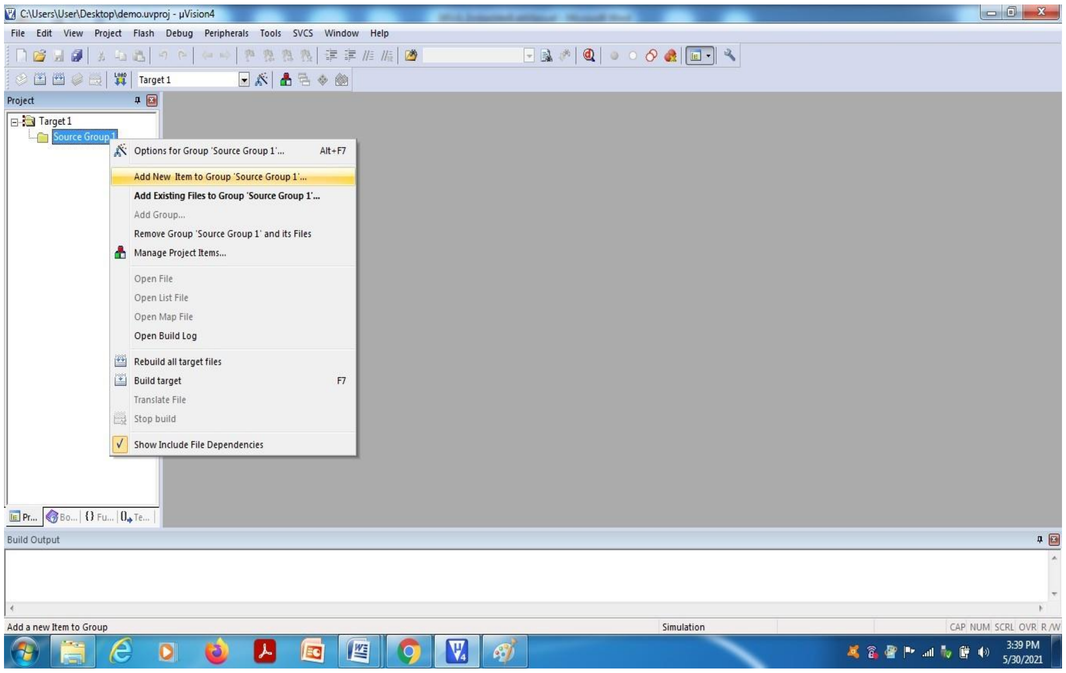
5. Copy start up to Project folder and add to project file”?- PressNO.



C:\Users\User\Desktop\demo.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Build Output
µVision
Yes
No

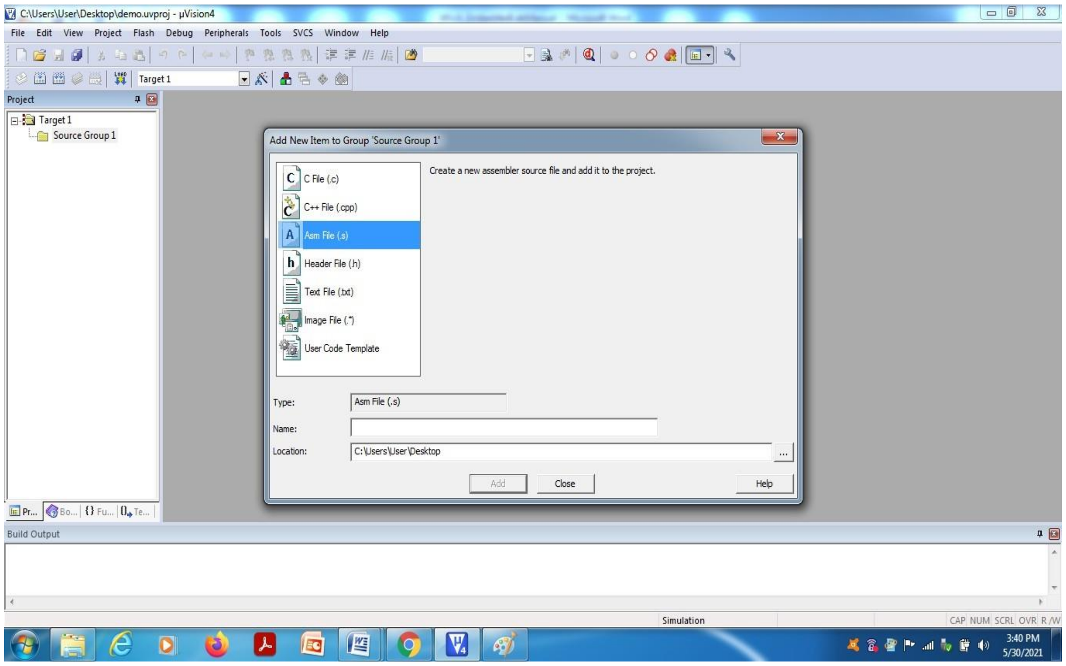
Copy 'startup_LPC17xx.s' to Project Folder and Add File to Project ?

6. In the project window, right click on source and select Add new item to group “source group1”.



C:\Users\User\Desktop\demo.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Target 1
Source Group 1
Options for Group 'Source Group 1'... Alt+F7
Add New Item to Group 'Source Group 1'...
Add Existing Files to Group 'Source Group 1'...
Add Group...
Remove Group 'Source Group 1' and its Files
Manage Project Items...
Open File
Open List File
Open Map File
Open Build Log
Rebuild all target files
Build target F7
Translate File
Stop build
Show Include File Dependencies
Build Output
Add a new Item to Group
Simulation

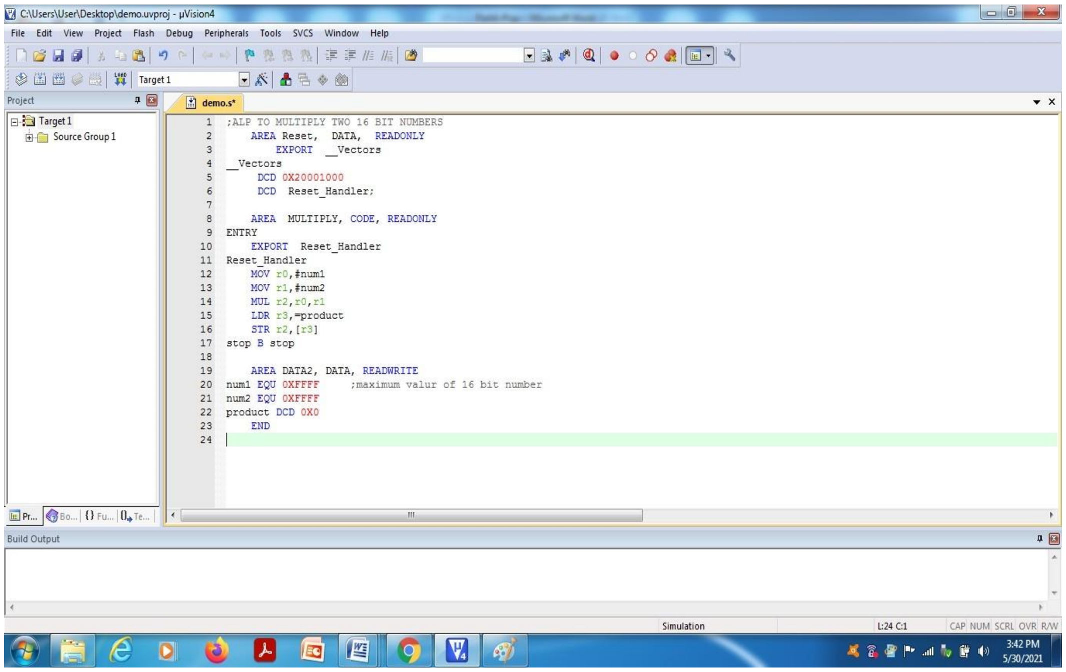
7. Select ASM file and give name of the file with .s extension and pressADD.



Create a new assembler source file and add it to the project.

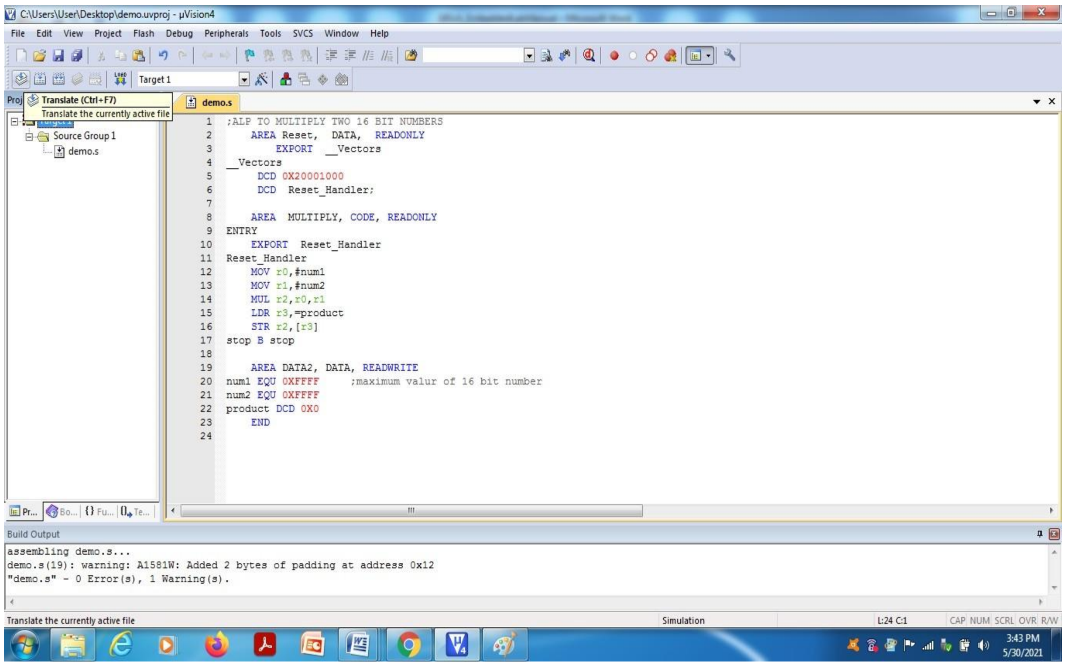
Type: Asm File (.s)
Name:
Location: C:\Users\User\Desktop

8. Type the program in the editor space andsave.


C:\Users\User\Desktop\demo.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Target 1
Source Group 1
demo.s*
Build Output
Simulation
L:24 C:1
3:42 PM
5/30/2021

```
;ALP TO MULTIPLY TWO 16 BIT NUMBERS
    AREA Reset,  DATA,  READONLY
        EXPORT  __Vectors
__Vectors
     DCD 0X20001000
     DCD  Reset_Handler;

    AREA  MULTIPLY, CODE, READONLY
ENTRY
    EXPORT  Reset_Handler
Reset_Handler
    MOV r0,#num1
    MOV r1,#num2
    MUL r2,r0,r1
    LDR r3,=product
    STR r2,[r3]
stop B stop

    AREA DATA2, DATA, READWRITE
num1 EQU 0XFFFF     ;maximum valur of 16 bit number
num2 EQU 0XFFFF
product DCD 0X0
    END
```

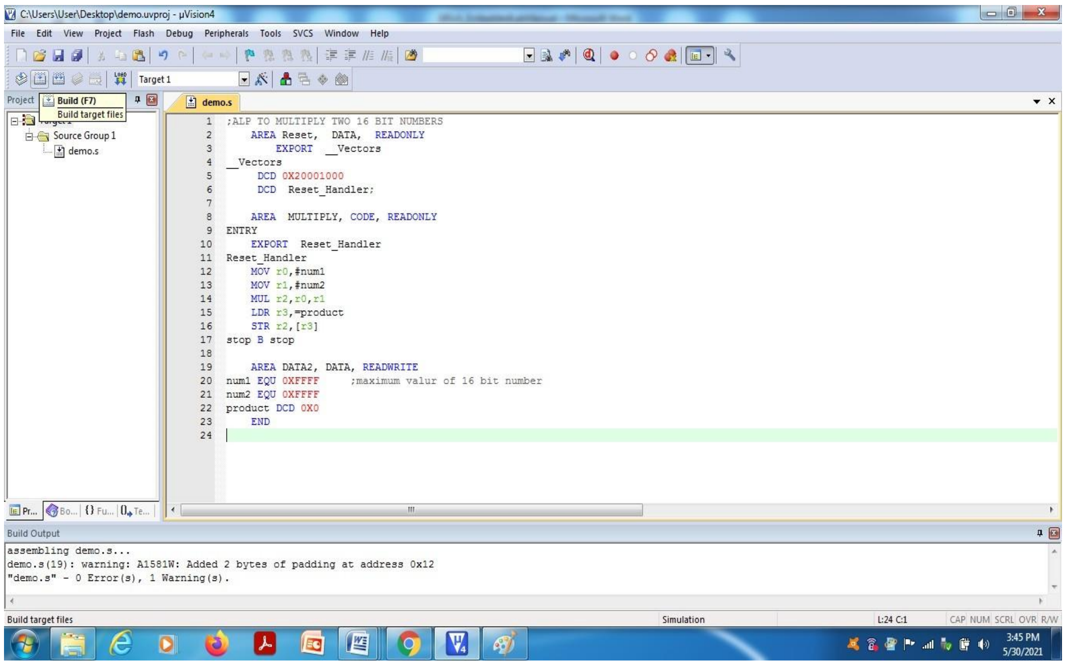
9. Translate the program by select the icon from tool bar or from menubar, also Check for errors and warnings in the bottom window.


C:\Users\User\Desktop\demo.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Translate (Ctrl+F7)
Translate the currently active file
Source Group 1
demo.s
Build Output
Translate the currently active file
Simulation
L:24 C:1
3:43 PM
5/30/2021

```
;ALP TO MULTIPLY TWO 16 BIT NUMBERS
    AREA Reset,  DATA,  READONLY
        EXPORT  __Vectors
__Vectors
     DCD 0X20001000
     DCD  Reset_Handler;

    AREA  MULTIPLY, CODE, READONLY
ENTRY
    EXPORT  Reset_Handler
Reset_Handler
    MOV r0,#num1
    MOV r1,#num2
    MUL r2,r0,r1
    LDR r3,=product
    STR r2,[r3]
stop B stop

    AREA DATA2, DATA, READWRITE
num1 EQU 0XFFFF     ;maximum valur of 16 bit number
num2 EQU 0XFFFF
product DCD 0X0
    END
```

```
assembling demo.s...
demo.s(19): warning: A1581W: Added 2 bytes of padding at address 0x12
"demo.s" - 0 Error(s), 1 Warning(s).
```

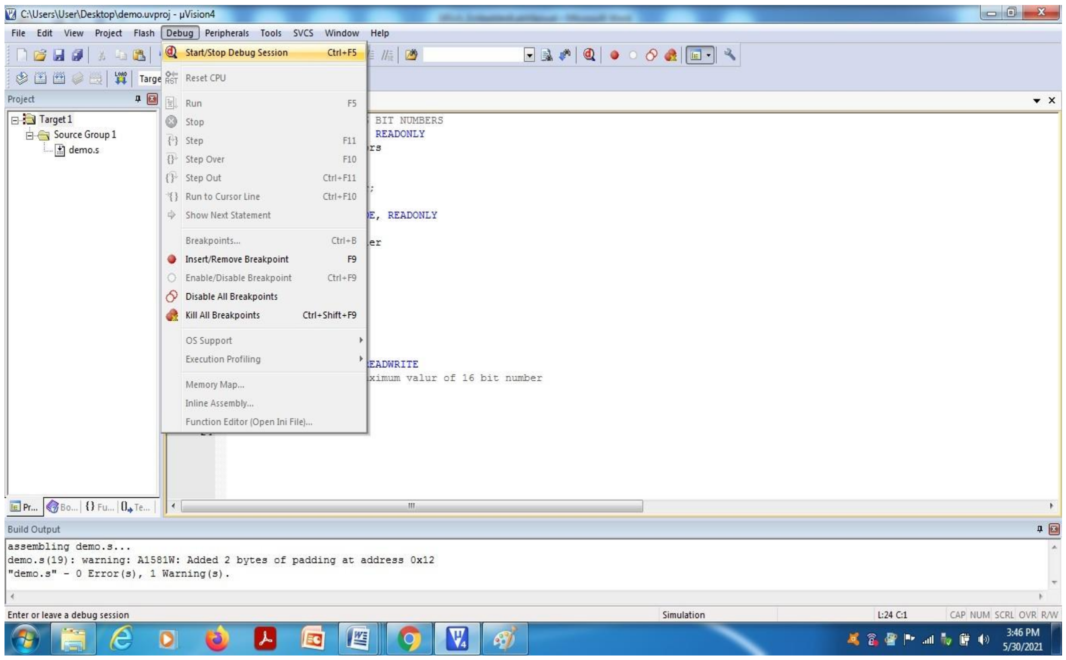
10. If no error,Select "Build" icon from tool bar or from menubar.



```
;ALP TO MULTIPLY TWO 16 BIT NUMBERS
    AREA Reset,  DATA,  READONLY
        EXPORT  __Vectors
__Vectors
     DCD 0X20001000
     DCD  Reset_Handler;

    AREA  MULTIPLY, CODE, READONLY
ENTRY
    EXPORT  Reset_Handler
Reset_Handler
    MOV r0,#num1
    MOV r1,#num2
    MUL r2,r0,r1
    LDR r3,=product
    STR r2,[r3]
stop B stop

    AREA DATA2, DATA, READWRITE
num1 EQU 0XFFFF     ;maximum valur of 16 bit number
num2 EQU 0XFFFF
product DCD 0X0
    END
```

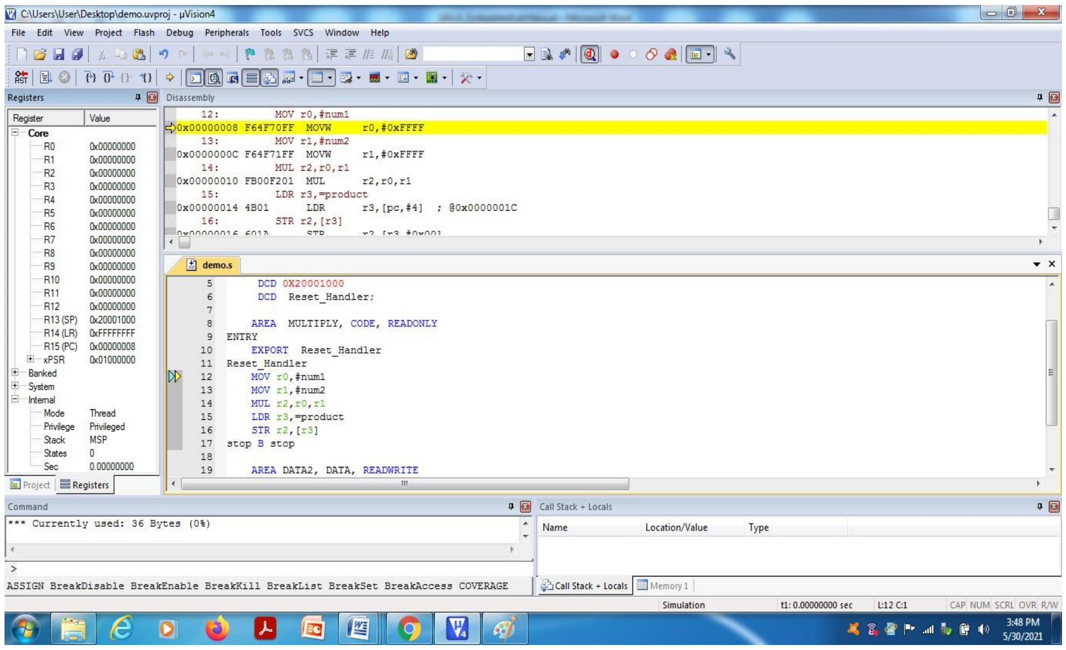
```
assembling demo.s...
demo.s(19): warning: A1581W: Added 2 bytes of padding at address 0x12
"demo.s" - 0 Error(s), 1 Warning(s).
```

11. Start the debug session from Menubar.


C:\Users\User\Desktop\demo.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Start/Stop Debug Session Ctrl+F5
Reset CPU
Run F5
Stop
Step F11
Step Over F10
Step Out Ctrl+F11
Run to Cursor Line Ctrl+F10
Show Next Statement
Breakpoints... Ctrl+B
Insert/Remove Breakpoint F9
Enable/Disable Breakpoint Ctrl+F9
Disable All Breakpoints
Kill All Breakpoints Ctrl+Shift+F9
OS Support
Execution Profiling
Memory Map...
Inline Assembly...
Function Editor (Open Ini File)...
Project
Target 1
Source Group 1
demo.s
Build Output
Enter or leave a debug session
Simulation
L:24 C:1

```
assembling demo.s...
demo.s(19): warning: A1581W: Added 2 bytes of padding at address 0x12
"demo.s" - 0 Error(s), 1 Warning(s).
```

12. PressOK


C:\Users\User\Desktop\demo.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
demo.s
Command
Call Stack + Locals
Name Location/Value Type
Project Registers
Call Stack + Locals Memory 1
Simulation
t1: 0.00000000 sec
L:12 C:1

Register	Value
Core	
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20001000
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x00000008
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	0
Sec	0.00000000

```
    12:         MOV r0,#num1
0x00000008 F64F70FF  MOVW     r0,#0xFFFF
    13:         MOV r1,#num2
0x0000000C F64F71FF  MOVW     r1,#0xFFFF
    14:         MUL r2,r0,r1
0x00000010 FB00F201  MUL      r2,r0,r1
    15:         LDR r3,=product
0x00000014 4B01      LDR      r3,[pc,#4]  ; @0x0000001C
    16:         STR r2,[r3]
```

```
      DCD 0X20001000
      DCD  Reset_Handler;

     AREA  MULTIPLY, CODE, READONLY
ENTRY
     EXPORT  Reset_Handler
Reset_Handler
     MOV r0,#num1
     MOV r1,#num2
     MUL r2,r0,r1
     LDR r3,=product
     STR r2,[r3]
stop B stop

     AREA DATA2, DATA, READWRITE
```

```
*** Currently used: 36 Bytes (0%)
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE
```

13. Press function key F11 or select “step” option under Debug menu for single step execution and verify the outputin register window/Memorywindow/xPSR.

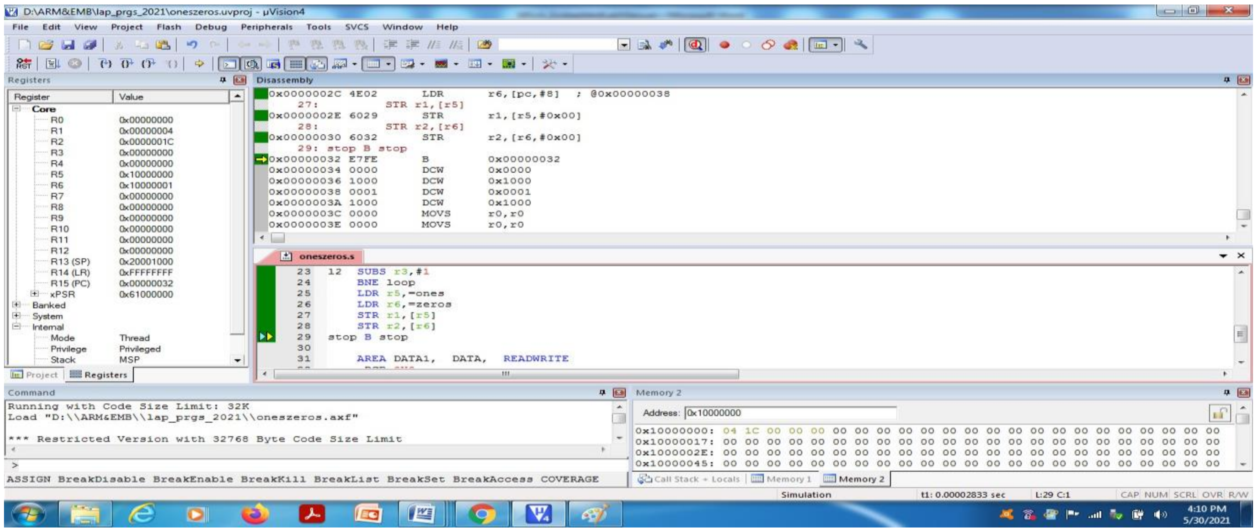
PROGRAM

ALP TO FIND THE 1’S AND 0’ IN THE GIVEN 32-BITDATA.

```
AREA Reset, DATA, READONLY
EXPORT   Vectors
  Vectors
DCD 0X20001000
DCD Reset_Handler;

AREA onezero, CODE, READONLY
num EQU 15

ENTRY
EXPORT Reset_Handler Reset_Handler
MOV r0,#num MOV r1,#0 MOV r2,#0 MOV r3,#32
loop LSRS r0,r0,#1
BCS l1 ADD r2,#1
B l2
l1 ADD r1,#1
l2 SUBS r3,#1
BNE loop

LDR r5,=ones LDR r6,=zeros STR r1,[r5] STR r2,[r6]
stop B stop

AREA DATA1, DATA, READWRITE
ones DCB 0X0 zeros DCB 0X0
END
```

RESULT:

If num=15d□no of 1's=4 and No.of 0's=28d=1Ch


D:\ARM&EMB\lap_prgs_2021\oneszeros.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
oneszeros.s
Command
Memory 2
Project Registers
Call Stack + Locals Memory 1 Memory 2
Simulation
CAP NUM SCRL OVR R/W
4:10 PM
5/30/2021

Register	Value
Core	
R0	0x00000000
R1	0x00000004
R2	0x0000001C
R3	0x00000000
R4	0x00000000
R5	0x10000000
R6	0x10000001
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20001000
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x00000032
xPSR	0x61000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP

```
0x0000002C 4E02      LDR      r6,[pc,#8]   ; @0x00000038
    27:          STR r1,[r5]
0x0000002E 6029      STR      r1,[r5,#0x00]
    28:          STR r2,[r6]
0x00000030 6032      STR      r2,[r6,#0x00]
    29: stop B stop
0x00000032 E7FE      B        0x00000032
0x00000034 0000      DCW      0x0000
0x00000036 1000      DCW      0x1000
0x00000038 0001      DCW      0x0001
0x0000003A 1000      DCW      0x1000
0x0000003C 0000      MOVS     r0,r0
0x0000003E 0000      MOVS     r0,r0
```

```
23   12   SUBS r3,#1
24        BNE loop
25        LDR r5,=ones
26        LDR r6,=zeros
27        STR r1,[r5]
28        STR r2,[r6]
29   stop B stop
30
31        AREA DATA1,  DATA,  READWRITE
```

```
Running with Code Size Limit: 32K
Load "D:\\ARM&EMB\\lap_prgs_2021\\oneszeros.axf"

*** Restricted Version with 32768 Byte Code Size Limit
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE
```

```
Address: 0x10000000
0x10000000: 04 1C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x10000017: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x1000002E: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x10000045: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
t1: 0.00002833 sec    L:29 C:1
```

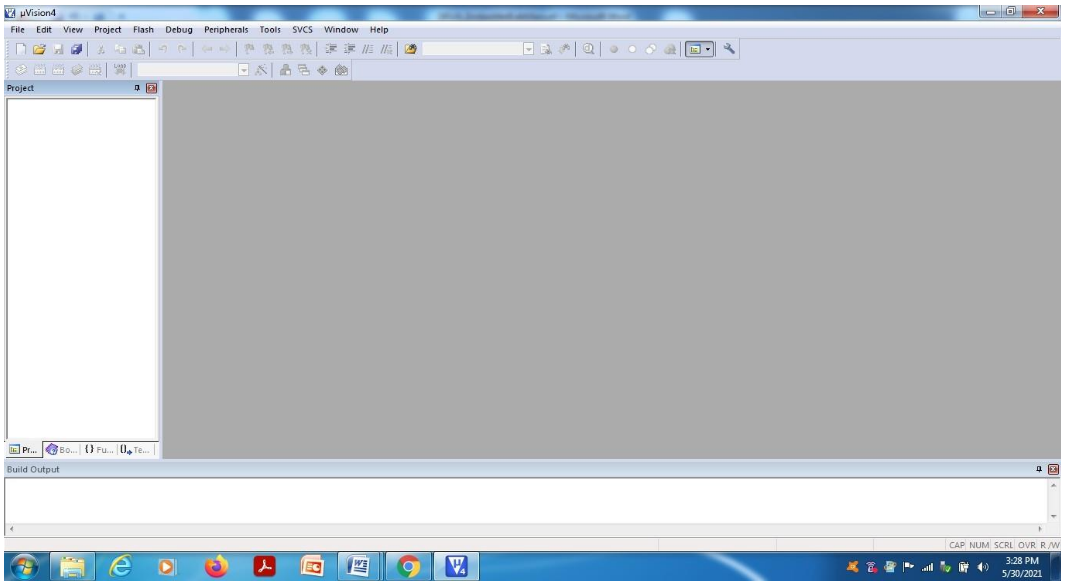
12. ALP to determine whether the given16-bitnumber is ODD or EVEN.

Aim: To find whether the given16-bitnumber is ODD or EVEN using ARM cortex-3 ALP program.

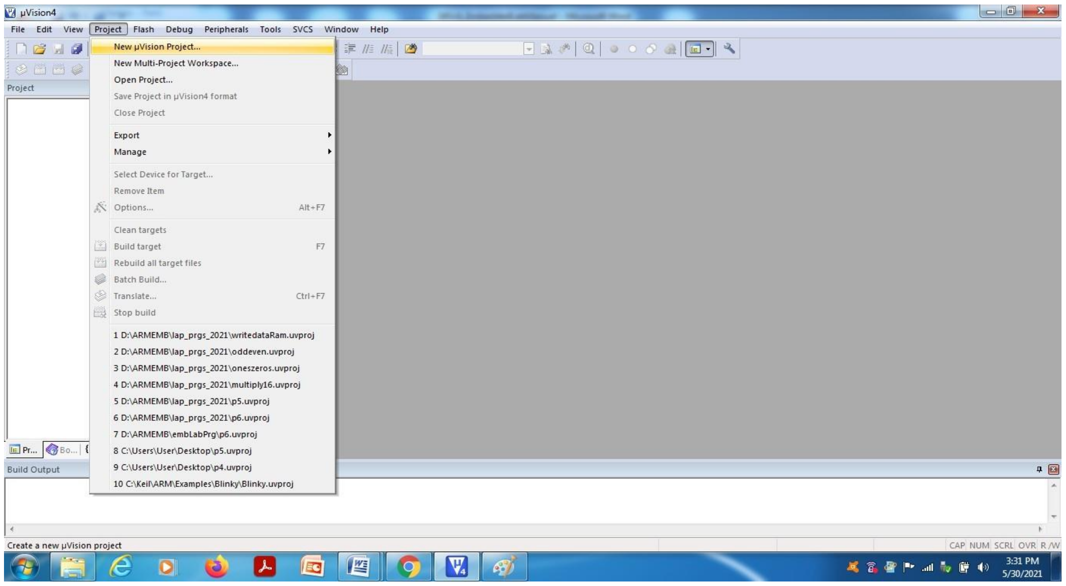
APPARATUS:
1. PC with Windows 10, 64-bit OS.
2. Keil µVision4 Software.

PROCEDURE:

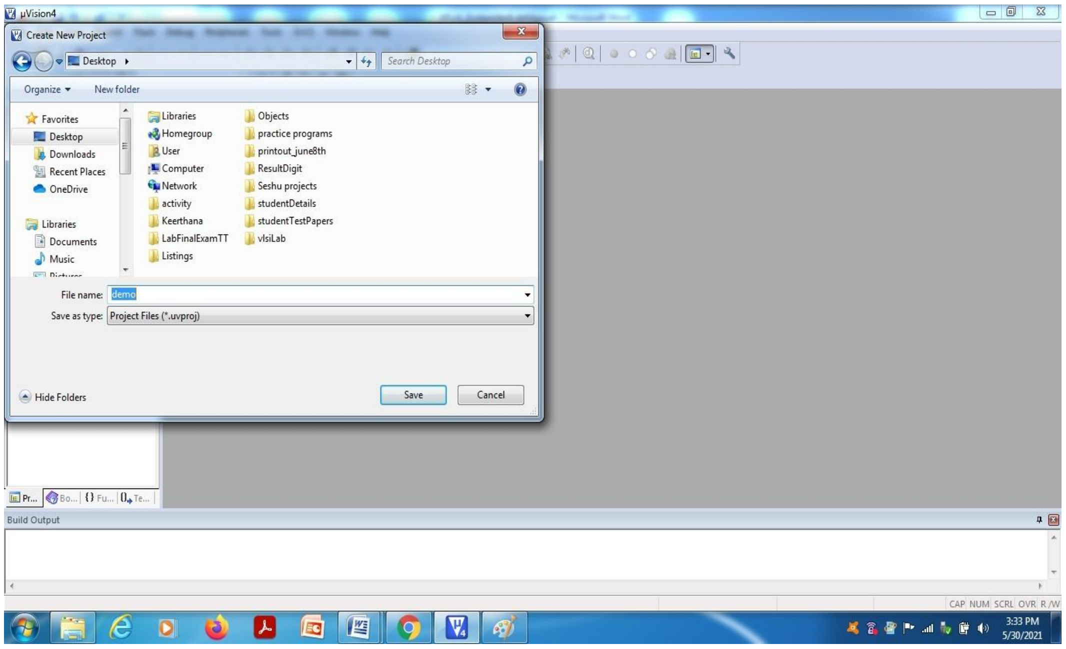
1. Double click on µvision 4 icon in the desktop.


µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Project
Build Output
CAP NUM SCRL OVR R/W
3:28 PM
5/30/2021

2. Select "New µvision Project" from project in the menu bar.


µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
New µVision Project...
New Multi-Project Workspace...
Open Project...
Save Project in µVision4 format
Close Project
Export
Manage
Select Device for Target...
Remove Item
Options... Alt+F7
Clean targets
Build target F7
Rebuild all target files
Batch Build...
Translate... Ctrl+F7
Stop build
1 D:\ARMEMB\lap_prgs_2021\writedataRam.uvproj
2 D:\ARMEMB\lap_prgs_2021\oddeven.uvproj
3 D:\ARMEMB\lap_prgs_2021\oneszeros.uvproj
4 D:\ARMEMB\lap_prgs_2021\multiply16.uvproj
5 D:\ARMEMB\lap_prgs_2021\p5.uvproj
6 D:\ARMEMB\lap_prgs_2021\p6.uvproj
7 D:\ARMEMB\embLabPrg\p6.uvproj
8 C:\Users\User\Desktop\p5.uvproj
9 C:\Users\User\Desktop\p4.uvproj
10 C:\Keil\ARM\Examples\Blinky\Blinky.uvproj
Project
Build Output
Create a new µVision project
CAP NUM SCRL OVR R/W
3:31 PM
5/30/2021

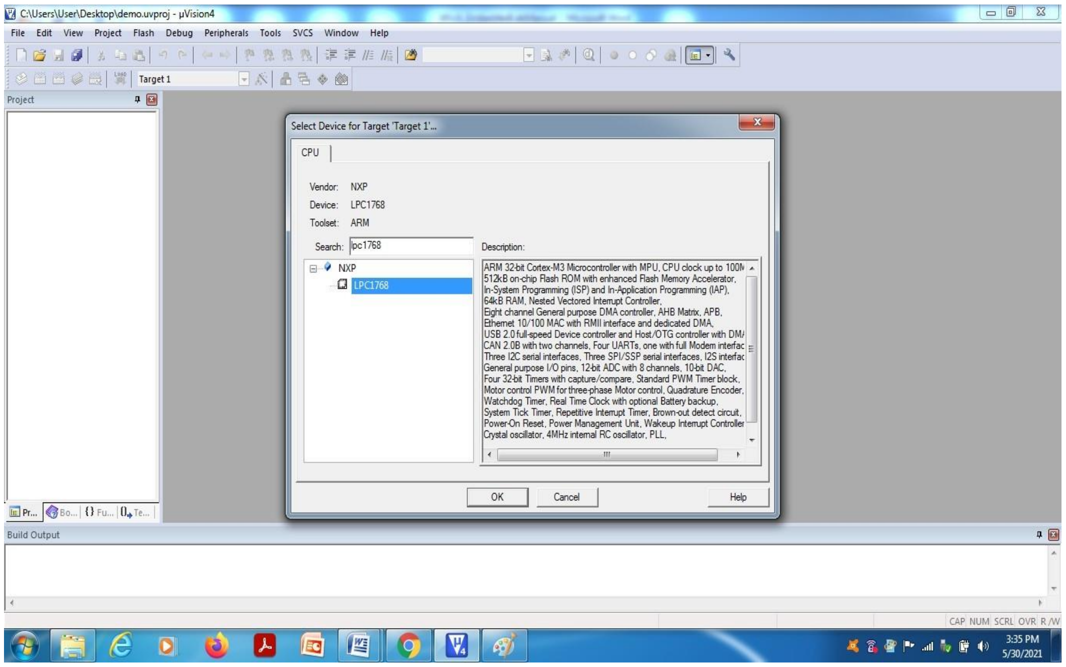
3. Browse and create a new project in the required location.



File name: demo

Save as type: Project Files (*.uvproj)

4. Select the target device (here, LPC1768 from NXP) from the list or type the exact name of the device. Press OK.



Vendor: NXP

Device: LPC1768

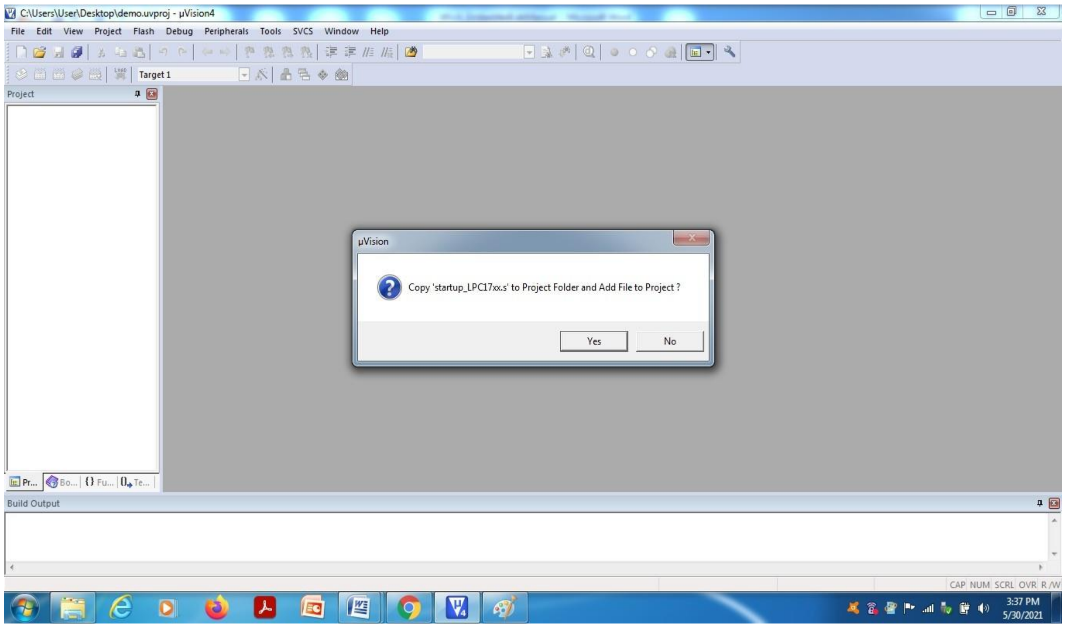
Toolset: ARM

Search: lpc1768

Description:

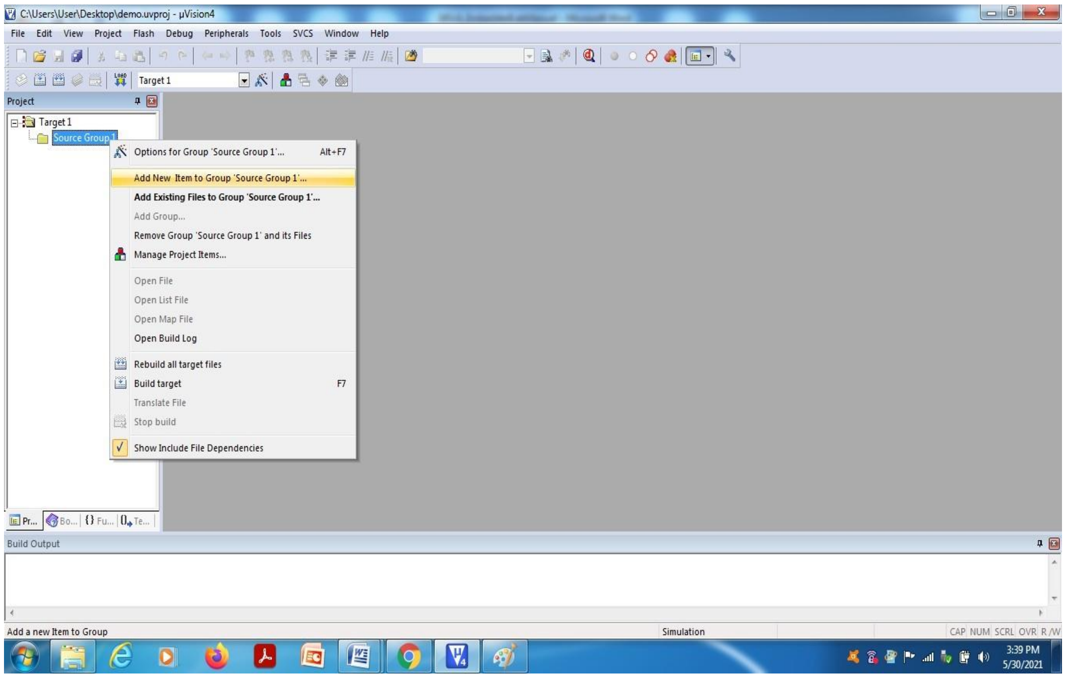
ARM 32-bit Cortex-M3 Microcontroller with MPU, CPU clock up to 100M
512kB on-chip Flash ROM with enhanced Flash Memory Accelerator,
In-System Programming (ISP) and In-Application Programming (IAP),
64kB RAM, Nested Vectored Interrupt Controller,
Eight channel General purpose DMA controller, AHB Matrix, APB,
Ethernet 10/100 MAC with RMII interface and dedicated DMA,
USB 2.0 full-speed Device controller and Host/OTG controller with DM/
CAN 2.0B with two channels, Four UARTs, one with full Modem interfac
Three I2C serial interfaces, Three SPI/SSP serial interfaces, I2S interfac
General purpose I/O pins, 12-bit ADC with 8 channels, 10-bit DAC,
Four 32-bit Timers with capture/compare, Standard PWM Timer block,
Motor control PWM for three-phase Motor control, Quadrature Encoder,
Watchdog Timer, Real Time Clock with optional Battery backup,
System Tick Timer, Repetitive Interrupt Timer, Brown-out detect circuit,
Power-On Reset, Power Management Unit, Wakeup Interrupt Controller
Crystal oscillator, 4MHz internal RC oscillator, PLL,

5. Copy start up to Project folder and add to project file"?- Press NO.

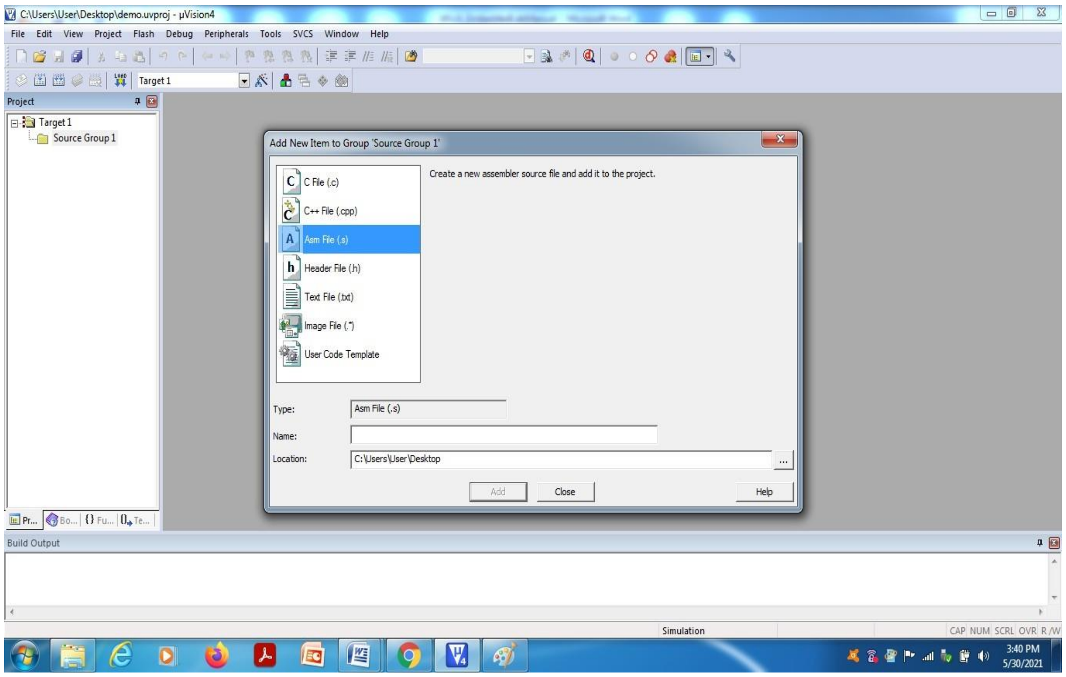


Copy 'startup_LPC17xx.s' to Project Folder and Add File to Project ?

6. In the project window, right click on source and select Add new item to group "source group1".


C:\Users\User\Desktop\demo.uvproj - µVision4
Target 1
Source Group 1
Options for Group 'Source Group 1'... Alt+F7
Add New Item to Group 'Source Group 1'...
Add Existing Files to Group 'Source Group 1'...
Add Group...
Remove Group 'Source Group 1' and its Files
Manage Project Items...
Open File
Open List File
Open Map File
Open Build Log
Rebuild all target files
Build target F7
Translate File
Stop build
Show Include File Dependencies
Build Output
Add a new Item to Group
Simulation

7. Select ASM file and give name of the file with .s extension and press ADD.



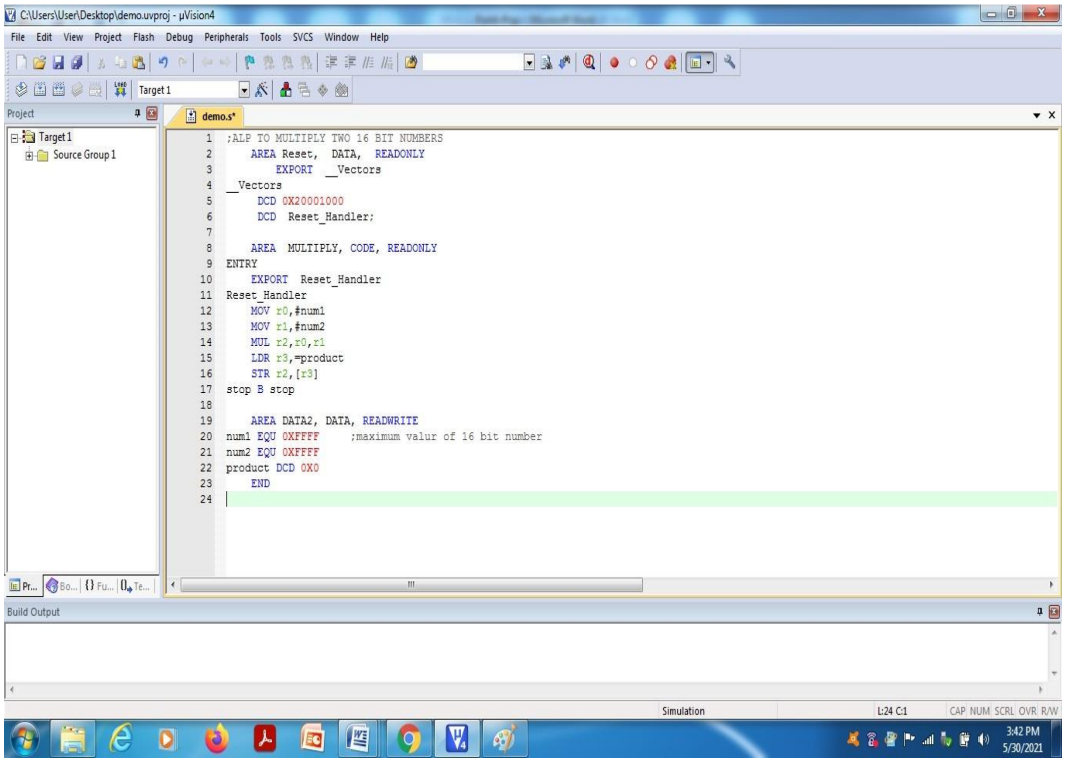
Create a new assembler source file and add it to the project.

Type: Asm File (.s)

Name:

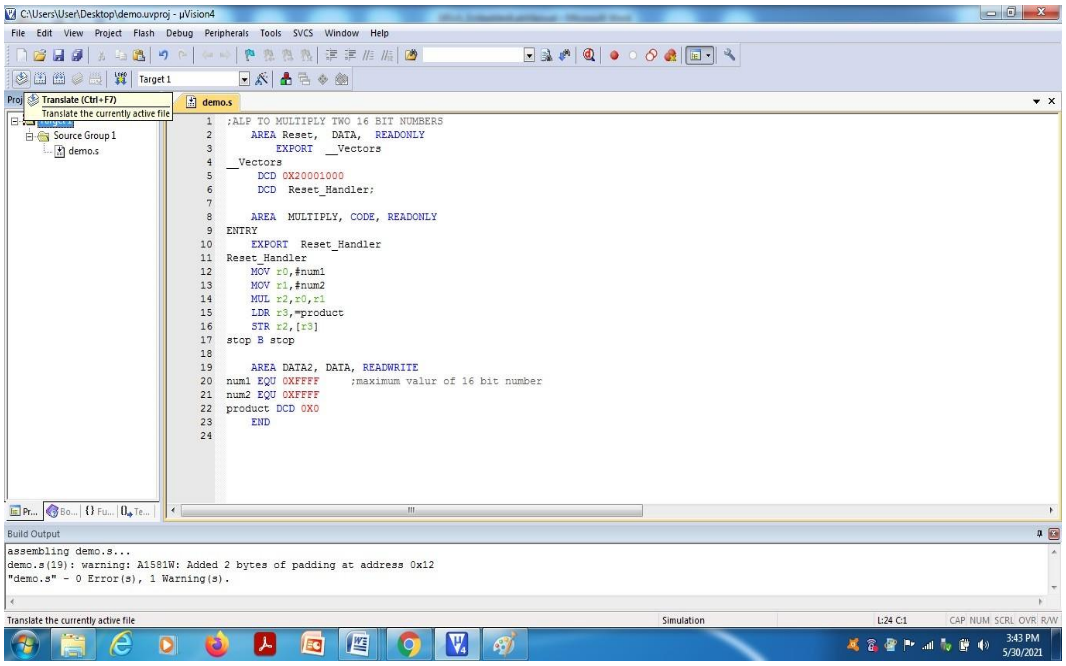
Location: C:\Users\User\Desktop

8. Type the program in the editor space and save.


C:\Users\User\Desktop\demo.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Target 1
Source Group 1
demo.s*
Build Output
Simulation
L:24 C:1
CAP NUM SCRL OVR R/W
3:42 PM
5/30/2021

```
;ALP TO MULTIPLY TWO 16 BIT NUMBERS
    AREA Reset,  DATA,  READONLY
        EXPORT  __Vectors
__Vectors
     DCD 0X20001000
     DCD  Reset_Handler;

    AREA  MULTIPLY, CODE, READONLY
ENTRY
    EXPORT  Reset_Handler
Reset_Handler
    MOV r0,#num1
    MOV r1,#num2
    MUL r2,r0,r1
    LDR r3,=product
    STR r2,[r3]
stop B stop

    AREA DATA2, DATA, READWRITE
num1 EQU 0XFFFF     ;maximum valur of 16 bit number
num2 EQU 0XFFFF
product DCD 0X0
    END
```

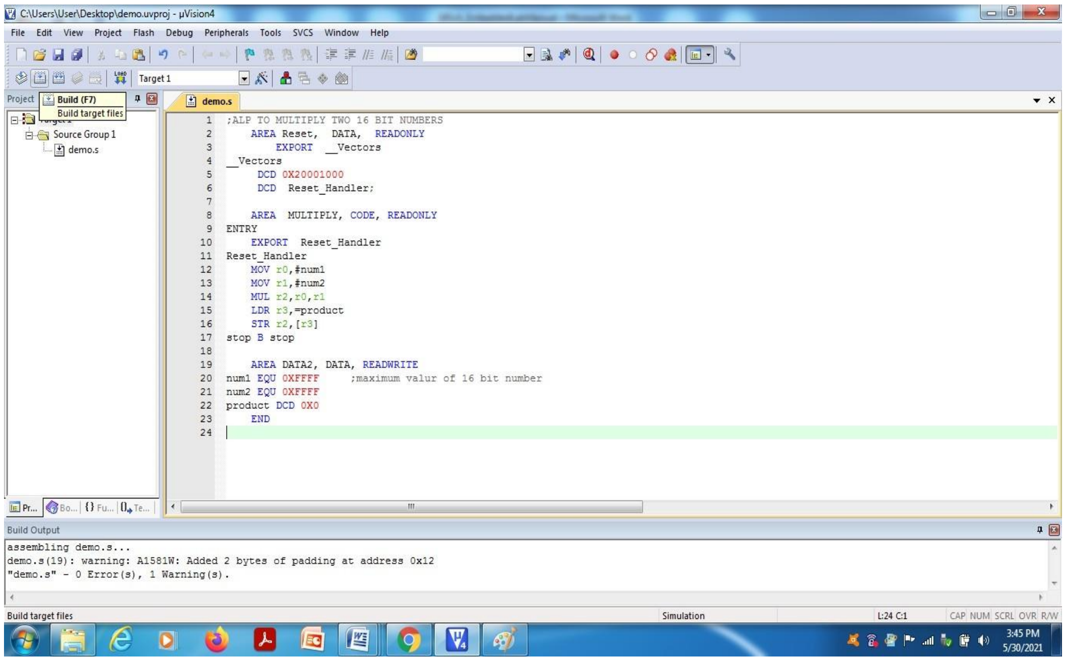
9. Translate the program by select the icon from tool bar or from menubar, also Check for errors and warnings in the bottom window.


C:\Users\User\Desktop\demo.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Translate (Ctrl+F7)
Translate the currently active file
Source Group 1
demo.s
Build Output
Simulation
L:24 C:1
CAP NUM SCRL OVR R/W
3:43 PM
5/30/2021

```
;ALP TO MULTIPLY TWO 16 BIT NUMBERS
    AREA Reset,  DATA,  READONLY
        EXPORT  __Vectors
__Vectors
     DCD 0X20001000
     DCD  Reset_Handler;

    AREA  MULTIPLY, CODE, READONLY
ENTRY
    EXPORT  Reset_Handler
Reset_Handler
    MOV r0,#num1
    MOV r1,#num2
    MUL r2,r0,r1
    LDR r3,=product
    STR r2,[r3]
stop B stop

    AREA DATA2, DATA, READWRITE
num1 EQU 0XFFFF     ;maximum valur of 16 bit number
num2 EQU 0XFFFF
product DCD 0X0
    END
```

```
assembling demo.s...
demo.s(19): warning: A1581W: Added 2 bytes of padding at address 0x12
"demo.s" - 0 Error(s), 1 Warning(s).
```

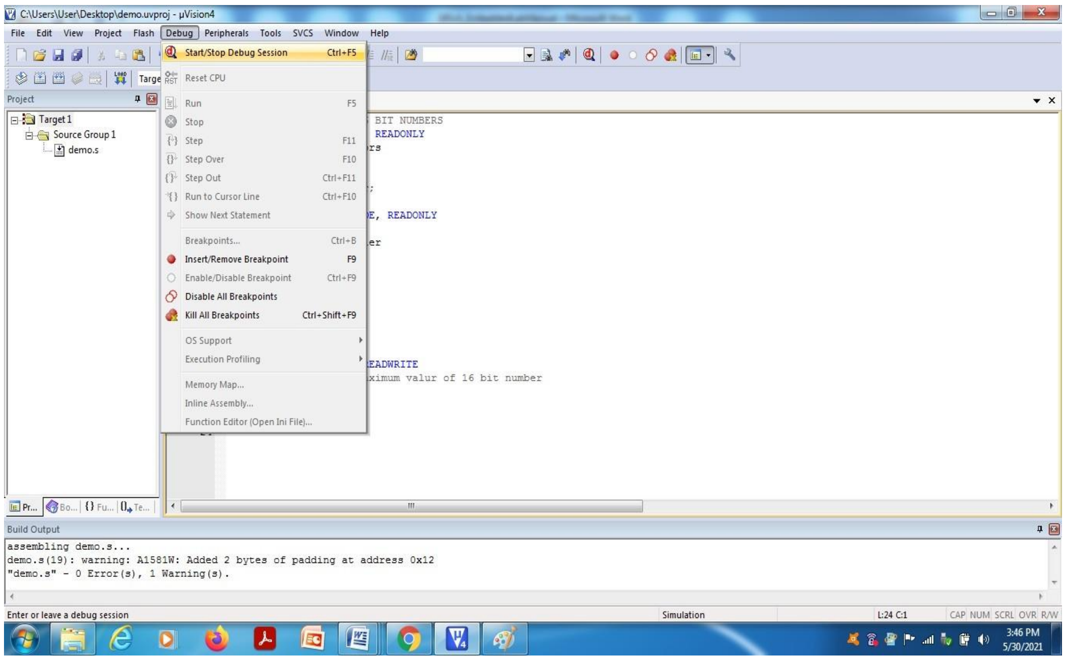
10. If no error, Select "Build" icon from tool bar or from menubar.



```
;ALP TO MULTIPLY TWO 16 BIT NUMBERS
    AREA Reset,  DATA,  READONLY
        EXPORT  __Vectors
__Vectors
     DCD 0X20001000
     DCD  Reset_Handler;

    AREA  MULTIPLY, CODE, READONLY
ENTRY
    EXPORT  Reset_Handler
Reset_Handler
    MOV r0,#num1
    MOV r1,#num2
    MUL r2,r0,r1
    LDR r3,=product
    STR r2,[r3]
stop B stop

    AREA DATA2, DATA, READWRITE
num1 EQU 0XFFFF     ;maximum valur of 16 bit number
num2 EQU 0XFFFF
product DCD 0X0
    END
```

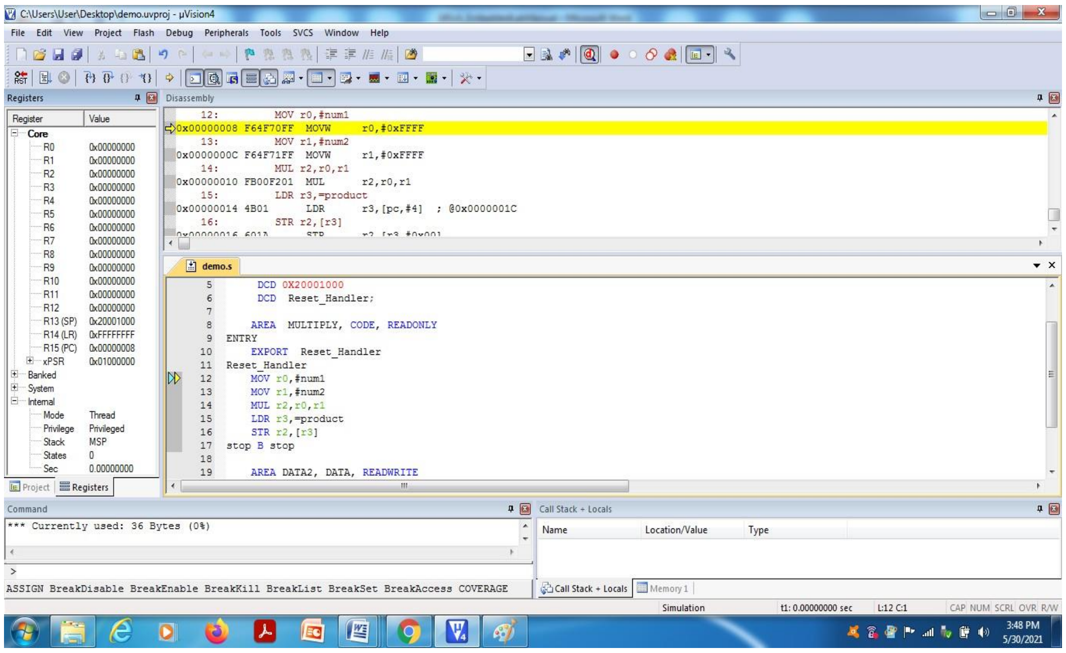
```
assembling demo.s...
demo.s(19): warning: A1581W: Added 2 bytes of padding at address 0x12
"demo.s" - 0 Error(s), 1 Warning(s).
```

11. Start the debug session from Menubar.


C:\Users\User\Desktop\demo.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Start/Stop Debug Session Ctrl+F5
Reset CPU
Run F5
Stop
Step F11
Step Over F10
Step Out Ctrl+F11
Run to Cursor Line Ctrl+F10
Show Next Statement
Breakpoints... Ctrl+B
Insert/Remove Breakpoint F9
Enable/Disable Breakpoint Ctrl+F9
Disable All Breakpoints
Kill All Breakpoints Ctrl+Shift+F9
OS Support
Execution Profiling
Memory Map...
Inline Assembly...
Function Editor (Open Ini File)...
Project
Target 1
Source Group 1
demo.s
Build Output
Enter or leave a debug session
Simulation
L:24 C:1
3:46 PM
5/30/2021

```
assembling demo.s...
demo.s(19): warning: A1581W: Added 2 bytes of padding at address 0x12
"demo.s" - 0 Error(s), 1 Warning(s).
```

12. PressOK



Register	Value
Core	
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20001000
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x00000008
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	0
Sec	0.00000000

```
    12:          MOV r0,#num1
0x00000008 F64F70FF  MOVW      r0,#0xFFFF
    13:          MOV r1,#num2
0x0000000C F64F71FF  MOVW      r1,#0xFFFF
    14:          MUL r2,r0,r1
0x00000010 FB00F201  MUL       r2,r0,r1
    15:          LDR r3,=product
0x00000014 4B01      LDR       r3,[pc,#4]  ; @0x0000001C
    16:          STR r2,[r3]
```

```
      DCD 0X20001000
      DCD  Reset_Handler;

    AREA  MULTIPLY, CODE, READONLY
ENTRY
    EXPORT  Reset_Handler
Reset_Handler
    MOV r0,#num1
    MOV r1,#num2
    MUL r2,r0,r1
    LDR r3,=product
    STR r2,[r3]
stop B stop

    AREA DATA2, DATA, READWRITE
```

```
*** Currently used: 36 Bytes (0%)
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE
```

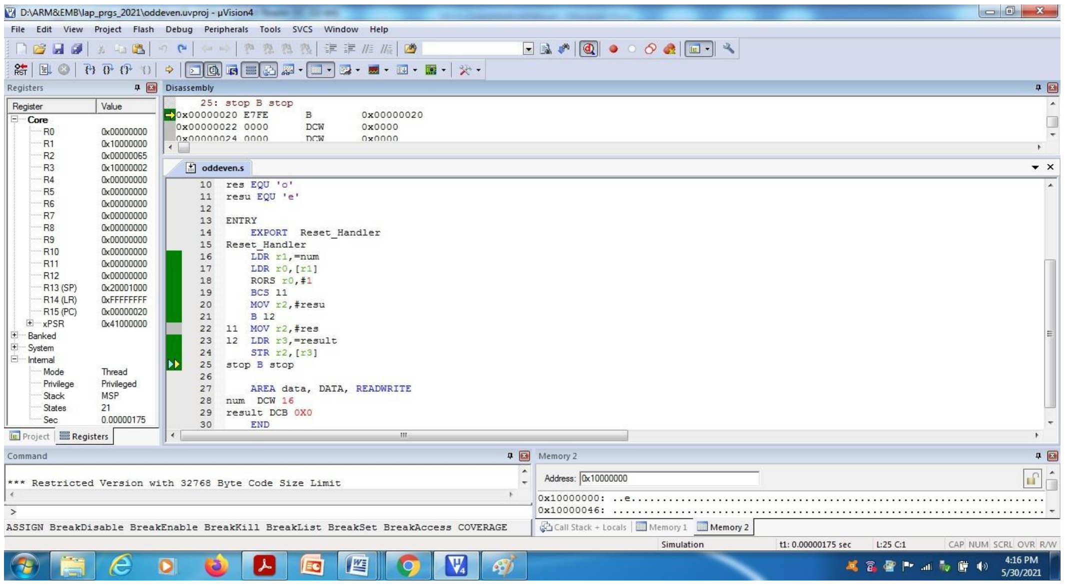
13. Press function key F11 or select “step” option under Debug menu for single step execution and verify the outputin register window/Memorywindow/xPSR.

PROGRAM:

ALP to determine whether the given16-bitnumber is ODD or EVEN

```
AREA Reset, DATA, READONLY
EXPORT   Vectors
  Vectors
DCD 0X20001000
DCD Reset_Handler;
AREA oddeven, CODE, READONLY
res EQU 'o' resu EQU 'e'
ENTRY
EXPORT Reset_Handler Reset_Handler
LDR r1,=num LDR r0,[r1] RORS r0,#1 BCS l1
MOV r2,#resu B l2
l1        MOV r2,#res l2 LDR r3,=result
STR r2,[r3]
stop B stop
AREA data, DATA, READWRITE
num DCW 16 result DCB 0X0
END
```

RESULT:
num=16d.Hence it is EVEN


D:\ARM&EMB\lap_prgs_2021\oddeven.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
oddeven.s
Project Registers
Command
Memory 2
Call Stack + Locals Memory 1 Memory 2
Simulation
t1: 0.00000175 sec
L:25 C:1
CAP NUM SCRL OVR R/W
4:16 PM
5/30/2021

Register	Value
Core	
R0	0x00000000
R1	0x10000000
R2	0x00000065
R3	0x10000002
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20001000
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x00000020
xPSR	0x41000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	21
Sec	0.00000175

```
    25: stop B stop
0x00000020 E7FE      B        0x00000020
0x00000022 0000      DCW      0x0000
0x00000024 0000      DCW      0x0000
```

```
res EQU 'o'
resu EQU 'e'

ENTRY
    EXPORT  Reset_Handler
Reset_Handler
    LDR r1,=num
    LDR r0,[r1]
    RORS r0,#1
    BCS l1
    MOV r2,#resu
    B l2
l1  MOV r2,#res
l2  LDR r3,=result
    STR r2,[r3]
stop B stop

    AREA data, DATA, READWRITE
num  DCW 16
result DCB 0X0
    END
```

```
*** Restricted Version with 32768 Byte Code Size Limit
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE
```

```
Address: 0x10000000
0x10000000: ..e.................................................................
0x10000046: ....................................................................
```

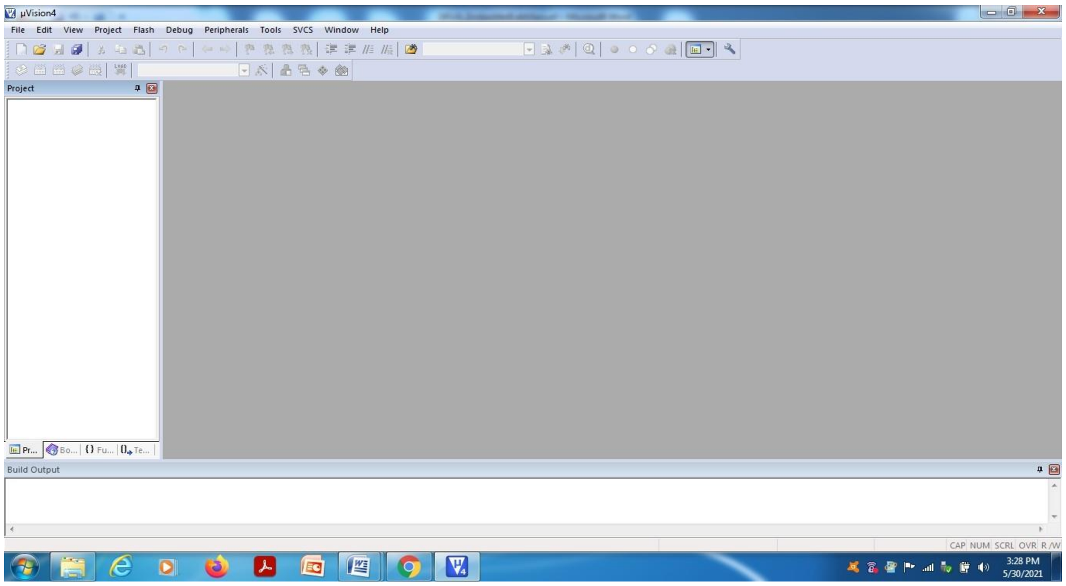
13. ALP to write data in RAM

Aim: To write data in RAM using ARM cortex-3 ALP program.

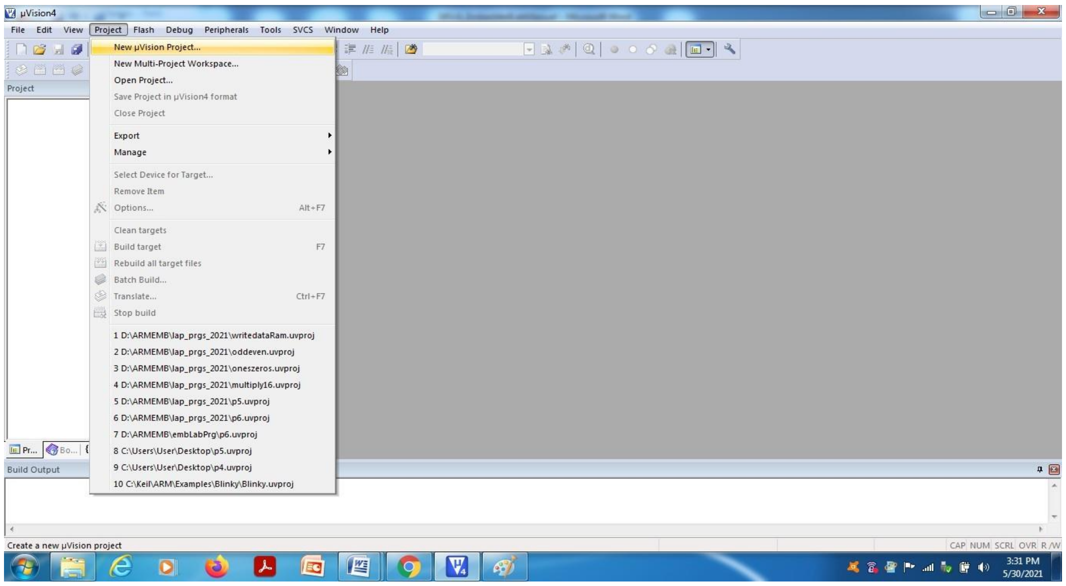
APPARATUS:
1. PC with Windows 10, 64-bit OS.
2. Keil µVision4 Software.

PROCEDURE:

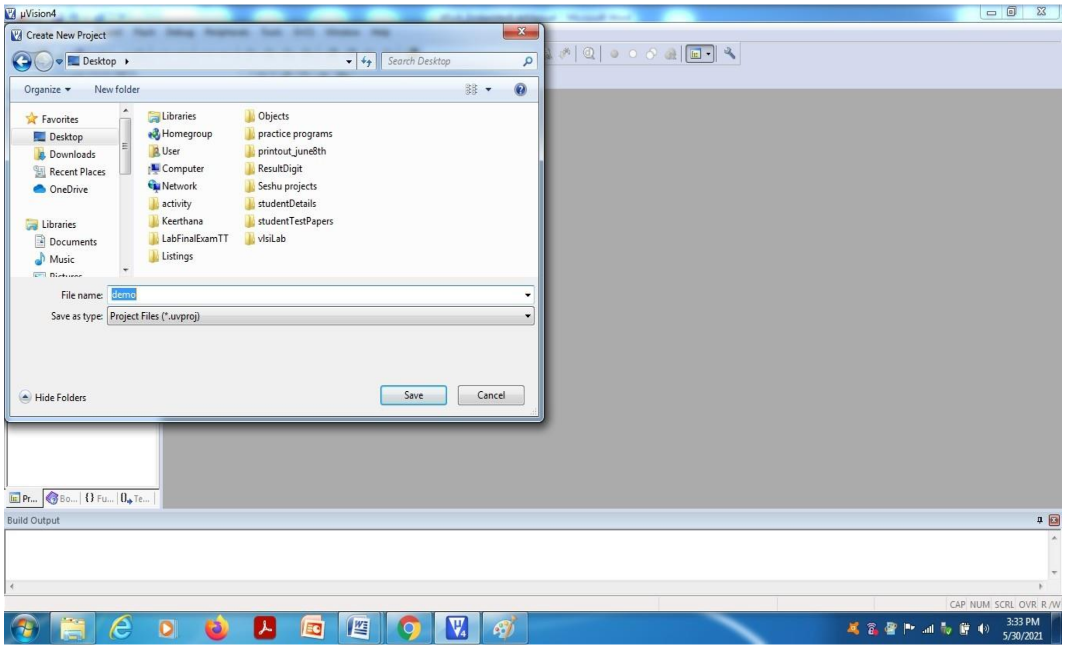
1. Double click on µvision 4 icon in the desktop.


µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Project
Build Output
CAP NUM SCRL OVR R/W
3:28 PM
5/30/2021

2. Select "New µvision Project" from project in the menu bar.


µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
New µVision Project...
New Multi-Project Workspace...
Open Project...
Save Project in µVision4 format
Close Project
Export
Manage
Select Device for Target...
Remove Item
Options... Alt+F7
Clean targets
Build target F7
Rebuild all target files
Batch Build...
Translate... Ctrl+F7
Stop build
1 D:\ARMEMB\lap_prgs_2021\writedataRam.uvproj
2 D:\ARMEMB\lap_prgs_2021\oddeven.uvproj
3 D:\ARMEMB\lap_prgs_2021\oneszeros.uvproj
4 D:\ARMEMB\lap_prgs_2021\multiply16.uvproj
5 D:\ARMEMB\lap_prgs_2021\p5.uvproj
6 D:\ARMEMB\lap_prgs_2021\p6.uvproj
7 D:\ARMEMB\embLabPrg\p6.uvproj
8 C:\Users\User\Desktop\p5.uvproj
9 C:\Users\User\Desktop\p4.uvproj
10 C:\Keil\ARM\Examples\Blinky\Blinky.uvproj
Project
Build Output
Create a new µVision project
CAP NUM SCRL OVR R/W
3:31 PM
5/30/2021

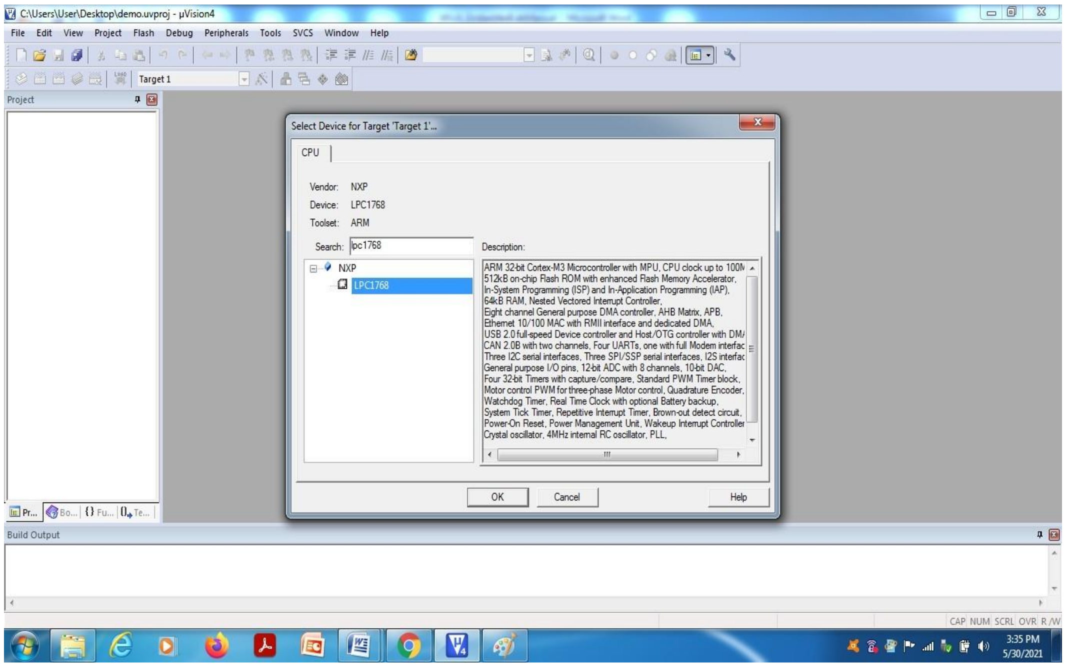
3. Browse and create a new project in the required location.



File name: demo

Save as type: Project Files (*.uvproj)

4. Select the target device (here, LPC1768 from NXP) from the list or type the exact name of the device. Press OK.

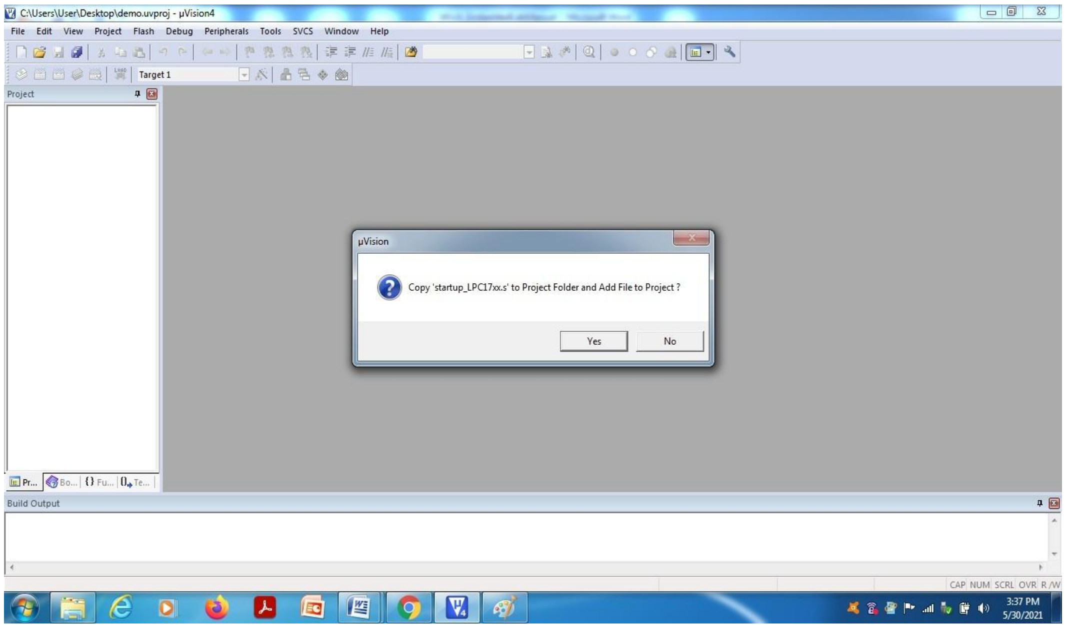


Vendor: NXP
Device: LPC1768
Toolset: ARM
Search: lpc1768

Description:

```
ARM 32-bit Cortex-M3 Microcontroller with MPU, CPU clock up to 100M
512kB on-chip Flash ROM with enhanced Flash Memory Accelerator,
In-System Programming (ISP) and In-Application Programming (IAP),
64kB RAM, Nested Vectored Interrupt Controller,
Eight channel General purpose DMA controller, AHB Matrix, APB,
Ethernet 10/100 MAC with RMII interface and dedicated DMA,
USB 2.0 full-speed Device controller and Host/OTG controller with DM/
CAN 2.0B with two channels, Four UARTs, one with full Modem interfac
Three I2C serial interfaces, Three SPI/SSP serial interfaces, I2S interfac
General purpose I/O pins, 12-bit ADC with 8 channels, 10-bit DAC,
Four 32-bit Timers with capture/compare, Standard PWM Timer block,
Motor control PWM for three-phase Motor control, Quadrature Encoder,
Watchdog Timer, Real Time Clock with optional Battery backup,
System Tick Timer, Repetitive Interrupt Timer, Brown-out detect circuit,
Power-On Reset, Power Management Unit, Wakeup Interrupt Controller
Crystal oscillator, 4MHz internal RC oscillator, PLL,
```

5. Copy start up to Project folder and add to project file”?- Press NO.

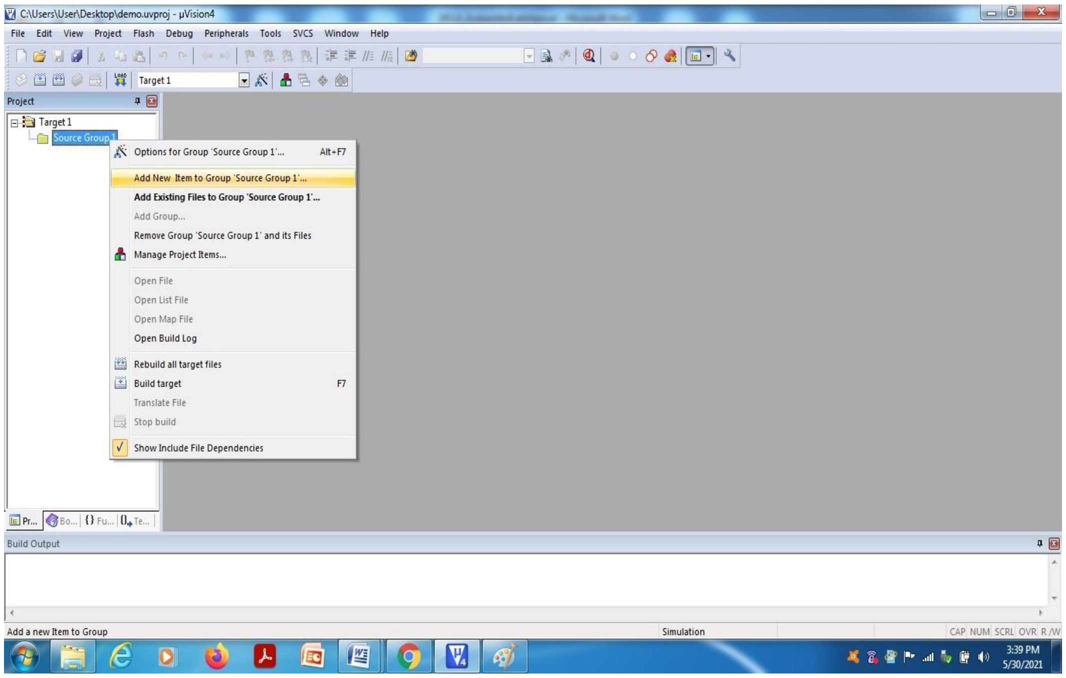

C:\Users\User\Desktop\demo.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Build Output

µVision

Copy 'startup_LPC17xx.s' to Project Folder and Add File to Project ?

Yes No

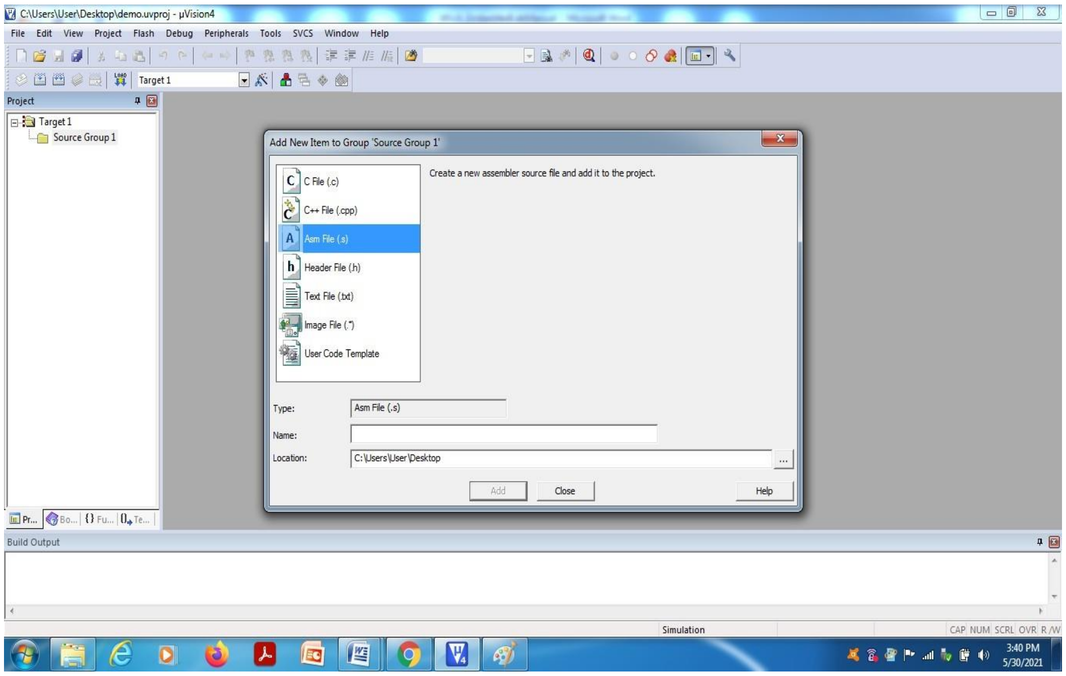
6. In the project window, right click on source and select Add new item to group “source group1”.


C:\Users\User\Desktop\demo.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Target 1
Source Group 1
Build Output
Simulation

Options for Group 'Source Group 1'... Alt+F7
Add New Item to Group 'Source Group 1'...
Add Existing Files to Group 'Source Group 1'...
Add Group...
Remove Group 'Source Group 1' and its Files
Manage Project Items...
Open File
Open List File
Open Map File
Open Build Log
Rebuild all target files
Build target F7
Translate File
Stop build
Show Include File Dependencies

Add a new Item to Group

7. Select ASM file and give name of the file with .s extension and press ADD.



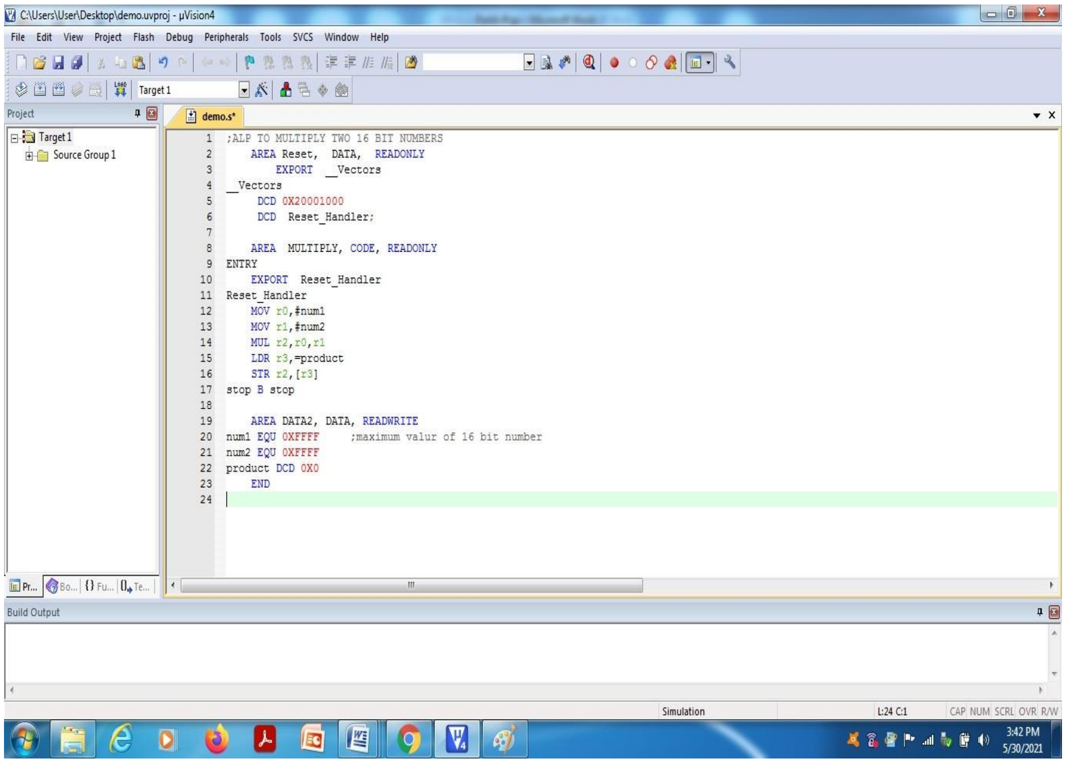
Create a new assembler source file and add it to the project.

Type: Asm File (.s)

Name:

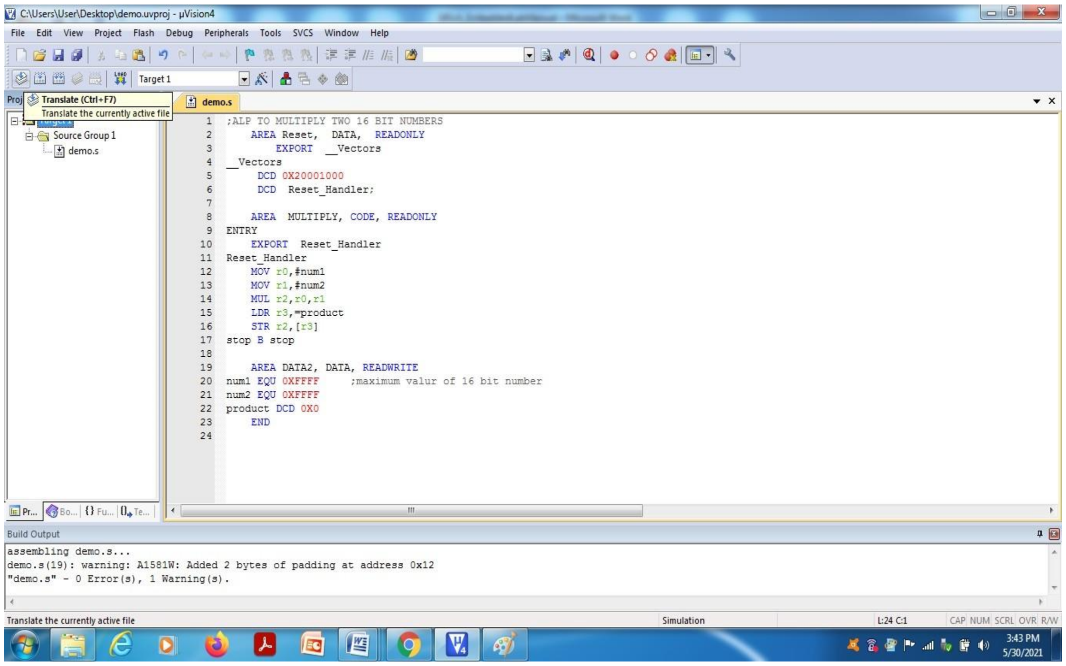
Location: C:\Users\User\Desktop

8. Type the program in the editor space and save.


C:\Users\User\Desktop\demo.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Target 1
Source Group 1
demo.s*
Build Output
Simulation
L:24 C:1
CAP NUM SCRL OVR R/W
3:42 PM
5/30/2021

```
;ALP TO MULTIPLY TWO 16 BIT NUMBERS
    AREA Reset,  DATA,  READONLY
        EXPORT  __Vectors
__Vectors
     DCD 0X20001000
     DCD  Reset_Handler;

    AREA  MULTIPLY, CODE, READONLY
ENTRY
    EXPORT  Reset_Handler
Reset_Handler
    MOV r0,#num1
    MOV r1,#num2
    MUL r2,r0,r1
    LDR r3,=product
    STR r2,[r3]
stop B stop

    AREA DATA2, DATA, READWRITE
num1 EQU 0XFFFF     ;maximum valur of 16 bit number
num2 EQU 0XFFFF
product DCD 0X0
    END

```

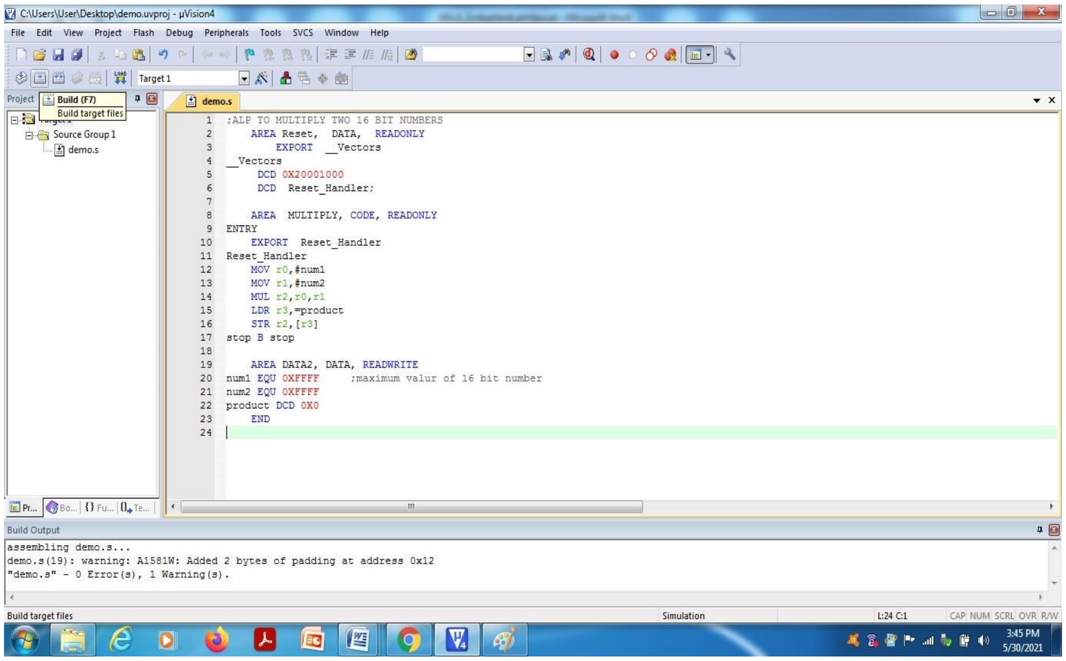
9. Translate the program by select the icon from tool bar or from menubar, also Check for errors and warnings in the bottom window.


C:\Users\User\Desktop\demo.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Translate (Ctrl+F7)
Translate the currently active file
Source Group 1
demo.s
Build Output
Translate the currently active file
Simulation
L:24 C:1
CAP NUM SCRL OVR R/W
3:43 PM
5/30/2021

```
;ALP TO MULTIPLY TWO 16 BIT NUMBERS
    AREA Reset,  DATA,  READONLY
        EXPORT  __Vectors
__Vectors
     DCD 0X20001000
     DCD  Reset_Handler;

    AREA  MULTIPLY, CODE, READONLY
ENTRY
    EXPORT  Reset_Handler
Reset_Handler
    MOV r0,#num1
    MOV r1,#num2
    MUL r2,r0,r1
    LDR r3,=product
    STR r2,[r3]
stop B stop

    AREA DATA2, DATA, READWRITE
num1 EQU 0XFFFF     ;maximum valur of 16 bit number
num2 EQU 0XFFFF
product DCD 0X0
    END
```

```
assembling demo.s...
demo.s(19): warning: A1581W: Added 2 bytes of padding at address 0x12
"demo.s" - 0 Error(s), 1 Warning(s).
```

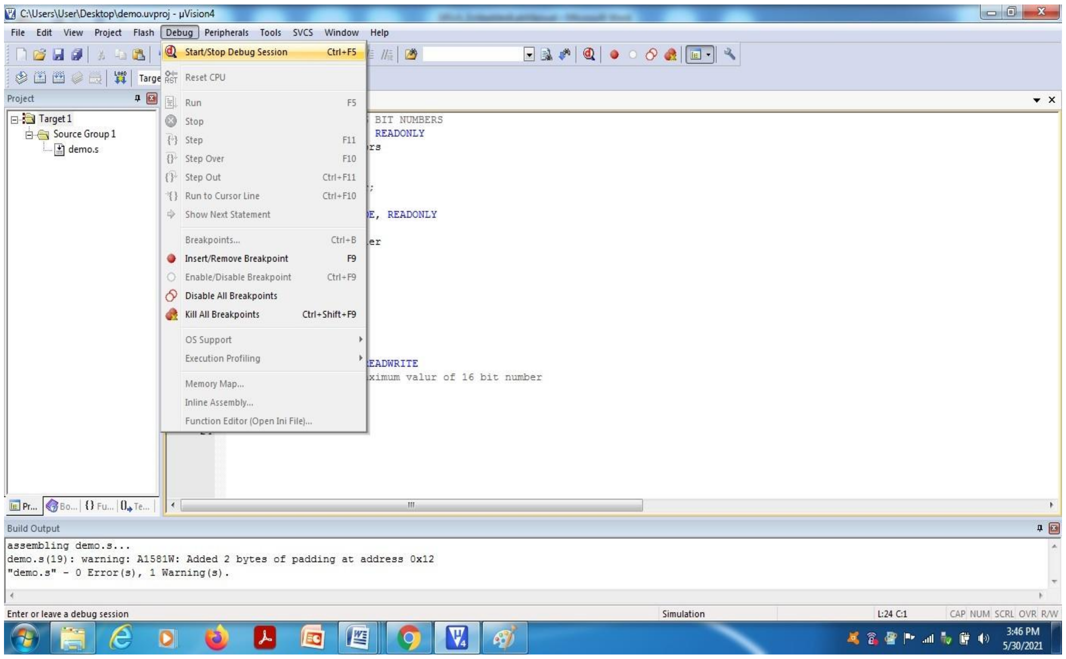
10. If no error, Select “Build” icon from tool bar or from menubar.



```
;ALP TO MULTIPLY TWO 16 BIT NUMBERS
    AREA Reset,  DATA,  READONLY
        EXPORT  __Vectors
__Vectors
     DCD 0X20001000
     DCD  Reset_Handler;

    AREA  MULTIPLY, CODE, READONLY
ENTRY
    EXPORT  Reset_Handler
Reset_Handler
    MOV r0,#num1
    MOV r1,#num2
    MUL r2,r0,r1
    LDR r3,=product
    STR r2,[r3]
stop B stop

    AREA DATA2, DATA, READWRITE
num1 EQU 0XFFFF     ;maximum valur of 16 bit number
num2 EQU 0XFFFF
product DCD 0X0
    END
```

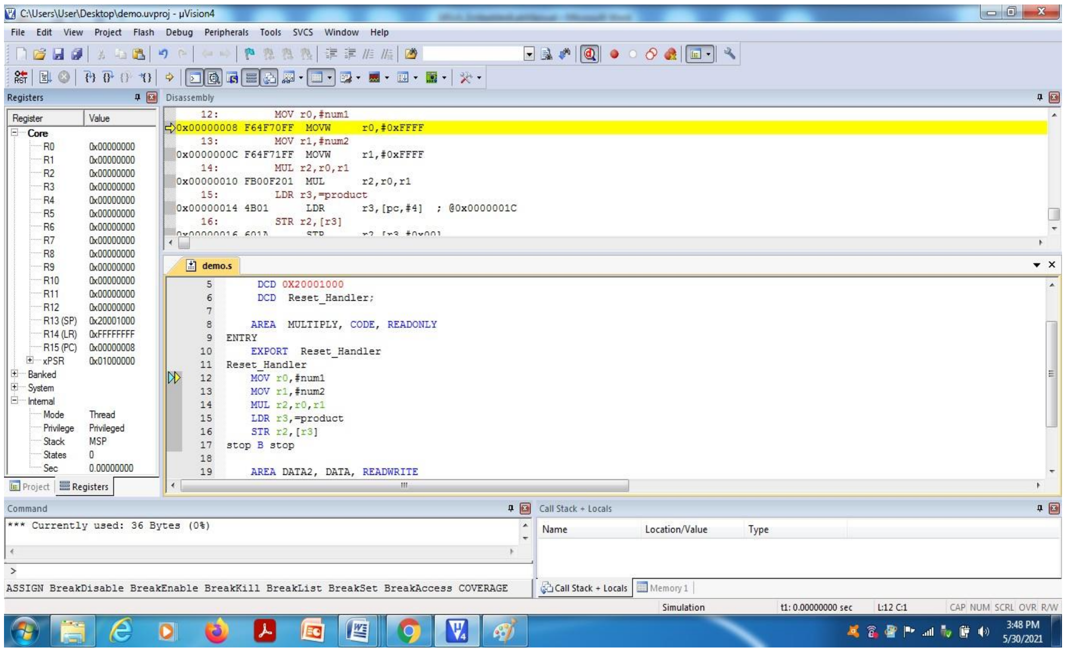
```
assembling demo.s...
demo.s(19): warning: A1581W: Added 2 bytes of padding at address 0x12
"demo.s" - 0 Error(s), 1 Warning(s).
```

11. Start the debug session from Menubar.


C:\Users\User\Desktop\demo.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Start/Stop Debug Session Ctrl+F5
Reset CPU
Run F5
Stop
Step F11
Step Over F10
Step Out Ctrl+F11
Run to Cursor Line Ctrl+F10
Show Next Statement
Breakpoints... Ctrl+B
Insert/Remove Breakpoint F9
Enable/Disable Breakpoint Ctrl+F9
Disable All Breakpoints
Kill All Breakpoints Ctrl+Shift+F9
OS Support
Execution Profiling
Memory Map...
Inline Assembly...
Function Editor (Open Ini File)...
Project
Target 1
Source Group 1
demo.s
Build Output
Enter or leave a debug session
Simulation
L:24 C:1
CAP NUM SCRL OVR R/W
3:46 PM
5/30/2021

```
assembling demo.s...
demo.s(19): warning: A1581W: Added 2 bytes of padding at address 0x12
"demo.s" - 0 Error(s), 1 Warning(s).
```

12. PressOK



Register	Value
Core	
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20001000
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x00000008
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	0
Sec	0.00000000

```
    12:         MOV r0,#num1
0x00000008 F64F70FF  MOVW     r0,#0xFFFF
    13:         MOV r1,#num2
0x0000000C F64F71FF  MOVW     r1,#0xFFFF
    14:         MUL r2,r0,r1
0x00000010 FB00F201  MUL      r2,r0,r1
    15:         LDR r3,=product
0x00000014 4B01      LDR      r3,[pc,#4]  ; @0x0000001C
    16:         STR r2,[r3]
```

```
      DCD 0X20001000
      DCD  Reset_Handler;

    AREA  MULTIPLY, CODE, READONLY
ENTRY
    EXPORT  Reset_Handler
Reset_Handler
    MOV r0,#num1
    MOV r1,#num2
    MUL r2,r0,r1
    LDR r3,=product
    STR r2,[r3]
stop B stop

    AREA DATA2, DATA, READWRITE
```

```
*** Currently used: 36 Bytes (0%)
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE
```

13. Press function key F11 or select “step” option under Debug menu for single step execution and verify the outputin register window/Memorywindow/xPSR.

Program:

ALP TO MOVE A BLOCK OF DATA FROM CODE TO RAM MEMORY

Method1:

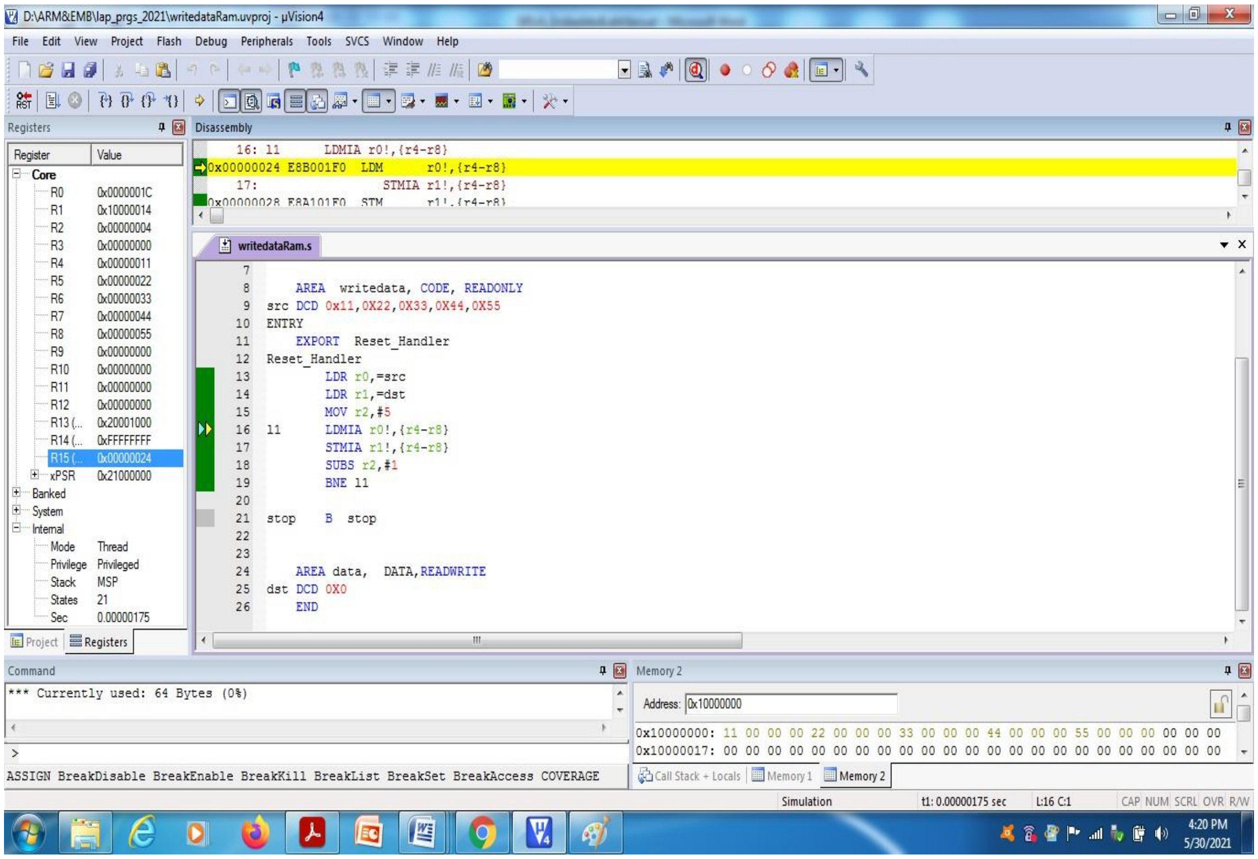
```
AREA Reset, DATA, READONLY
EXPORT   Vectors
  Vectors
DCD 0X20001000
DCD Reset_Handler;

AREA  writedata,  CODE,  READONLY  src  DCD  0x11,0X22,0X33,0X44,0X55
ENTRY
EXPORT Reset_Handler Reset_Handler
LDR r0,=src LDR r1,=dst MOV r2,#5
l1      LDR r3,[r0],#4
STR r3,[r1],#4 SUBS r2,#1 BNE l1

stop    B stop


AREA data, DATA,READWRITE
dst     DCD  0X0 END
```

Method 2:

;**ALP TO MOVE A BLOCK OF DATA FROM CODE TO RAM MEMORY-USING LDM and STM INSTRUCTIONS(MULTIPLE DATA TRANSFER)**

```
AREA Reset, DATA, READONLY
EXPORT   Vectors
  Vectors
DCD 0X20001000
DCD Reset_Handler;
AREA  writedata,  CODE,  READONLY  src  DCD  0x11,0X22,0X33,0X44,0X55
ENTRY
EXPORT Reset_Handler Reset_Handler

LDR r0,=src LDR r1,=dst MOV r2,#5
l1      LDMIA r0!,{r4-r8} STMIA r1!,{r4-r8} SUBS r2,#1
BNE l1

stop    B stop


AREA data, DATA,READWRITE
dst     DCD  0X0 END
```

Result:
INPUT: 00000011h,00000022h,00000033h,00000044h,00000055h. OUTPUT at dst : 00000011h,00000022h,00000033h,00000044h,00000055h.



Register	Value
Core	
R0	0x0000001C
R1	0x10000014
R2	0x00000004
R3	0x00000000
R4	0x00000011
R5	0x00000022
R6	0x00000033
R7	0x00000044
R8	0x00000055
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (...	0x20001000
R14 (...	0xFFFFFFFF
R15 (...	0x00000024
xPSR	0x21000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	21
Sec	0.00000175

```
    16: l1       LDMIA r0!,{r4-r8}
0x00000024 E8B001F0  LDM       r0!,{r4-r8}
    17:                  STMIA r1!,{r4-r8}
0x00000028 E8A101F0  STM       r1!,{r4-r8}
```

```

    AREA  writedata, CODE, READONLY
src DCD 0x11,0X22,0X33,0X44,0X55
ENTRY
    EXPORT  Reset_Handler
Reset_Handler
        LDR r0,=src
        LDR r1,=dst
        MOV r2,#5
l1      LDMIA r0!,{r4-r8}
        STMIA r1!,{r4-r8}
        SUBS r2,#1
        BNE l1

stop    B  stop


    AREA data,  DATA,READWRITE
dst DCD 0X0
    END
```

```
*** Currently used: 64 Bytes (0%)
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE
```

```
Address: 0x10000000
0x10000000: 11 00 00 00 22 00 00 00 33 00 00 00 44 00 00 00 55 00 00 00 00 00 00
0x10000017: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

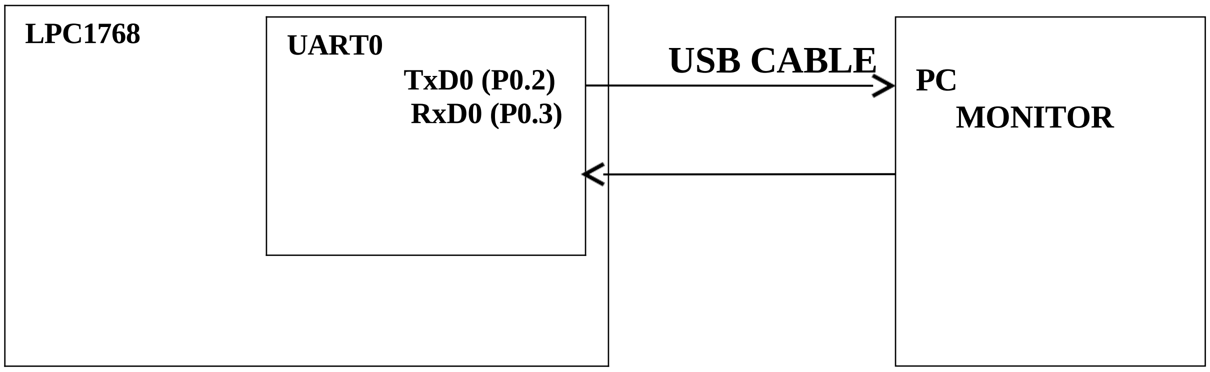
14. Display Hello World Message Using Internal UART

AIM: To display “Hello World” message using Internal UART

APPARATUS:
1. PC with Windows 10, 64-bit OS.
2. Keil µVision4 Software.

PROCEDURE:

Connection Details:



UART Registers:
The below table shows the registers associated with LPC1768 UART.

Register	Description
RBR	Contains the recently received Data
THR	Contains the data to be transmitted
FCR	FIFO Control Register
LCR	Controls the UART frame formatting(Number of Data Bits, Stop bits)
DLL	Least Significant Byte of the UART baud rate generator value.
DLM	Most Significant Byte of the UART baud rate generator value.

UART Register formats or configuration:
FCR (FIFO Control Register):
LPC1768 has inbuilt 16byte FIFO for Receiver/Transmitter. Thus it can store 16-bytes of data received on UART without overwriting. If the data is not read before the Queue(FIFO) is filled then the new data will be lost and the OVERRUN error bit will be set.
FCR

31:8	7:6	5:4	3	2	1	0
RESERVED	**RX TRIGGE R**	**RESERVE D**	**DMA MOD E**	**TX FIFO RESET**	**RX FIFO RESET**	**FIFO ENABLE**

LSR (Line Status Register):
The is a read-only register that provides status information of the UART TX and RX blocks.
LSR Format:

31:8	7	6	5	4	3	2	1	0
Reserved	**RXFE**	**TEMT**	**THRE**	**BI**	**FE**	**PE**	**OE**	**RDR**

TER (Transmitter Enable register): This register is used to Enable/Disable the transmission
TER Format:

31:8	7	6-0
Reserved	**TXEN**	**Reserved**

Baudrate Calculation
LPC1768 generates the baud rate depending on the values of DLM,DLL.
Baudrate = PCLK/ (16 * ((256 * DLM) + DLL) * (1+ DivAddVal/MulVal))
where, DLM=0, DLL= , (DivaddVal/MulVal)=0.

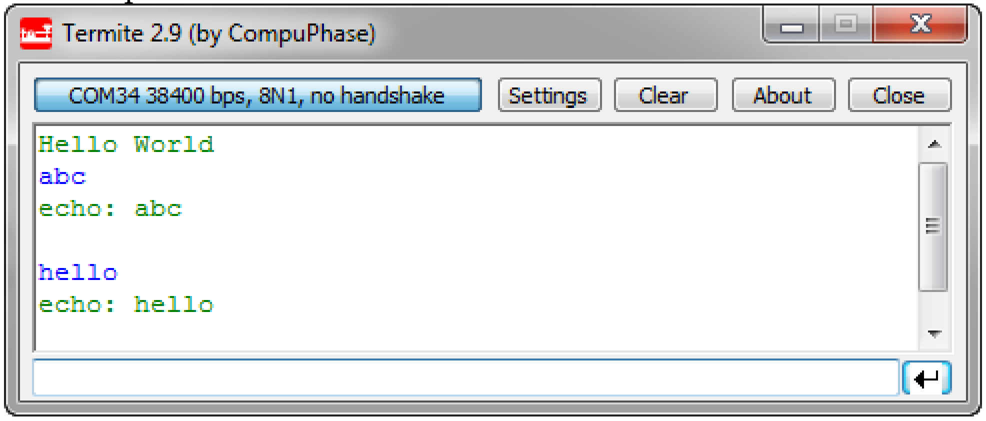
<u>Steps for Configuring UART0</u>
Below are the steps for configuring the UART0.
1. Configure the P0.2 and P0.3 as first alternate function UART0 function using PINSEL0 register. 2.Configure the FCR for enabling the FIFO and Reset both the Rx/Tx FIFO.
3. Configure LCR for 8-data bits, 1 Stop bit, Disable Parity and Enable DLAB.
4. Calculate the DLM,DLL values for required baudrate from PCLK.
6. Update the DLM,DLL with the calculated values(i.e DLM=0;DLL=163).
7. Finally clear DLAB to disable the access to DLM,DLL.
After this the UART will be ready to Transmit/Receive Data at the specified

baudrate, by sending the string character by character.

PROGRAM:

```
#include<lpc17xx.h>
void U0Write( char txdata)
{
while(!(LPC_UART0->LSR & 0x20));
LPC_UART0->THR=txdata;
}

void initUART0(void)
{
LPC_PINCON->PINSEL0 =(1<<4)|(1<<6); LPC_UART0->LCR=0x83;
LPC_UART0->DLL=163; LPC_UART0->DLM=0; LPC_UART0->FCR =0x7;
LPC_UART0->FDR=0x0; LPC_UART0->LCR = 0x03;
}

int main(void)
{
charmsg[]= "Hello World"; int i=0;
initUART0(); for(i=0;msg[i];i++)
{Program:
#include<lpc17xx.h>
void U0Write( char txdata)
{
while(!(LPC_UART0->LSR & 0x20));
LPC_UART0->THR=txdata;
}

void initUART0(void)
{
LPC_PINCON->PINSEL0 =(1<<4)|(1<<6); LPC_UART0->LCR=0x83;
LPC_UART0->DLL=163; LPC_UART0->DLM=0; LPC_UART0->FCR =0x7;
LPC_UART0->FDR=0x0; LPC_UART0->LCR = 0x03;
}

int main(void)
{
charmsg[]= "Hello World"; int i=0;
initUART0(); for(i=0;msg[i];i++)
{
U0Write(msg[i]);

}
}
U0Write(msg[i]);

}
}
```

RESULT:
The output is observed on the monitor.


Termite 2.9 (by CompuPhase)
COM34 38400 bps, 8N1, no handshake
Settings
Clear
About
Close

```
Hello World
abc
echo: abc

hello
echo: hello
```

15. Interface a Stepper motor and rotate it in clock wise and anti-clock wise direction

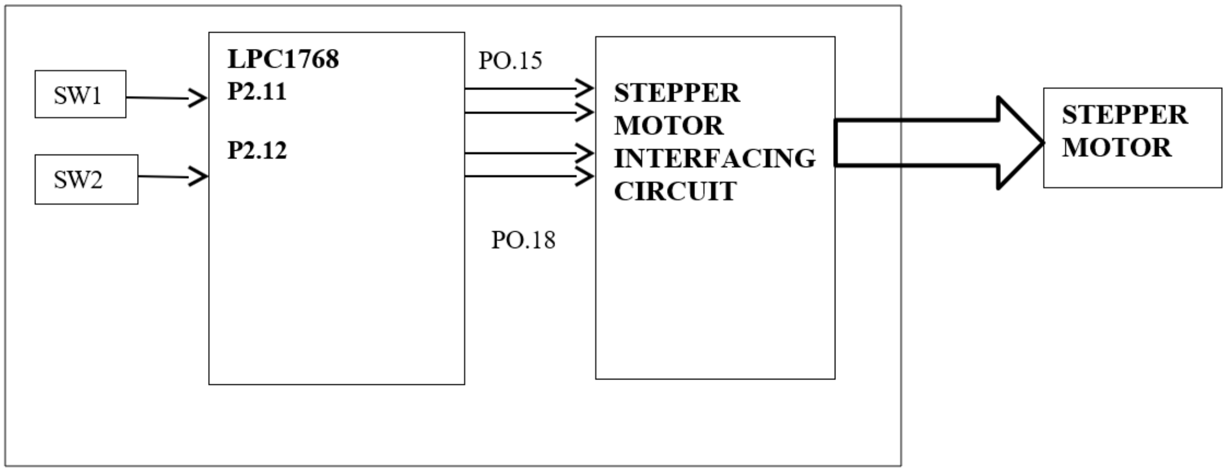
AIM: To Interface a Stepper motor and rotate it in clock wise and anti-clock wise direction.

AIM: To display "Hello World" message using Internal UART

APPARATUS: 1. PC with Windows 10, 64-bit OS.

2. Keil µVision4 Software.

CONNECTION DETAILS:

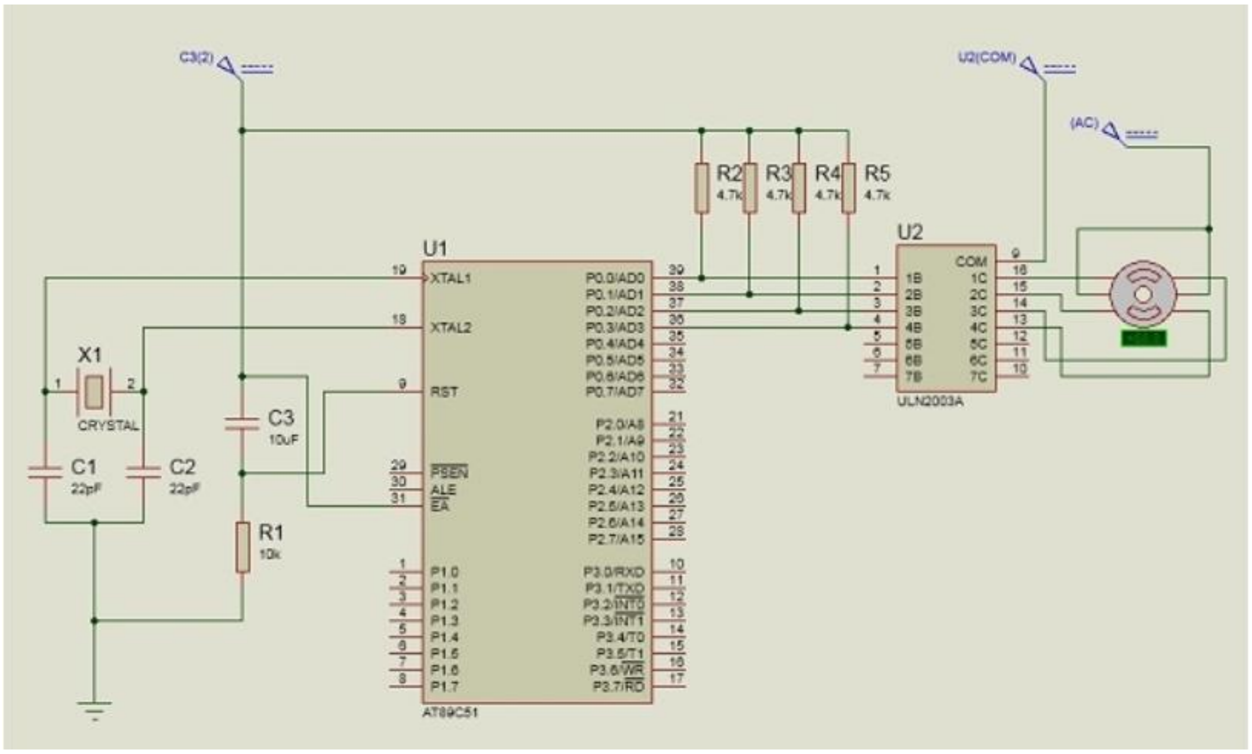


1. Configure the Port 0 and Port 2 as GPIO.
2. Configure the Port 2 in input direction and Port 0 in output direction.
3. Read the status of the switch 1. If it is pressed, set the direction as 0 for clock wise rotation.
4. Else read the status of switch 2. If it is pressed, set the direction as 1 for anticlock wise rotation.
5. If the direction is 0, send the data to energize the stepper motor coils in a sequence A-B-C-D else in D-C-B-A sequence.
6. Insert an appropriate delay between energizing two consecutive coils.
7. Repeat from steps 3 unconditionally.

PROGRAM:

```
#include<lpc17xx.h>
#define SW1 11
#define SW2 12
void delay(unsigned int x)
{
unsignedinti,j; for(i=0;i<x;i++)
{ for(j=0;j<90000;j++);
}
}
int main(void)
{
unsignedint direct;
LPC_PINCON->PINSEL0=0X00000000;                    LPC_PINCON-
>PINSEL1=0X00000000;      LPC_PINCON->PINSEL4=0X00000000;
LPC_GPIO0->FIODIR=0xFFFFFFFF;                        LPC_GPIO2-
>FIODIR=0X00000000;
LPC_GPIO0->FIOCLR=0X00078000;// CLEAR P0.15 TO p0.18
while(1)
{
if(!((LPC_GPIO2->FIOPIN>>SW1)& 0X1))
{
while(!((LPC_GPIO2->FIOPIN>>SW1) & 0X1));
direct=1;
}
else if(!((LPC_GPIO2->FIOPIN>>SW2) & 0X1))
{
while(!((LPC_GPIO2->FIOPIN>>SW2) & 0X1));
direct=0;
}
if(direct==1)
{
LPC_GPIO0->FIOPIN=0X00008000;
delay(15);
LPC_GPIO0->FIOPIN=0X00010000;
delay(15);
LPC_GPIO0->FIOPIN=0X00020000;
delay(15);
LPC_GPIO0->FIOPIN=0X00040000;
delay(15);
}
else
{
LPC_GPIO0->FIOPIN=0X00040000;
delay(15);
LPC_GPIO0->FIOPIN=0X00020000;
delay(15);
LPC_GPIO0->FIOPIN=0X00010000;
delay(15);
LPC_GPIO0->FIOPIN=0x00008000;

delay(15);
}
}
}
```

RESULT:
The is observed on the virtual simulator.


C3(2)
U2(COM)
(AC)
R2 4.7k
R3 4.7k
R4 4.7k
R5 4.7k
U1
U2
X1
CRYSTAL
C1 22pF
C2 22pF
C3 10uF
R1 10k
XTAL1
XTAL2
RST
PSEN
ALE
EA
P0.0/AD0
P0.1/AD1
P0.2/AD2
P0.3/AD3
P0.4/AD4
P0.5/AD5
P0.6/AD6
P0.7/AD7
P2.0/A8
P2.1/A9
P2.2/A10
P2.3/A11
P2.4/A12
P2.5/A13
P2.6/A14
P2.7/A15
P3.0/RXD
P3.1/TXD
P3.2/INT0
P3.3/INT1
P3.4/T0
P3.5/T1
P3.6/WR
P3.7/RD
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
AT89C51
COM
1B
2B
3B
4B
5B
6B
7B
1C
2C
3C
4C
5C
6C
7C
ULN2003A