

Lab Code: 20EC504/J01A 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:<u>bec.principal@becbapatla.ac.in</u> Web:www.becbapatla.ac.in

Contents

S.No.	Title of the Experiment
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

Bapatla Engineering College :: Bapatla (Autonomous)

<u>Vision</u>

- 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.

<u>Mission</u>

- 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.

<u>Mission</u>

- 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)															
Le	ctures		: 2 H	lours/W	Veek	Tu	torial	:	0 Hoi	ur/W	eek	Pra	ctical	:	2 Hoi	ır/Week
CI	E Marks	3	: 30			SE	E Marks	:	70			Cre	edits	:		3
Pr	Pre-Requisite : Microprocessors, and Microcontrollers.															
	The metaphotobood, and increasing of the															
Co	Course Objectives: Students will learn how to															
	\succ	Lea	rn basi	c design	n and	archi	tectural co	nce	pts of e	mbed	lded s	ystem	s.			
	~	Und	lerstand	d the d	conce	pts o	of Real-Ti	me	Opera	ting	Syste	ms ai	nd pro	vide	the sch	eduling
	~	Alg	orithms	s					•	Ŭ	•		•			C
	\triangleright	Fan	niliarize	e with t	he fur	ndam	entals of p	reva	alent IP	P-Core	e: AR	M Co	rtex M	3/M 4	& Desig	gn of an
	, ,	emt	edded	system	using	AR	M Cortex F	roc	essor							
	\succ	Be	able to	use the	e insti	ruction	on set of A	ARN	A Corte	ex M.	3/M4	proce	ssor ar	nd ex	plain the	ALP's
		USII	ig AR	vi proce	essor											
C		- 4		A. C	·1!		·•	41.	1 .			-1-1 -	4 -			
	ourse O		omes:	After s	tuayi	ng ti	nis course,		e stude	nts w	V111 DE		to	1	11.1	
		Des	cribe (umeren	nt me	thou of D	lologies ar		approac	cnes 1	in the	desig	gn of e	mbe	adea sys	stems
	202	Ana	alyze t	he con	cepts	01 K	eal-lime	Op	erating	g syst	ems a	and sc	nedul	$\lim_{n \to \infty} P$	Algorith	ns.
C	.03	nius cim	strate th	<u>A DM</u>	$\frac{res, b}{Drag}$	asic a	architecture	$\frac{1}{2}$ and	a mem	ory n	1anage	ement Vision	unit of	AK	M Proces	sors
(CO4	Simulate ARM Programming models using Keil μ Vision for different embedded														
		лрг	meano	115.												
-	Manni	ng (of Cou	rse Or	itcom	ies v	vith Prog	ran	n Outc	ome	s & P	rogr	am Sr	ecifi	ic Outco	mes
			2 000	2.50 00			PO ²	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	
									1		1		I			I

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 highperformance 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.

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 States:

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.

RTX51 Tiny Task Management Functions:

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
RTOSn'';
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);
}
```

ECE Dept.

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:

File Edit View Project Flash De	bug Peripherals Tools SVCS Window Help
🗋 😂 🖬 🕔 🕹 🚵 🥙	○ (···· · ·) ● 魯 魯 魯 ○ (達 達 // // // // // // // // // // // //
🥩 🏥 🕮 🧼 拱 🖉 🖉 Target 1	🖃 🔊 🚡
Project 📮 🔝	exploring_keil.c embeddedcpgm_exp2.c task_creation_deletion_exp3.c*
B Target 1 B Source Group 1 B D Task_creation_deletion_e L TrtS1tny.h	<pre>01 #include <rtxsitny.h> 02 long counter0; 03 long counter0; 04 long counter1; 04 long counter2; 05 long counter3; 06 long counter3; 06 long counter3; 07 job0 () task_0 (; 08 os_create_task (0); 09 os_create_task (1); 10 os_create_task (3); 11 os_create_task (3); 12 os_create_task (3); 12 os_create_task (3); 13 while (1) (14 counter0++; 15) </rtxsitny.h></pre>
✓ m → m Pro ③ Pro ③ Pru [0, Tc] Suid Output	<pre>16 } 17 jobl () _task_ 1 { 18 whle (i) { 19 counterl++; 20 } 21 } 22 job2 () _task_ 2 { 23 whle (i) { 24 counterl++; 24 c</pre>

Sompling task_creation_detecton___xpotett linking... Program Size: data=50.1 xdata=0 const=0 code=695 "eosdd&may" - 0 Error(s), 0 Warning(s).



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:

	pherals Tools SVCS Window Help
🗋 🐸 🖬 🕼 🕹 🖧 🧐 🍽 🖛	-> 隆 🎄 🍇 連 連 ////////////////////////////////
😂 🖾 🕮 🥪 🗮 🛛 🚟 🕇 Target 1	🖃 🔊 📇 🔁
Project 🗜 🖬 🚺 e	xploring_keil.c embeddedcpgm_exp2.c task_creation_deletion_ea
□ Target 1 □ Source Group 1 □ Image: Target 1 Image: Target 1 <	<pre>xploing_kentembeddeddqgm_exp2.ttask_dreation_detention_ee include <rtxsitny.h> ong counter1; ong counter2; ong counter3; ob0 ()task_ 0 { os_create_task (1); os_create_task (2); os_create_task (3); while (1) { counter0++; } } ob1 ()task_ 1 { while (1) { counter1++; } } ob2 ()task_ 2 { while (1) { counter2++; } }</rtxsitny.h></pre>
4 III + 24	counter2++;
E Pro Bo Fu	
Build Output	
<pre>compliing round_robin_scheduling_ex linking Program Size: data=43.1 xdata=0 cor "eosdd8may" - 0 Error(s), 0 Warning</pre>	nst=0 code=656 g(s).
℃ C:\Users\Admin\Documents\cosdd8may.uvproj - µVision4 File Edit View Project Flash Debug Peripherals Tools SVC □ 😂 I Image: Superind state Image: Superind state Image: Superind state Image: Superind state	S Window Help
C:\Users\Admin\Documents\cosdd8may.uvproj - µVision4 File Edit View Project Flash Debug Peripherals Tools SVC Image: Signal S	5 Window Help 2 译译///////////////////////////////////
C<\Users\Admin\Documents\eosdd8may.uvproj - µVision4	S Window Help (译译///////////////////////////////////
✓ C:\Users\Admin\Documents\eosdd8may.uvproj - µVision4 File Edit View Project Flash Debug Peripherals Tools SVC Image: Solution of the state of the s	S Window Help 注 译 評 // // // // // // // // // // // // /
✓ C C Users Admin Decuments Version4 File Edit View Project Flash Debug Peripherals Tools SVC Image: Solution of the state o	S Window Help F F H H K M F F K K K K K K K K K K K K K K K K
V: C:Users\Admin\Documents\cosddBmay.uvproj-µVision4 File Edit View Project Flash Debug Peripherals Tools SVC Image: Strain St	S Window Help

*** 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
ł
while(1)
ł
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:

The cuit view Project Plash Debug Peri	pherals Tools SVCS Window Help
🗋 😂 🛃 🥔 🕹 📾 🚵 🍤 (> 🖛)	-> 🏴 🎕 🏨 谭 谭 /// //// 🙆 💽 [
😂 🕮 😂 🤐 🗮 🛛 🗱 🖾 Target 1	🖃 🌋 📇 🕾
Project 🕈 🖬 🚺 e	exploring_keil.c embeddedcpgm_exp2.c task_creation_deletion_ex
□ Target 1 □ Source Group 1 □ Image: Target 1 Image: Target 1 Image: Targ	<pre>kploing_kentembeddeddpgm_exp2.ttask_dreahon_detention_ee include <rtx51tny.h> ong counter0; ong counter1; ong counter3; ob0 () _task_ 0 { os_create_task (1); os_create_task (2); os_create_task (3); while (1) { counter0++; } } ob1 () _task_ 1 { while (1) { counter1++; } ob2 () _task_ 2 { while (1) { counter2++; } </rtx51tny.h></pre>
* <u> </u>	3
■ Pro {} Fu 0, Te	
Build Output	
<pre>compiling round_robin_scheduling_es linking Program Size: data=43.1 xdata=0 cos "ecsdd8may" - 0 Error(s), 0 Warning "C(Users\Admin\Documents\cosdd8may.uvproj-µVision4</pre>	xp1.c nst=0 code=656 g(s).
File Edit View Project Flash Debug Peripherals Tools SVC	S Window Help
19 # # 19 (4 + 4) つ (20 4 4) 🖏 🖬 🗋 🗍	x 達達/// () 🖉 💽 🔍 🖉 🔍 🔍 🔍 🔍 🖉 🖉 🖉
🕷 🗟 🖶 🗗 🖓 🖓 🖉 👘 👘	💷 • 🔯 • 🔜 • 🔜 • 🎌 • 🖳
Registers 📮 🖬 Disassembly	
Register MAIN:	
Regs r0 r1 r2 r2 r	R0, #RTX_TASKSP(0x29) @R0, SP(0x81) A.#0x03
Regs C:0x095E 7629 HOV r0 C:0x09C0 A681 MOV r1 r2 TASK_COMMUNICATION.c Task	R0, #RTX_TASKSP(0x29) @R0, SP(0x81) A. #0x03 sk_communication_exp7.c round_robin_scheduling_exp4.c tasksynchronization.c task_synch
Image: Second and a light provide the second and and a light provide the second and a light p	R0, #RTX_TASKSP (0x29) @R0, SP (0x81) A. #0x03 sk_communication_exp7.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 concurrencyrelated 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.
- 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.
 - 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	CONTRACTOR AND A DESCRIPTION OF THE OWNER OF T	
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help		
□●日日 1 日本 1 日 1 H		
Project 🛛 🕫		
E Pr.,. 6 Bo.,. {} Fu.,. 0, Te.,.		
Build Output		# 📔
		*
*		E E
		CAP NUM SCRL OVR R /W
(?) 📋 💪 🔉 🔞 🖪 📧 🖉 🧿) 🔣 🕺 🖉 🗠 🖬 🕻	3:28 PM 5/30/2021

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


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

Vision4		and the interest strengt		
Create New Project	may begreen test into means may	×		
🔾 🗢 🔳 Desktop 🔸	• 47	Search Desktop	参 ① ● ○ ◇ @ 🔟 • 🔦	
Organize - New felder		88 - 0		
organize • New Tolder		85 ° U		
🖈 Favorites 🍈 🧊 Lib	raries 🔒 Objects			
E Desktop	megroup 🥼 practice programs			
😹 Downloads 👘 🧸 Us	er 🕌 printout_june8th			
强 Recent Places 🛄 🌉 Co	mputer 🔒 ResultDigit			
 OneDrive Ne 	twork 🔐 Seshu projects			
act 🔐 act	ivity 🍶 studentDetails			
🥽 Libraries 🥼 Ke	erthana 📕 studentTestPapers			
🖹 Documents 🌙 Lai	FinalExamTT 🎳 vlsiLab			
🎝 Music 🌙 Lis	ings			
Distance				
File name: demo		-		
Save as type: Project Files (*.)	vproj)	•		
(a) 15da Caldara	0	Save Cancel		
Hide Folders	L	Jave Cancer		
Pr 🔞 Bo {} Fu 0, Te				
uild Output				a (f
				F
				CAP NUM SCRL OVR R
			× 3	🔐 🏴 📶 🧤 🎁 🌒 3:33 PM
				5/30/2021

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

C:\Users\User\Desktop\demo.uvproj - µVision4	And an and a start of the start	
File Edit View Project Flash Debug Peripherals Tool:	SVCS Window Help	
□ 2 3 3 4 4 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	き き 進 近 版 🙋 💿 😨 泳 参 🔍 🎍 〇 〇 🔗 🏨 🔟 🔍	
📀 🕮 🥔 🧱 💥 Target 1 💿 🔊	± 5 ♦ @	
Project # 🖬		
	Select Device for Target Target 11 CPU Vendor: NOP Device for Target 71 Search: [pc1788 Totler: ARM Search: [pc1788 Device for Paraget 72 AM 224 Cote+M3 Microsoftwale with MPU CPU dock up to 1004 + Dystem Target 74 Description: Difference Difference <td< th=""><th></th></td<>	
Build Output		
		<u>^</u>
		CAP NUM SCRL OVR R/W
🚱 📜 🙆 💟 🔞 🖪		💐 🚳 🔐 🏴 .all 🧞 🛱 🌒 3:35 PM 5/30/2021

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

ClUsers/User/Desktop/demouvproj - µVision4	
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help	
○ □ □ ② □ ③ □ [¥] Target1 ● K 品 告 ◆ 働	
Project 9 🖸	
VVision VVision Copy 'startup_LPC17xxs' to Project Folder and Add File to Project ? Ves No	10
	-
	NUM SCRI OVR R AM
🛞 🚆 🖉 🗴 🏟 📕 🔤 🖳 🥥 🔣 🛷	₩ ^{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	\den	no.uvproj - µVision4	and a		A Married Wood		
File Edit View Proj	ect	Flash Debug Peripherals Tools SVCS Window	Help				
🗋 🗃 🗑 🖉 🚳		四日 日本 日本 日本 日本 日本	1 // 🗄 📔 🏄	- 3	s 🖉 🔍 💿 🔿 🔗 🍓 🔲 🔍		
🔗 🕮 🕮 🥔 🗮	諡	Target 1 💽 🔊 📥 🖶 🚸 🍥					
Project		* 🖬					
E-I Target 1	N.	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					
	111	Rebuild all target files					
	i Xi	Build target F7					
	88 193	Translate File Stop build					
	1	Show Include File Dependencies					
נו Pr (א המושים ביותר ב	0.,						
Build Output							÷ 🖾
							^
4							P
Add a new Item to Group	_				Simulation		CAP NUM SCRL OVR R /W
🕐 📋 I	e	0 🕹 🖊 🗖 🖉		đ		× 6	📲 🏴 .all 🧞 📴 🌒 3:39 PM 5/30/2021

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

🖸 C/Users/User/Desktop/demo.uvproj - µVision4	
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help	
◎ ↓ ◎ ↓ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○	
② Ш 四 @ □ [¥] Target	
Project 🛛 🖉	
🕀 📷 Target 1	
Leng Source Group 1 Add New Item to Group 'Source Group 1'	
C File (a) C + File (cpp) A file file (f) P Header File (f) P Head	
Build Output	a 🖬
	*
s Simulation CAD MIRE S	
	3:40 PM 5/30/2021

8. Type the program in the editor space and save

C:\Users\User\Desktop\demo.uvpro	- µVision4		
File Edit View Project Flash	ebug Peripherals Tools SVCS Window Help		
🗋 🖸 🖬 🖉 🕹 🖧 💐	◇ ◇ ◇ 魯 魯 魯 魯 澤 淳 准 振 🙋 💿 🗟 ⋪ 🔍 🔶	ି 🔗 🍓 🔲 🔍	
🔗 🏥 🖽 🥔 🗮 💥 Target	🐱 🖈 📇 🍬 🎰		
Project 🛛 🚨	demo.s*		▼ ×
🖃 🛅 Target 1	1 ;ALP TO MULTIPLY TWO 16 BIT NUMBERS		
🛓 🧰 Source Group 1	2 AREA Reset, DATA, READONLY		
	3 EXPORT Vectors		
	4Vectors		
	6 DCD Reset Handler;		
	7		
	8 AREA MULTIPLY, CODE, READONLY		
	9 ENTRY		
	10 EXPORT Reset_Handler		
	12 MOV n0 from 1		
	13 MOV r1. #num2		
	14 MUL r2,r0,r1		
	15 LDR r3,=product		
	16 STR r2, [r3]		
	17 stop B stop		
	18 19 ADEA DATAS DATA DEADWRITE		
	20 num1 FOU OXFFFF ::maximum valur of 16 bit number		
	21 num2 EQU OXFFFF		
	22 product DCD 0X0		
	23 END		
	24		
E Pr (3 Bo {} Fu 0. Te	< [
Build Output			
¢			F
		Simulation	L:24 C:1 CAP NUM SCRL OVR R/W
🕐 📋 E 🕻	I 🥹 🖪 🖾 🧐 🔣 🛷		🂐 🗿 🖉 🏴 all 🦆 🛱 🌒 3:42 PM 🔤

EMBEDDED SYSTEM & DESIGN(20EC504/JO1A)

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.



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



11. Start the debug session from Menu bar.

EMBEDDED SYSTEM & DESIGN(20EC504/JO1A)

ECE	Dept.
-----	-------

J C:\Users\User\Desktop\demo.uv	proj - µVision4		Mill Delasted attend Texas Texas	
ile Edit View Project Flash	Debug Peripherals Tools SV	CS Window	Help	
🗋 🧉 🖉 🖉 🕹 🕰 🚺	Start/Stop Debug Session	Ctrl+F5	🖡 🕼 🖉 🔍 🗣 🏨 🗢 🔿 🚓 🛄 🖓	
😒 🏦 🕮 🥔 📇 🕎 🏹 Tare	REAT Reset CPU			
roject 🛪 🖬				• 3
Target 1	Run .	1.5	DTT MINUERDS	
B - Source Group 1	W Stop		READONLY	
demo.s	(1) Step	FIL	rs	
	() Step Over	Chili Ell		
	() Due to Concerting	Chi+TII		
	Chau Mart Chatamant	CUITFFID		
			E, KERDURLI	
	Breakpoints	Ctrl+B	ez	
	Insert/Remove Breakpoint	F9		
	C Enable/Disable Breakpoint	Ctrl+F9		
	Disable All Breakpoints			
	Kill All Breakpoints	Ctrl+Shift+F9		
	OS Support	+		
	Execution Profiling		EADWRITE	
	Memory Map		ximum valur of 16 bit number	
	Inline Assembly			
	Function Editor (Open Ini File)		
			1	
	_			
E Pr 6 Bo () Fu U., Te	1.			
uild Output				0
essembling demo.s	81W: Added 2 bytes of pa	dding at a	address 0x12	
demo.s" - 0 Error(s), 1	Warning(s).			
				F
nter or leave a debug session			Simulation L:24 C:1 CAP	NUM SCRL OVR RA
				3:46 PM
				5/30/2021

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\I	Desktop\demo.uvp	vroj - µVision4	
File Edit View	Project Flash	Debug Peripherals Tools SVCS Window Help	
n 🐸 🖬 🖉	X 🕹 🚨	🤊 🗠 🗇 🥐 整 巻 巻 準 準 准 准 🙆 💿 🖬 🐼 🕢 🕞 🐼 🥔 🕘 💿 🔗 🇶 💷 🔍	
計 🗟 🔕	0 6 6		
Registers	P 📧	Disassembly	4 🖬
Register	Value	16: STR r2,[r3]	*
Core		200000016 601A STR r2, [r3, #0x00]	
RO	0x0000FFFF	Slokososis ErfE B 0x0000018	
B1	0x0000FFFF	0x000001A 0000 DCW 0x0000	
R2	0×1000000	0x000001C 0000 DCW 0x0000	
R4	0x00000000	0x0000001E 1000 DCW 0x1000	
	0x00000000	0x0000020 0000 MOVS r0,r0	1.1
R6	0x00000000	0x0000022 0000 MOVS r0,r0	*
	0x00000000	4x0000004 0000 M0VS +0 +0	F.
R8	0x00000000		
R9	0x00000000	📩 demo.s	÷×
R10	0x00000000	5 DCD 0X20001000	
R11	0x00000000	6 DCD Reset_Handler;	
R12	0x00000000	7	
P14 (IP)	0x20001000	8 AREA MULTIPLY, CODE, READONLY	
R14 (Ch)	0x00000018	9 ENTRY	
P PSB	0x01000000	10 EXPORT Reset Handler	
E Banked	010100000	11 Kest Handler	=
B System		12 NOV EV, el unit	
E-Internal		14 MUL r2 r0 r1	
Mode	Thread	15 LDB r3.=product	
Privilege	Privileged	16 STR r2, [r3]	
Stack	MSP	17 stop B stop	
States	13	18	
Sec	0.00000108	19 AREA DATA2, DATA, READWRITE	-
🔚 Project 🔛 Re	gisters	e	•
Command		🗸 🙆 Memory 2	# 🖸
*** Currentl	y used: 36 B	ytes (0%) Address [0:1000000	- 0
		+ Plates. Jartoudou	L 🗌 🗌
<		0x10000000: 01 00 FE FF 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
ASSIGN Break	Disable Bread	KEnable BreakKill BreakList BreakSet BreakAccess COVERAGE 🕼 Call Stack + Locals 🔠 Memory 1 📰 Memory 2	
		Simulation 11: 0.0000108 sec L:17 C:1 CAP NUM S0	CRL OVR R/W
@		🖸 🙆 🖊 🚾 🕎 🕅 🧭 🔛 🧭	3:57 PM
			COLORD CARDON CARD

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 μ Vision4 Software.

PROCEDURE:

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



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



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

🕅 µVision4	and the second second	adapted to a second sec	
Create New Project	an fast 201 manue ma		
🚱 🗢 🗖 Desktop 🔸	• 49 Search Desktop		
Organize 🔻 New folder	\$\$ •	0	
Favorites Dexitop Downloads Decent Places OneDrive Dubraies Documents Documents	Objects Objects protout_june8th Besult/ligit student/Details student/Details student/Details		
Music Listings File name Save as type Project Files ("uvproj)			
Hide Folders	Save	cd	
Build Output			a 🖬
4			* *
			CAP NUM SCRL OVR R/W
🛞 📋 🙆 🔉 🔞		🧭 🚽 🖉	🂐 🖀 🔮 🏴 💷 🧞 🛱 🐠 3:33 PM 5/30/2021

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

C:\Users\User\Desktop\demo.uvproj - µVision4		A Descent attack states		
File Edit View Project Flash Debug Peripherals Tools	SVCS Window Help			
	🎗 🎊 🗊 課 //注 //注 🙆 👘	- 1 🖉 🖉 🔍 🖉 🖉 🐨		
🔗 🕮 🕼 🥥 🧱 🖉 🕅 Target 1 💿 🌾 🖌	8 8 * @			
Project 🏨 🔀				
	Select Device for Target 1: CPU Vendor: NXP Device: LPC1768 Toole: ARM Search: bc1758 □····································	Description: ATM 32bit Cottex-M3 Monocontroller with MPU, CPU clock up to 100 512.6 on-chip Flash RMM with enhanced Reah Memory Accelerator, In-System Programmy (SP) and V-hip-Robotic Integrations Betherent 1010 MAC with RHMI Interface and dedicated RMA. USS 20 full-speed Device controller and Hon/ DTG controller with DMC CAN 20 BW this ocharantis. Controller and Hon/ DTG controller with DMC RHMI RHMI Interface and edicated RMA. USS 20 full-speed Device controller and Hon/ DTG controller with DMC CAN 20 BW this ocharantis. Conc WHR 51, one with UMC Meet Interface Three IZC senial Interfaces. Three SPVSSF senial Interfaces. IZS Interf Three IZC senial Interfaces. Three SPVSSF senial Interfaces. IZS Interface Concentrol PVM for three phase Motor control. Quaditume Ficoded Watchdog Imer, Real TTm Colock with optional Battery Badou. System Tick Timer, Reattery Internet ILL, Wakeeu Internet, Cormol Coystal conclutor. 4MHz Internet RL Cocollitor, PLL. e III III OK Cancel Het		
Build Output			_	4 🖸
i. i				* *
			CAP	NUM SCRL OVR R /W
			💐 🖧 🔤 🍽 🖬 🤴	(*) 5/30/2021

EMBEDDED SYSTEM & DESIGN(20EC504/JO1A)

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	
Project a B	
Copy 'startup_LPC17xx,' to Project Folder and Add File to Project ? Ve: No	
suia output	u 🖬
	<u></u>
	-
	F.
	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	ALL COMPARING AND ALL AND A	
File Edit View Project Flash Debug Peripherals	Tools SVCS Window Help	
D 🗳 🗟 🖉 3 & 2 & 2 9 P 4 + 4 9	整 巻 巻 孝 孝 唐 振 🍎 🚽 🚽 🚽 🍕 🔍 🔍 🔹 🔗 🗶 💷 🔍	
🔗 🕮 🖉 🥪 📇 🛛 🐺 🛛 Target 1 🕢 🕞 🛦	ŝ 🛔 🗟 🚸 🎰	
Project 📮 🖬		
E Target 1		
- Source Group 1	Add New Item to Group 'Source Group 1'	
	Crite (c) Crite (cp) A fair fie (z) Test Fie (zc) Test Fie (zc) User Code Template Type: Arm Fie (.3) Hame: Locator: C:\User\User\User\User\User\User\User\User	
Duild Output		
Duild Output		4 🖬
*		F
📀 📋 🙆 🛛 🔞 🖡	Simulation 🛛 🖉 🖉 📲 🗸 🎖 📽 🕨 🖬 🏷	3:40 PM 5/30/2021

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

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

C:\Users\User\Desktop\demo.uvp	j - μVision4	There are a second s	
File Edit View Project Flash	Debug Peripherals Tools SVCS Window Help		
🗋 🞯 🖬 🕼 🛔 🖉 🔤 👘	- > > > 🥐 雅 雅 雅 澤 澤 川 川 🙆 🧶 🖉	• 🔍 🖕 · 🔗 🍕 💷 • 🔧	
🖗 🏦 🕮 🥔 📇 🛛 🎬 🛛 Targe	1 💽 🔊 🛔 🗟 🗇 🎯		
Project 🛛 🗣 🔯	demo.s*		▼ X
B ≧ Target1 d	1 ;ALF TO MULTIFLY TWO 16 BIT NURBERS AREA Reset, DATA, READONLY 3 EXFORTVectors 4Vectors 5 DCC 0X2001000 6 DCD Reset_Handler; 7 8 AREA MULTIFLY, CODE, READONLY 9 ENTRY 10 EXFORT Reset_Handler 11 Reset_Handler 12 MOV r0,fnuml 13 MOV r1,fnum2 14 MUL r2,f0,fnl 15 LDR r3,-product 16 STR r2,fs] 19 AREA DATA2, DATA, READWRITE 20 numl EQU 0XFFFF ;maximum value of 16 bit number 21 num2 EQU 0XFFFF ;maximum value of 16 bit number 22 product DCD 0X0 24		
Build Output	۲ m.		, a @
4). F
		Simulation	L:24 C:1 CAP NUM SCRL OVR R/W
🕘 📋 🙆 🛛) 🔞 🖪 🖻 🖉 🥥 🔣 🚿 🗌		342 PM 👔 🔐 🕪 .al Խ 🛱 🕪 3342 PM 5/30/2021

EMBEDDED SYSTEM & DESIGN(20EC504/JO1A)

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



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



EMBEDDED SYSTEM & DESIGN(20EC504/J01A)

11. Start the debug session from Menubar.

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help	
📔 🖀 🖉 😹 🖕 🧟 🔤 🕲 Start/Stop Debug Session Ctri+55 🧯 🎼 🔯 📃 🕞 🗟 🥀 💽 🖉	
😢 🕮 🥔 兴 🕌 Targe 🎎 Reset CPU	
Project 🗣 📓 🖹 Run F5	▼ ×
B Targel Stop DIT NUMBERS	
→ Successful () Step Fill Automatic	
O step over Fil	
() [*] step Out cm++ii	
() Kun to Cursor Line Currerzu	
- JIOW NEW Statement E, NEW STATEMENT	
Breakpoints Ctri+B er	
InserURemove Breakpoint F9	
C Enable/Disable Breakpoint Ctrl+F9	
O Disable All Breakpoints	
Kiii Ali Breakpoints Ctri+Snitt+19	
OS Support	
Execution Profiling	
Memory Map ximum valur of 16 bit number	
Inline Assembly	
Function Editor (Open Ini File)	
III I'm 😵 Bon, M Plun Ug Ban Y	
Build Output	u 🗎
assembling demo.sl.) demo.sl.9: warning: A1581W: Added 2 bytes of padding at address 0x12	^
"demc.s" - 0 Error(s), 1 Warning(s).	
	F.
Enter or leave a debug session Li24	C1 CAP NUM SCRL OVR R/W
🚱 🚞 💪 🛛 🔞 📕 🖾 🖉 🚫 🕅 🚿 🔺 😵	2 📭 .al 🦆 🛱 🌒 3:46 PM 🎽

12. PressOK

C:\Users\User\	Desktop\demo.uvp	xroj - µVision4	
File Edit View	v Project Flash	Debug Peripherals Tools SVCS Window Help	
0 📬 🖬 🖉	1 8 4 6	9 ~ ~ ◆ 巻 巻 巻 淳 定 店 図 ● ◆ ◆ ● ● ◆ ◆ ● ● ◆ ◆	
0+ 15 Q	0.0.0		
RST 🖽 🤍	0000		
Registers	÷ 🖬	Disasembly	ц 19
Register	Value	12: MOVID, fnumi Physnonese Reference and the second	^
E Core R0 R1 R2 R3 R4	0x00000000 0x00000000 0x00000000 0x000000	13 MOV T1,4num2 0x000000C F6471FF Mx0VM 21,40xFFFF 14: MUL x2,x0,x1 0x000000C F800F201 MUL x2,x0,x1 15: LDR x3,-product 0x00000014401 LDR x3,-product	
R5	0x0000000	16: STR r2, [r3]	
R7	0x00000000	4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4	
R8	0x00000000		
R9	0x00000000		• *
R11 R13 (SP) R13 (SP) R14 (LR) R15 (PC) ⊕ _RPSR ⊕ Banked ⊕ System ⊡ Internal Mode Nde Stack Stacks Stacks Stacks Sec	0x0000000 0x20001000 0x20001000 0x2FFFFFF 0x0000008 0x01000000 Thread Philleged MSP 0 0 0.00000000	 CDD Reset_Handler; AREA MULTIPLY, CODE, READONLY PITRY EXFORT Reset_Handler Reset_Handler NOW ro, frumat MOW ro, frumat MOW ro, frumat MULTIPLE, c0, r1 Eds FS, =product SIR r2, (r3) AREA DATA2, DATA, READWRITE 	E
🛅 Project 🛛 🧱 Re	egisters	em	+
Command		a 🖬 Cali Stack + Locals	# 🖬
*** Currentl < >	y used: 36 B	ytes (0%) Name Location/Value Type	
NOJION DIGSK	bisable brea		
		Simulation TE:00000000 See E12 CI CAP	NUM SCRE OVR R/W
	l e	0 🔞 🛃 🖾 🦉 🔣 🛷 🛛 🔺 🍇 🖉 🖬 🛝 🕯	5/30/2021

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 11 ADD r0,r0,r1 ADD r1,r1,#1 SUBS r3,#1 BNE11 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)



11. TO FIND THE 1'S AND O' 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.



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



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

EMBEDDED SYSTEM & DESIGN(20EC504/J01A)

JuVision4		A DESCRIPTION OF THE OWNER OF THE		
Create New Project	August fat it's many ma-	x		
🙆 🔍 📼 Desktop 🔸	- 49 Searc	Desktop	* Q • • • • & III • ×	
	1.5.0	22		
Organize • New folder		33 • 🔮		
🔆 Favorites 🏠 🧊 Librarie	i Objects			
E Desktop	oup 🌙 practice programs			
🐌 Downloads 👘 🔒 User	🍶 printout_june8th			
🔛 Recent Places 📃 📑 Compu	ter 🌙 ResultDigit			
OneDrive	: 🍶 Seshu projects			
🍶 activity	studentDetails			
词 Libraries 🥔 Keertha	na 🤰 studentTestPapers			
Documents LabFina	ExamTT 🍶 visiLab			
🎝 Music 🌙 Listings				
Distance T				
File name: demo		•		
Save as type: Project Files (*.uvpro)	-		
0		Caral		
Hide Folders	3	Cancer		
Pr. Bo. D Fun Da Tem				
Suild Output				
und output				· · ·
4				E State Stat
				CAP NUM SCRL OVR R /M
			× 2 B 1	3:33 PM
			4 6 Y F	5/30/2021

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		and instant attack when a		
File Edit View Project Flash Debug Peripherals Tool	s SVCS Window Help			
○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○	魯 魯 連 連 旭 順 🙆	- 🖬 🔬 🖉 🥥 🖉 🖷 -	2	
📀 🗂 😂 🥘 🕎 🔤 Target 1 💽 🔊	1 7 × 10			
Project 📮 🔟				
	Select Device for Target 'Target 1'			
	Ventor: NOP Device: LPC1788 Todiet: ARM Search: [os1768 □ ♥ NXP □ ♥ NXP □ ♥ IPC1765	Description: ARM 32-bit Cotex-M3 Microcontroller with MPU, CPU clock up 15.8 dis on-chip Rash ROM with enhances firsh Memory Accel (%) System Programming (157) and 1%-bipolication Programming (%) System Programming (157) and 1%-bipolication Programming (%) System Programming (%) System (%) System (%) System (%) System (%) System (%) System (%) System (%) System (%) System (%) System (%) System (%) System (%) System (%) System (%) System (%) System (%) System (%) System (%) System (%) Sys	to 100W ▲ Herder, MPJ, WPB, Streffac, Photo, Photo, Photo, Carefolder, P.	
		OK Cancel	Help	
Build Output	-			10
				*
				-
*				
🚱 📜 E o o o 🖪		X	a 🕯 🕨 🖬 🕷	3:35 PM 5/30/2021

EMBEDDED SYSTEM & DESIGN(20EC504/JO1A)

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

😰 C(Users/User/Desktop/demo.ursproj - µVision4	
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help	
◇ □ □ ◇ □ 第 Target ● K ▲ 各 金	
Project 🛛 🗣 🖻	
Vision Vision Copy Startup_IPCI7xx* to Project Folder and Add File to Project ? Ves No	
Build Output	# 🖂
	^
4	÷
CA	NUM SCRL OVR R /W
🛞 🗒 🖉 🛛 🥹 📕 🖻 🖳 🧿 🔣 🚿 🛛 🔺 😵 Marke	3:37 PM 5/30/2021

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



EMBEDDED SYSTEM & DESIGN(20EC504/J01A)

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

🕜 CAUsers/User/Desktop/demo.uvproj - µVision4	
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help	
1 🥶 😹 参 () () () () () () () () () (
📀 🛗 🛗 🥪 📇 🙀 Target 1 🕢 🛣 📥 🐟 🎰	
Project 🛛 🗶	
E Target1	
Add New Item to Group 1 Add New Item to Group 'Source Group 1'	
C File (c) C File (c) C + File (cpp) A John File (s) Header File (h) Tot File (cd) Wingo File (7) Will User Code Template Type: Ann File (s) Name: Location: C [/Users/User/Desktop 	
Build Output	# 🖬
4	*
Simulation	CAP NUM SCRI OVR R AW
🛞 🚞 🖉 o 🔞 🖪 🖻 🖉 🥥 💟 🛷	🂐 🗟 🖉 🏴 💷 🤯 🗑 🕴 3:40 PM

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

C:\Users\User\Desktop\demo.uvpro	- µVision4		
File Edit View Project Flash	ebug Peripherals Tools SVCS Window Help		
🗋 🗋 🗃 🗿 🛛 🕹 🛍 🖉	○ ◇ ◇ ● 意思 数 津 津 川 20	ឮ ● ○ 🏨 🔳 🔦	
🔗 🕮 🕮 🥔 📇 💥 Target	💌 🖈 🖶 🖘 🍩		
Project 🛛 🖗 🔟	demo.s*		▼ ×
B ≧a Target 1 ⊕ ∰ Source Group 1	<pre>1 .ALP TO MULTIPLY TWO 16 BIT NUMBERS 2 ABEA Reser, DATA, READONLY 4 _Vectors 4 _Vectors 5 DED 002000000 6 DED Reser_Handler; 7 AREA MULTIPLY, CODE, READONLY 9 ENTRY 10 EXFORT Reser_Handler 12 MOV r0, 4nunl 13 MOV r1, 4nunl 14 MUL r2, 40, r1 15 LDR r3,-product 16 SIR r2, [r3] 17 stop 8 stop 18 19 AREA DATA2, DATA, READWRITE 20 UNEFFF : rmaximum value of 16 bit number 21 numl EQU 0XFFFF : rmaximum value of 16 bit number 22 product DED 0X0 24 24 24 24 24 24 24 24 24 24 24 24 24</pre>		
	4 m		
Build Output			4 🖬
			A
4			
		Simulation	L:24 C:1 CAP NUM SCRL OVR R/W
📀 📃 🙆 🕻) 🥹 🖪 🖻 🖉 🥥 🔣 🚿		🂐 🚳 🚰 🏴 💷 🧞 🛱 🌒 3:42 PM 5/30/2021

EMBEDDED SYSTEM & DESIGN(20EC504/JO1A)

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



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



ECE Dept.

EMBEDDED SYSTEM & DESIGN(20EC504/JO1A)

11. Start the debug session from Menubar.



12. PressOK

C:\Users\User\[Desktop\demo.uvp	roj - µVision4	
File Edit View	Project Flash	Debug Peripherals Tools SVCS Window Help	
D 🐸 🖬 🖉	X & B	🤊 🏱 今日 🕐 熟 熟 連 連 進 版 🙆 💽 🖬 🐨 💽 🍇 🌮 🞯 🖕 🔹 🔗 🍓 💷 🔦	
8 🗉 🚳	0 0 0		
Registers	P 🔝	Disassembly	÷ 🖬
Register	Value	12: MOV r0, #num1	*
E Core R0 R1 R2 R3	0x00000000 0x00000000 0x00000000 0x000000	CPCM00000000000000000000000000000000000	
	0x00000000	0x0000014 4B01 LDR r3,[pc,#4] ; @0x000001C	
R6	0x00000000	16: STR 22, [r3]	*
R7	0x00000000		Þ
R8	0x0000000	(i) demos	• ×
R10	0x00000000		
R11	0x00000000	6 DCD Reset_Handler;	
R13 (SP) R14 (LR) R15 (PC)	0x20001000 0xFFFFFFF 0x00000008 0x01000000	8 AREA MULTIPLY, CODE, READONLY 9 ENTRY 10 EXFORT Reset_Handler	
 Banked ⇒ System ⇒ Internal Mode → Privilege → Stack 	Thread Privileged MSP	1 keret_manlast 12 MOV rd, smult 13 MOV rd, smult 14 Mov rd, smult 15 Strate, smult 16 STR sd, producet 17 Toro B, storigit	E
States	0	18	
Sec	0.00000000	19 AREA DATA2, DATA, READWRITE	-
Project Re	gisters	<u>.</u>	•
Command		🗸 🔝 Call Stack + Locals	
*** Currentl	y used: 36 By	rtes (0%) ^ / Name Location/Value Type	
ASSIGN Break	Disable Break	Enable BreakKill BreakList BreakSet BreakAccess COVERAGE	
		Simulation t1: 0.00000000 sec L12 C1 CAP	NUM SCRL OVR R/W
📀 📜	e	0 🤞 🎝 📧 🖳 🕥 🔣 🛷 🔍 🕹	4) 3:48 PM 5/30/2021

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 O' 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 11 ADD r2,#1 B 12 11 ADD r1,#1 12 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\la	p_prgs_2021\oneszeros	uvproj - µVision4	
File Edit View	Project Flash Debu	ig Peripherals Tools SVCS Window Help	
🗋 😂 🖬 🥔	36 - 🕰 🔊 P		
詩 🗉 🕥 🕇	9 0 0 0 0		
Registers		4 Disassembly	# 63
Register	Value	→ Ox000002C 4E02 LDR r6,[pc,#8] : @0x00000038	
Com	Value	27: STR r1, [r5]	
BO	0~0000000	0x0000002E 6029 STR r1,[r5,#0x00]	
B1	0x00000004	28: STR r2, [r6]	
B2	0x0000001C	0x00000030 6032 STR r2,[r6,#0x00]	
	0x00000000	29: stop B stop	
R4	0x00000000	CX00000032 E7FE B 0x00000032	
R5	0x10000000	0x0000034 0000 DCW 0x0000	
R6	0x10000001	0x00000036 1000 DCW 0x1000	
	0x00000000	0x0000038 0001 DCW 0x0001	
R8	0x00000000	0x000003A 1000 DCW 0x1000	-
R9	0x0000000	0x000003C 0000 MOVS P0.P0	
R10	0x0000000	discourse bood Movs F0.F0	
R11	0x00000000		
R12	0x00000000		- ×
R13 (SP)	0x20001000		
R14 (LR)	OxFFFFFFFF	23 12 SUBS r3,#1	~
R15 (PC)	0x00000032	24 BNE loop	
± xPSR	0x61000000	25 LDR r5, mones	
Banked		26 LDR r6, =zeros	
+ System		27 SIR 11, [15]	
internal	221 32	28 STR 12, [16]	100
Mode	Thread	29 Stop B Stop	100
Privilege	Privileged		
Stack	MSP	JI AREA DAIAI, DAIA, READWRITE	*
III Project Reg	isters	*	•
Command		A 🔝 Memory 2	4 6
Running with	Code Size Limit:	32K	01
Load "D:\\ARM	sEMB\\lap_prgs_2	021\\oneszeros.axf" Address: UK 1000000	
		0x10000000: 04 1C 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00
*** Restricte	d Version with 3	2768 Byte Code Size Limit * 0x10000017: 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00
4		0x1000002E: 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 -
ASSIGN BreakD	isable BreakEnab	le BreakKill BreakList BreakAccess COVERAGE	
		Simulation 11-0 0002833 rac 1/39 C1 C42 NUMA S2	CPI OVP PAW
	1 4	Simulation Li 0.0002833 Sec Lis CI Chr Hom S	ALC OTH FOW
			4:10 PM

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.



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



EMBEDDED SYSTEM & DESIGN(20EC504/J01A)

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

🕅 µVision4		the second	station front and	
Create New Project	Image Response Sum 2413 and			
😋 🗢 🗖 Desktop 🔸		 ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓		
Organize 👻 New folder		88 -	0	
	Libraries Dóbjects Homegroup pratice programs Ure printou june8th Computer ResultDigit Network Seehu projecta activity StudentDetails activity studentDetails activity VisiLab LabFinalExamTT VisiLab			
Hide Folders	_	Save Cance		
■ Pr ③ Bo {} Fu 0., Te				
Build Output				4 🖸
4				A • •
🛞 📜 🙆 (D 🕹 🖪 🖾	🖭 🌔 💟 4	ý	Image: CAP NUM SCRE OVR R.W Image: CAP NUM SCRE OVR R.W </td

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

C:\Users\User\Desktop\demo.uvproj - µVision4		candidation of the second second	
File Edit View Project Flash Debug Peripherals Tools	SVCS Window Help		
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	🛝 務 津 津 //: //:: 🏙	- 34 🖉 🔍 💿 🔿 🖓 🛄 - 🌂	
🔗 🗂 😂 🥘 💥 Target 1 💽 🌾 d	65 * 60		
Project 📮 🖾			
	Select Device for Target 'Target 1'	X	
	CPU Vendor: NOP Device: LPC1768 Todeet: ARM Search: [pc1768 ⊡-ØNDP - CI [PC1768	Description: ARN 32-bit Cottex-M3 Microcontroller with MPU, CPU clock up to 100% + 513-86 anchip Rash ROM with enhanced Rash Memory Accelerator, System Fingermanning (ISP) and H-Valcation Horganning (IAP), 16-86 RMM, Neuted Vectorel Internut Controller, Behmen 10/100 Acchim RMI Internation and declared DMM. USB 2.044 speed Device controller and Hosti /OTG controller with DMM, USB 2.044 speed Device controller and Hosti /OTG controller with DMM. USB 2.044 speed Device controller and Hosti /OTG controller with DMM. USB 2.044 speed Device controller and Hosti /OTG controller with DMM. CMA 2.08 with two chamels, Faux URATA, Faux end that III, Mock and Hosti /OTG Prove 2.06 Faux III, Devin Relative and the Control Could State Speed Neurol Control FWM for three spheres Motor control. Quadrature Encoder. Visitanding Time, Repetitive Internut, Time, Borow Judit Betary Social Cystem Control WMH for three spheres Motor control. Quadrature Encoder. Visitanding Time, Repetitive Internut, Time, Borow Judit Betary Social Cystem Control WMH for three spheres Motor control. Quadrature Encoder. Visitanding Time, Repetitive Internut, Time, Borow Judit Betary Social Cystem Control WMH for three spheres Motor control. Quadrature Encoder. Visitan USA: The Immed Repetitive Internut, Controller Cystem Contack with this internet Rocottador, PLL.	
E Pr 8 Bo 8 Fu 0. Te		OK Cancel Help	
Build Output			
7			CAP NUM SCRL OVR R./W
📀 📋 🙆 👂 📕		Ø	🂐 👸 🖉 🏴 .all Խ 庁 🅪 3:35 PM 🗾 5/30/2021

EMBEDDED SYSTEM & DESIGN(20EC504/JO1A)

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



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



EMBEDDED SYSTEM & DESIGN(20EC504/J01A)

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

CAUsers/User/Desktop/demouvproj - µVision4	3
e Edit View Project Flash Debug Peripherals Tools SVCS Window Help	
1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	
2 🗄 🗇 🥪 🗄 🗱 Target 1 🕢 🖌 📥 🕾 🚸	
eet 🧧 🖬	
Add New Item to Group Source Group 1'	
	4 🖸
Simulation CAP NUM SCRL OVR	R/W
🔋 📜 🧀 🛯 🧶 🖪 🖉 🚫 🔣 🛷	1

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

C:\Users\User\Desktop\demo.uvpro	oj - µVision4	- 0 X
File Edit View Project Flash	Debug Peripherals Tools SVCS Window Help	
🗋 D 📴 🖬 🕼 🛔 🕹 🛍 🖌 🤻	🤊 🖻 🖗 🕸 微微 幸幸 進 版 🙋 🕢 見 🌲 🥙 🔍 🗕 🔿 🔗 🍓 💷 🔍	
🔗 🗈 🛗 🥔 🗮 🕌 🎇 Target		
Project 🛛 🗣 🔯	demo.s*	▼ X
🖃 🛅 Target 1	1 ;ALP TO MULTIPLY TWO 16 BIT NUMBERS	
🗄 🛅 Source Group 1	2 AREA Reset, DATA, READONLY	
	3 EXPORTVectors	
	5 VECTORS	
	6 DCD Reset Handler:	
	7	
	8 AREA MULTIPLY, CODE, READONLY	
	9 ENTRY	
	10 EXPORT Reset_Handler	
	11 Reset Handler	
	12 MOV FU, HILMII 13 MOV FU, HILMII	
	14 MUL r2,r0,r1	
	15 LDR r3,=product	
	16 STR r2,[r3]	
	17 stop B stop	
	18	
	19 AREA DATAZ, DATA, KEADWRITE	
	20 Humi Ego WAFFFF , maximum valut ol 10 bit Humber	
	22 product Dot 0X0	
	23 END	
	24	
<u>■ Pr</u>	· · · · · · · · · · · · · · · · · · ·	•
Build Output		P 🔁
		~
		-
4		F
	Simulation L24 C1 CAP NUM	SCRL OVR R/W
		3:42 PM
		5/30/2021

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.uvpr	j - µVision4	a - House I had	
File Edit View Project Flash	Debug Peripherals Tools SVCS Window Help		
🗋 🗃 🖬 🗿 🕹 🕰 🗖	○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○	à 🕫 🍳 🖕 ୦ 🔗 🏨 💷 🔍	
🐼 🕮 🧭 🥘 💥 🛛 Targe	1 💽 🔊 📥 🗟 🚸 🍏		
Proj 🤣 Translate (Ctrl+F7)	demo.s		▼ ×
Translate the currently active fi	1 TALE TO MULTIPLY TWO 16 BIT NUMBERS		
A Source Group 1	2 AREA Reset, DATA, READONLY		
1 demo.s	3 EXPORT Vectors		
	4Vectors		
	5 DCD 0X20001000		
	6 DCD Reset_Handler;		
	7		
	8 AREA MULTIPLY, CODE, READONLY		
	9 ENTRY		
	10 EXFORT Reset_Handler		
	12 MOV no frum1		
	12 NOV 10, FILMI		
	14 MIII. r2 r0 r1		
	15 LDR r3.=product		
	16 STR r2, [r3]		
	17 stop B stop		
	18		
	19 AREA DATA2, DATA, READWRITE		
	20 num1 EQU OXFFFF ;maximum valur of 16 bit number		
	21 num2 EQU OXFFFF		
	22 product DCD 0X0		
	23 END		
	24		
🔄 Pr 🗑 Bo {} Fu 🛛 " Te	· [•
Build Output			a 🗃
assembling demo.s			*
demo.s(19): warning: A158	W: Added 2 bytes of padding at address 0x12		
"demo.s" - 0 Error(s), 1	Jarning(s).		
4			•
Translate the currently active file		Simulation	L:24 C:1 CAP NUM SCRL OVR R/W
) 🔞 🖪 🖻 🖉 🥥 🕅 🧭		💐 🚳 🔐 🟴 💷 🤯 🦉 🕪 343 PM

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



EMBEDDED SYSTEM & DESIGN(20EC504/J01A)

11. Start the debug session from Menubar.

C:\Users\User\Desktop\demo.uvp	roj - µVision4		(Art.) Constant of Name (Name) 71-		
File Edit View Project Flash	Debug Peripherals Tools S	VCS Window Help			
🗋 🗃 🖬 🖉 🕹 🙇 🕵 !	Q Start/Stop Debug Session	Ctrl+F5 /= //= / 🖄	💽 🗟 🥐 🛛 🔕 🖢	o 🔗 🍓 🔲 🔍	
🖗 🕮 🗃 🥔 🧮 💥 Targe	RST Reset CPU				
Project 🛛 🕫 🔝	Run	F5			▼ ×
🖂 🚵 Target 1	🔇 Stop	BIT NUMBER	RS		
B G Source Group 1		F11 READONLY			
tere 👔 demo.s	3. Step Over	F10			
	{ } Step Out	Ctrl+F11			
	-{} Run to Cursor Line	Ctrl+F10			
	Show Next Statement	E, READONLY	Y		
	Breakpoints	Ctrl+B er			
	Insert/Remove Breakpoint	F9			
	O Enable/Disable Breakpoint	Ctrl+F9			
	🔗 Disable All Breakpoints				
	🚓 Kill All Breakpoints	Ctrl+Shift+F9			
	OS Support	>			
	Execution Profiling	EADWRITE			
	Memory Map	ximum valu	r of 16 bit number		
	Inline Assembly				
	Function Editor (Open Ini Fil	e)			
Pr. Bo. B. Fu. D. Te.					
Build Output	<u></u>				# B
assembling demo.s					
demo.s(19): warning: A158	1W: Added 2 bytes of p	adding at address 0x12	2		
"demo.s" - 0 Error(s), 1	warning(s).				•
4					F.
Enter or leave a debug session				Simulation	L:24 C:1 CAP NUM SCRL OVR R/W
🛞 📋 🥭 🛛	0 🕹 🔼		M 🚿		🂐 👸 🔐 🏴 📶 🧤 🔐 🌒 3:46 PM

12. PressOK

C:\Users\User\Desktop\demo.u	vproj - µVision4	
File Edit View Project Flas	h Debug Peripherals Tools SVCS Window Help	
🗋 🞯 🖬 🌒 🕹 🕰	🤊 🌣 伊 🕸 整 整 連 連 進 版 🖉 📃 見 🕸 💽 🔍 🖉 🔍 🔍 🖉 💭 🔧	
😫 🗄 🔕 🔁 🗗 🖓 🕇		
Registers #	Disassembly	Ф 🔀
Register Value	12: MOV r0, #num1	*
Core 0x00000000 R1 0x00000000 R2 0x00000000 R3 0x00000000 R4 0x00000000	C)xC0000000 cbs/00r 80W r0,40xrrr 13: MOV 11,4nm2 0x0000000 F64F1FF MOVM x1,40xFFFF 14: MUL r2,r0,r1 0x00000010 FB00F201 MUL r2,r0,r1 15: LDR r3,=product 0x0000014 4B01 LDR r3,fpc,41 : 00x000001C	
R5 0x0000000	16: STR r2, [r3]	
R7 0x00000000	A 4 4 5 5 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5	
R8 0x0000000		
R9 0x00000000	i deno.s	• •
R11 ∆c0000000 R13 (SP) ∆c0000000 R14 (SP) ∆c0000000 R14 (SP) ∆c0000000 R15 (PC) ∆c0000000 B+ SFR ∆c010000000 B+ System Thread Phylage Phylage States 0 Sates 0 Set 0	 ILD GALGOLUGO DCD Reseg Handler: AREA MULTFLY, CODE, READONLY ENTRY EXPORT Reseg Handler Reseg Handler NOW ri, humal MOW ri, humal MOW ri, humal LD RF3, product STR r3, [r3] AREA DATA2, DATA, READWRITE 	E
E Project Registers	(m	•
Command	🗶 🔝 Call Stack + Locals	4 🖂
*** Currently used: 36	Bytes (0%)	
ASSIGN BreakDisable Bre	akinable Breakkili Breaklist Breaklist Breaklist Gould Breaklist Gould Breakkili Memori	
	Simulation 11:0.0000000 sec E12 C1 G	AP NUM SCRE OVR R/W
💓 🔝 C	0 0 484 0 48	5/30/2021

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 11 MOV r2,#resu B l2 MOV r2, #res 12 LDR r3,=result 11 STR r2,[r3] stop B stop AREA data, DATA, READWRITE num DCW 16 result DCB 0X0 END

RESULT: num=16d.Hence it is EVEN



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.



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



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

		and appendix the second second second	
Create New Project	speak the life many real		
🕘 💭 🗢 🔳 Desktop 🔸	- 4 Search Desktop		
Oreanize = Newfolder		- 0	
organize + New Holder	83		
🔆 Favorites 🍵 🧊 Libraries	Dijects		
Desktop	p 📕 practice programs		
😹 Downloads 🦷 📙 User	printout_june8th		
🔢 Recent Places 📃 📲 Computer	📕 ResultDigit		
 OneDrive Network 	📕 Seshu projects		
🎍 activity	🕌 studentDetails		
词 Libraries 🔰 Keerthana	studentTestPapers		
Documents	im I I 👍 visiLab		
J Music			
Dietuese			
File name: demo		•	
Save as type: Project Files (*.uvproj)		•	
Owner	Sam	and a second sec	
Hide Folders	Jave		
Pr 🗑 Bo 🛛 Fu 0., Te			
Pr 🗑 50 () Fu (), Te			a
Pr ∰ B0 () FU ()			a
]Pr]@≣o () ∓u 0., Te ild Output			a
Pr ⊕ 50 Ω Fu Ω _φ Te id Output			a
Pr ∰ 50 {} 1 ⊓u 0 ⊺e Id Output			a ,
Pr 0 F0 0 70 0			a Cap' Num' screl over r

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

C:\Users\User\Desktop\demo.uvproj - µVision4			
File Edit View Project Flash Debug Peripherals Tool	SVCS Window Help		
	微微 課 課 /// //// 🐸 👘	- 1 1 1 Q 0 · 0 1 1 1 - 4	
😢 🗂 🥔 🤐 💥 🛛 Target 1 💽 🔊	8 8 ♦ @		
Project 🗣 🙆			
	Select Device for Target 'Target 1'		
■Pr. (0 00 () Fu [0, 7c]	CPU Vendar: IXXP Device: LPC1768 Totelet: ARM Search: (sc1768) ■ Ø NXP ■ Ø NXP ■ Ø LPC1766	Description: ARM 32bB Cates-M3 Microcontroller with MPU, CPU clock up to 100° + 52:85 or -che Rein FADM with enhanced Reina Memory Accelerator, hysterin Regimmers (BPA), and hysterication Regimmers (BPA). Byster Regimmers (BPA) and the retranscel Reina Memory Accelerator, hysterin Regimmers (BPA). Byster Regimmers (BPA) and hysterication Regimmers (BPA). Byster Regimmers (BPA) and hysterication Regimmers (BPA). Byster Regimmers (BPA) and hysterication Regimmers (BPA). Better Regimmers (BPA) and hysterication Regimmers (BPA). Better Regimmers (BPA) and hysterication Regimmers (BPA). CAV 2.8 bit his charafter (BPA) and hystericate (BPA). Three IZ seed reflexions. These SVASS mean instructors, 125.8 bits. Review States (Sectionses. States) and Rest Level. Three IZ seed reflexions. Rest Rest Control. Cloadstrue Encoder. Mater control PVM for three-phase. Mater control. Cloadstrue Encoder. Nator control PVM for three-phase. Mater control. Cloadstrue Encoder. Nator control PVM for three-phase. Mater control. Cloadstrue Encoder. Nator control PVM for three-phase. Mater control. Cloadstrue Encoder. Nator control PVM for three-phase. Mater control. Cloadstrue Encoder. Nator control PVM for three-phase. Mater control. Cloadstrue Encoder. Nator control PVM for three-phase. Nator control PVM for three-phase.	
Build Output			a 🖬
<i>ϵ</i>			4 * 4
			CAP NUM SCRL OVR R /W
			🍕 👸 🔐 🏴 📶 🧤 🛱 🌒 5/30/2021 🔤

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

🔞 C\User\User\Desktop\demouvproj - µVision4	
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help	
▶◎月◎ ↓ 555 つ ● ◇ ● 巻き 8 準 進 推 施 ◎ ● ■ ▲ ◇ ◎ ● ◆ ④ ■ ▲	
② □ □ @ @ [] Target - 水 晶 音 金 飾	
Project 	
Build Output	μ 🔝
4	
	CAP NUM SCRL OVR R/W
a a a a a a a a a a a a a a a a a a a	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\Deskt	op\de	mo.uvproj - µVision4	and an	and allow thread the		
File Edit View Pr	oject	Flash Debug Peripherals Tools SVCS Window	Help			
🗋 🗋 😂 📓 🖉		四一 の の の の の の の の の の の の	1 // it 🆄	💽 🗟 🦑 🔍 🔍 💿	🔗 🍂 🔳 🔧	
🔗 🖽 🕮 🥔 📇	L	Target 1 💽 🎊 🛔 🗟 🚸 🍏				
Project		P 🗃				
🖃 🚞 Target 1						
Source Grou	P AN	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				
	122	Stop build				
	1	Show Include File Dependencies				
	_					
	10	70				
Build Output	and c					
build Output						÷ •
						*
*					Lat. 1 and 1	۱
Add a new Item to Grou	p				simulation	CAP NUM SCRL OVR R/W
	E	· · · · · · · · · · · · · · · · · · ·	9 🛛	1		🍇 🚳 🦉 🏴 📶 🧤 🛱 🗤 5/30/2021 🔤

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

C:\Users\User\Desktop\demo.uvproj - µVision4		min properties allower research man		
File Edit View Project Flash Debug Peripheral	s Tools SVCS Window Help			
N 😂 😹 🕼 🛦 💩 🖓 🗠 🖗 👘	产业性化 建建用	୭ 🕞 🗟 🖉 🔍 🔍 🖉) 🚓 🔲 🔧	
🔗 🛗 😂 🤐 🕌 🙀 🛛 Target 1	🖌 👗 🗟 🚸 🎰			
Project 🛛 🛱 🙆				
E- 🔁 Target 1				
Source Group 1	Add New Item to Group 'Source G	iroup 1'		
 ■Pr@50[Ω_F0]	C File (c) C + File (cpp) Am File (c) Header File (h) Tot File (tx) Mage File (7) User Code Template Type: Am File (4) Name: Location: C:\User\User	Create a new assembler source file and add it to the project		
Build Output				# 🖬
				*
				-
4				F
			Simulation	CAP NUM SCRL OVR R/W
🚱 📋 🙆 🛛 🔞	📙 🗖 🖉 🄇	2 👿 🚿		🛛 🧸 🖓 🚰 🏴 💷 🧞 🛱 🌒 3:40 PM 5/30/2021

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

C:\Users\User\Desktop\demo.uvpro	uVision4	- 0 - X -
File Edit View Project Flash	zbug Peripherals Tools SVCS Window Help	
🗋 🗋 🖉 🖉 🕹 🛍 🖣 🤊	▷ (◇◇) 🖗 魯 魯 宰 非 止 版 🦉 💿 😡 🖉 🔍 🔍 🔍 🔍 🖉	
🔮 🎬 🛗 🧼 🗮 🙀 Target		
Project 🛛 🛱 🔯	demos*	▼ ×
⊡-in Target 1 ⊕ in Source Group 1	<pre>1 ;ALF TO MULTIPLY TWO 16 BIT NURBERS 2</pre>	
E Pr	M	+
Build Output		P 🖬
		*
	Simulation 1-24-C1 CAP NUM	SCRI OVR RAW
👧 📋 🖉 🕻	0 😼 🖪 🔄 🚫 🔛 🛷	3:42 PM 5/30/2021

EMBEDDED SYSTEM & DESIGN(20EC504/J01A)

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



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

C:\Users\User\Desktop\demo.uvpro	- µVision4	and have a second se	
File Edit View Project Flash	Pebug Peripherals Tools SVCS Window Help		
🗋 🗃 🖬 🖉 🕹 🛍 💆	○ ◆ → ❷ 魯 魯 魯 淳 淳 川 順 🙆 💽 🔒 🌮	Q 🗕 🖉 🔗 🏨 🗐 📲	
🖉 🕮 🕮 🧼 🧮 🛛 🗱 🛛 Target	🖬 🖈 📥 🐜 🍩		
Project 🔛 Build (F7) 🕈 🖾	demo.s		▼ ×
Build target files	1 ;ALP TO MULTIPLY TWO 16 BIT NUMBERS		
🖻 🤤 Source Group 1	2 AREA Reset, DATA, READONLY		
📩 demo.s	3 EXPORTVectors		
	5 DCD 0X20001000		
	6 DCD Reset Handler;		
	7		
	8 AREA MULTIPLY, CODE, READONLY		
	9 ENTRY		
	11 Reset Handler		
	12 MOV r0, #num1		
	13 MOV r1, #num2		
	14 MUL r2,r0,r1		
	15 LDR r3,=product		
	10 SIK IZ, [IS] 17 stop B stop		
	18		
	19 AREA DATA2, DATA, READWRITE		
	20 num1 EQU OXFFFF ;maximum valur of 16 bit number		
	21 num2 EQU OXFFFF		
	22 product DCD 0X0		
	23 END		
	**		
🖻 Pr 🔇 Bo {} Fu 🛛 🖓 Te	· [۴
Build Output			4 🖬
assembling demo.s			*
demo.s(19): warning: A1581	W: Added 2 bytes of padding at address 0x12		
"demo.s" - U Error(s), 1 W	arning(s).		-
4			E.
Build target files		Simulation	L:24 C:1 CAP NUM SCRL OVR R/W
👧 📋 🥝 🕻) 🥹 🖪 🔤 🎱 🕅 🚳		🧸 👸 🖉 🏴 .al 🍖 🛱 🌒 3:45 PM 5/30/2021

11. Start the debug session from Menubar.

C:\Users\User\Desktop\demo.uvp	roj - µVision4	Mint School	Additional - Manual New .	
File Edit View Project Flash	Debug Peripherals Tools SVCS	Window Help		
🗋 🖸 🖬 🕼 🕹 🛍 🗎	Start/Stop Debug Session	Ctrl+F5	🗖 🗟 🤌 🍭 🕘 🔹 🔗 🍓 🗐 🔍	
😵 🖽 🗃 🥔 📇 💥 Targe	RST Reset CPU			
Project 📮 🔝	1. Run	F5		▼ ×
🖃 🛅 Target 1	🔕 Stop	BIT NUMBERS		
🖻 🔄 Source Group 1	{} Step	F11 READONLY		
uenos.	3 Step Over	F10		
	{} Step Out	Ctrl+F11		
	"{} Run to Cursor Line	Ctrl+F10		
	Show Next Statement	E, READONLY		
	Breakpoints	Ctrl+B er		
	Insert/Remove Breakpoint	F9		
	O Enable/Disable Breakpoint	Ctrl+F9		
	Disable All Breakpoints			
	Kill All Breakpoints Ctrl-	Shift+F9		
	OS Support	÷		
	Execution Profiling	* EADWRITE		
	Memory Map	ximum valur of 16 bit	number	
	Inline Assembly			
	Function Editor (Open Ini File)			
E Pr 8 Fu 0 Te	4	III		•
Build Output				a 🖬
assembling demo.s				*
<pre>demo.s(19): warning: A158 "demo.s" = 0 Error(s). 1</pre>	1W: Added 2 bytes of padd: Warning(s).	ng at address 0x12		
				*
Foter or leave a debug session			Simulation	
	0 🔞 🖪 🗖		31101000	🤞 🆓 🏴 🖬 🦆 🗊 40 346 PM

12. PressOK

C:\Users\User\E	Desktop\demo.uvp	roj - µVision4	
File Edit View	Project Flash	Debug Peripherals Tools SVCS Window Help	
D 🐸 🖬 🖉	8 4 🕰	🤊 🗠 🖗 🕸 鶴 数 準 罪 准 版 🌌 🕢 🕢 🖬 🕢 🕢 😡 🖉 🔕 鱼 🔿 🔗 🍓 💷 🔦	
8 🗟 🖉	0* 10 10		
Registers	a 💌	Disassembly	# 🖪
Register	Value	12: MOV r0,#num1	*
Core R0 R1 R2 R3	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	CORRO000000 F64F70FF MOVW r0,40xFFFF 13: MOV r1,4 mm2 0x0000000 F64F71FF MOVW r1,4 0xFFFF 14: MUL r2,r0,r1 0x00000010 F800F201 MUL r2,r0,r1 15: LDR r3, product	
R5	0x00000000	0x0000014 4B01 LDR r3,[pc,#4] ; @0x000001C	
R6	0x00000000	16: STR r2, [r3]	-
R7	0x00000000		Þ
	0x00000000	Et demos	▼ ×
R10 R11 R12	0x00000000 0x00000000 0x00000000	5 DCD 0x20001000 6 DCD Rest_Handler; 7	*
	0x20001000 0xFFFFFFF 0x00000008 0x01000000	<pre>8 AREA MULTIPLY, CODE, READONLY 9 FINTY 10 EXPORT Reset_Handler 11 Reset_Handler 12 HOW to 0, found 13 MUV to 1 found 13 MUV to 1 found 14</pre>	E
E Internal Mode Privilege Stack States Sec	Thread Privileged MSP 0 0.00000000	14 MUL 22, 40, 41 15 LDR 45, product 16 STR 42, [13] 17 stop B stop 18 19 AREA DATA2, DATA, READWRITE	
🔃 Project 🗮 Re	gisters	« m	•
Command		a 🖬 Call Stack - Locals	4 🖬
*** Currentl; <	y used: 36 Bj	/rtes (0%) Caration/Value Type	
ASSIGN Break	Disable Break	cEnable BreakKill BreakList BreakSct BreakAccess COVERAGE 🔤 🚱 Call Stack + Locals 💷 Memory 1	
		Simulation t1: 0.0000000 sec L12 C1 CAP NU	JM SCRL OVR R/W
📀 🔚	e	0 🥹 📕 🖾 🚫 🖳 🧭	(*) 3:48 PM 5/30/2021

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 11 LDR r3,[r0],#4 STR r3,[r1],#4 SUBS r2,#1 BNE 11

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 11 LDMIA r0!,{r4-r8} STMIA r1!,{r4-r8} SUBS r2,#1 BNE 11

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.

D:\ARM&EME	B\lap_prgs_2021\wri	tedataRam.uvproj - µVision4	- 0 <mark>- X</mark> -		
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help					
n 🐸 🖬 🖉	7 3 45 🛍	っ ♀ ◆ ☆ 豫 豫 譯 進 版 / 20 ● ● ● ◇ ◆ ● ■ ● ◆ ◆ ◆			
🗱 🗉 🚳	()* () () ()				
Registers	4 	Disassembly	P 🖸		
Register	Value	16:11 LDMIA r01/(r4-r8)	^		
E Core	0.0000010	17: STML 11, (r4-r8)			
RU B1	0x10000016	■0x00000028 E8A101F0 STM +1'.{+4-+R}			
	0x00000004				
	0x0000000	🔛 writedataRam.s	▼ ×		
R4	0x00000011	7	*		
R5	0x00000022	8 AREA writedata, CODE, READONLY			
	0x00000044	9 src DCD 0x11,0X22,0X33,0X44,0X55			
	0x00000055	10 ENIKY 11 FXPORT Reset Handler			
	0x0000000	12 Reset Handler			
R10	0x00000000	13 LDR r0,=src			
	0x0000000	14 LDR r1,=dst			
	0x20001000	15 MOV $r_{2}, \pm 5$			
R14 (OxFFFFFFF	10 11 LUMIA 20:, (29-20) 17 STMIA 11:, (24-28)			
R15 (0x00000024				
E Resked	0x21000000	19 BNE 11	=		
System		20			
E Internal		21 stop B stop			
Mode	Thread	23			
Privilege	Privileged	24 AREA data, DATA, READWRITE			
Stack	21	25 dst DCD 0X0			
Sec	0.00000175	26 END			
📴 Project 🛛 🧱 🖡	Registers	и с	,		
Command		a 📓 Memory 2	a 🛐		
*** Current	ly used: 64 B	(0%) ↓ Address: 0x10000000	e î		
<		0x100000000: 11 00 00 02 20 00 00 33 00 00 04 00 00 55 00 00 00 00	0 00 00		
>		0x10000017: 00 00 00 00 00 00 00 00 00 00 00 00 00	• 00 00 -		
ASSIGN Brea	kDisable Breal	KEnable BreakKill BreakList BreakSet BreakAccess COVERAGE 🛛 🖗 Call Stack + Locals 🗐 Memory 1 📃 Memory 2			
		Simulation t1:0.00000175 sec L16 C1 CAP NUM	SCRL OVR R/W		
	I E	0 🔞 🛃 🖾 🦉 🚫 🕅 🛷 👘 🖉 🄞 🖸	4:20 PM 5/30/2021		
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 16bytes 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

where, DLM=0, DLL=

LPC1768 generates the baud rate depending on the values of DLM, DLL.

Baudrate = PCLK/ (16 * ((256 * DLM) + DLL) * (1+ DivAddVal/MulVal))

, (DivaddVal/MulVal)=0.

Steps for Configuring UARTO

Below are the steps for configuring the UARTO.

1. Configure the P0.2 and P0.3 as first alternate function UARTO 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 UARTO->THR=txdata;
}
void initUART0(void)
LPC_PINCON->PINSEL0 =(1<<4) | (1<<6); LPC_UART0->LCR=0x83;
LPC UARTO->DLL=163; LPC UARTO->DLM=0; LPC UARTO->FCR =0x7;
LPC UARTO->FDR=0x0; LPC UARTO->LCR = 0x03;
}
int main(void)
charmsg[]= "Hello World"; int i=0;
initUARTO(); for(i=0;msg[i];i++)
{Program:
#include<lpc17xx.h>
void UOWrite( 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.



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;
    }
}</pre>
```

```
LPC PINCON->PINSEL0=0X0000000;
                                               LPC PINCON-
                         LPC_PINCON->PINSEL4=0X0000000;
>PINSEL1=0X0000000;
LPC GPIOO->FIODIR=0xFFFFFFF;
                                                LPC_GPIO2-
>FIODIR=0X0000000;
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.

