

# **Lab Manual**

# **Microcontrollers**

## **(BCS402)**



**Rajeev Institute of Technology, Hassan**  
**Department of Computer Science and Engineering**  
**RIT, Hassan**

### Introduction to ARMCORTEX M3:

Cortex-M3 Processor is a RISC general purpose 32-bit microprocessor, released 2006 Cortex-M3 differs from previous generations of ARM processors by defining a number of key peripherals as part of the core: – interrupt controller – system timer – debug and trace hardware (including external interfaces) .This enables for real-time operating systems and hardware development tools such as debugger interfaces be common across the family of processors Various Cortex-M3 based microcontroller families differ significantly in terms of hardware peripherals and memory.

- Greater performance efficiency: more work to be done without increasing the frequency or power requirements – Implements the new Thumb-2 instruction set architecture , 70% more efficient per MHz than an ARM7TDMI-S processor executing Thumb instructions , 35% more efficient than the ARM7TDMI-S processor executing ARM instructions for Dhrystone benchmark , Low power consumption: longer battery life, especially critical in portable products including wireless networking applications ,Improved code density: code fits in even the smallest memory footprints , Core pipeline has 3 stages – Instruction Fetch – Instruction Decode – Instruction Execute

- LPC2148 microcontrollers are based on the Cortex-M3 processor with a set of peripherals distributed across three buses – Advanced High-performance Bus (AHB) and its two Advanced Peripheral Bus (APB) sub-buses APB1 and APB2. , These peripherals: – are controlled by the CM3 core with load and store instructions that access memory mapped registers – can “interrupt” the core to request attention through peripheral specific interrupt requests routed through the NVIC .Data transfers between peripherals and memory can be automated using DMA.

### INTRODUCTION TO KEIL:

One of the important part in making an embedded system is loading the software/program developed into the microcontroller. Usually it is called “burning software” into the controller. Prerequisite operations with the program that must be done before “burning a program” into a controller, this includes writing the program in assembly language or C language in a text editor like notepad, compiling the program in a compiler and finally generating the hex code from the compiled program. Earlier people used different software’s/applications for all these 3 tasks. a. Writing was done in a text editor like notepad/WordPad, b. compiling was done using a separate software (probably a dedicated compiler for a particular controller like 8051), c. converting the assembly code to hex code was done using another software etc. It takes lot of time and work to do all these separately, especially when the task involves lots of error debugging and reworking on the source code. Keil Micro Vision is a free software which solves many of the pain points for an embedded program developer.

This software is an integrated development environment (IDE), which integrated a text editor to write programs, a compiler and it will convert your source code to hex files too.

Keil was founded in 1982 by Günter and Reinhardt Keil, initially as a German. In April 1985 the company was converted to Keil Electronic to market add-on products for the development tools provided by many of the silicon vendors. Keil implemented the first Compiler designed from the ground-up specifically for the 8051 microcontroller. Keil provides a broad range of development tools like ANSI C compiler, macro assemblers, debuggers and simulators, linkers, IDE, library managers, real-time operating systems and evaluation boards for Intel 8051, Intel MCS-251 and ARM families.

## **Microcontrollers lab**

1. Using Keil software, observe the various registers, dump, CPSR, with a simple ALP program.
2. Develop and simulate ARM ALP for data transfer, Arithmetic and logical operations.
3. Write a program to find the sum of the first 10 integer numbers.
4. Write a program to add an array of 16 bit numbers and store the 32 bit result in internal RAM.
5. Write a program to find the largest or smallest number in an array of 32 numbers.
6. Write a program to count the number of ones and zeros in two consecutive memory locations.

1. Using Keil software, observe the various registers, dump, CPSR, with a simple ALP program.

### SIMPLE ALP PROGRAM:-

AREA ADD, CODE, READONLY

ENTRY

START

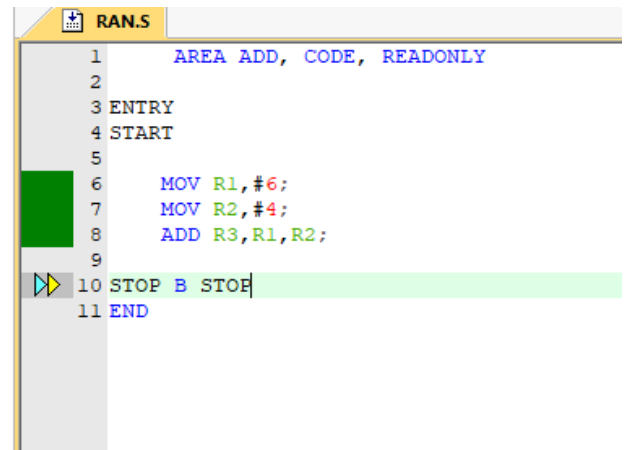
MOV R1,#6;

MOV R2,#4;

ADD R3,R1,R2;

STOP B STOP

END



### OUTPUT:-

| Register       | Value      |
|----------------|------------|
| <b>Current</b> |            |
| R0             | 0x00000000 |
| R1             | 0x00000006 |
| R2             | 0x00000004 |
| R3             | 0x0000000A |
| R4             | 0x00000000 |
| R5             | 0x00000000 |
| R6             | 0x00000000 |
| R7             | 0x00000000 |
| R8             | 0x00000000 |
| R9             | 0x00000000 |
| R10            | 0x00000000 |
| R11            | 0x00000000 |
| R12            | 0x00000000 |
| R13 (SP)       | 0x00000000 |
| R14 (LR)       | 0x00000000 |
| R15 (PC)       | 0x0000000C |
| CPSR           | 0x000000D3 |
| N              | 0          |
| Z              | 0          |
| C              | 0          |
| V              | 0          |
| I              | 1          |
| F              | 1          |

2. Develop and simulate ARM ALP for data transfer, Arithmetic and logical operations.

**DATA TRANSFER:-**

AREA ADD, CODE, READONLY

ENTRY

START

MOV R1,#6;

MOV R2,#4;

MOV R1,R2;

STOP B STOP

END

```

RAN.S
1  AREA ADD, CODE, READONLY
2
3  ENTRY
4  START
5
6  MOV R1,#6;
7  MOV R2,#4;
8  MOV R1,R2;
9
10 STOP B STOP
11 END
  
```

**OUTPUT:-**

| Registers      |                   |
|----------------|-------------------|
| Register       | Value             |
| <b>Current</b> |                   |
| R0             | 0x00000000        |
| R1             | 0x00000004        |
| R2             | 0x00000004        |
| R3             | 0x00000000        |
| R4             | 0x00000000        |
| R5             | 0x00000000        |
| R6             | 0x00000000        |
| R7             | 0x00000000        |
| R8             | 0x00000000        |
| R9             | 0x00000000        |
| R10            | 0x00000000        |
| R11            | 0x00000000        |
| R12            | 0x00000000        |
| R13 (SP)       | 0x00000000        |
| R14 (LR)       | 0x00000000        |
| R15 (PC)       | 0x0000000C        |
| CPSR           | 0x000000D3        |
| SPSR           | 0x00000000        |
| +              | User/System       |
| +              | Fast Interrupt    |
| +              | Interrupt         |
| +              | <b>Supervisor</b> |
| +              | Abort             |
| +              | Undefined         |
| Internal       |                   |
| PC \$          | 0x0000000C        |
| Mode           | Supervisor        |
| States         | 6                 |
| Sec            | 0.00000050        |

**ARITHMETIC AND LOGICAL OPERATIONS:-**

AREA ADD, CODE, READONLY

ENTRY

START

MOV R0,#6;

MOV R1,#4;

ADD R2,R1,R0;

ADC R3,R1,R0;

SUB R4,R1,R0;

SBC R5,R1,R0;

RSB R6,R1,R0;

RSC R7,R1,R0;

AND R8,R1,R0;

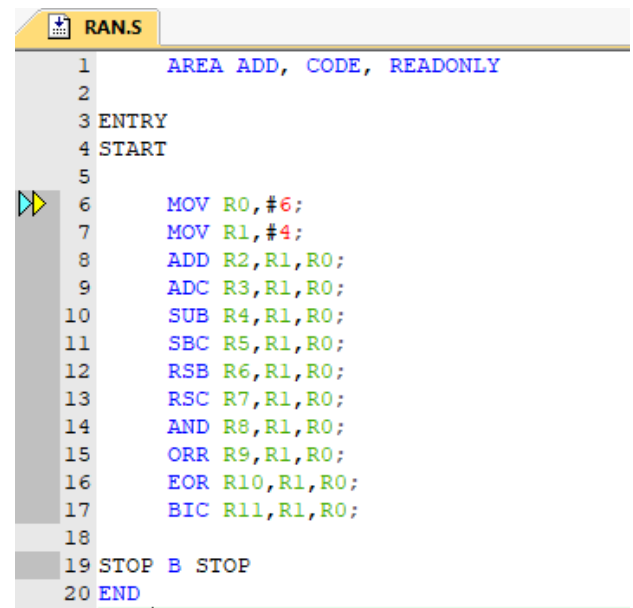
ORR R9,R1,R0;

EOR R10,R1,R0;

BIC R11,R1,R0;

STOP B STOP

END



```
1 AREA ADD, CODE, READONLY
2
3 ENTRY
4 START
5
6 MOV R0,#6;
7 MOV R1,#4;
8 ADD R2,R1,R0;
9 ADC R3,R1,R0;
10 SUB R4,R1,R0;
11 SBC R5,R1,R0;
12 RSB R6,R1,R0;
13 RSC R7,R1,R0;
14 AND R8,R1,R0;
15 ORR R9,R1,R0;
16 EOR R10,R1,R0;
17 BIC R11,R1,R0;
18
19 STOP B STOP
20 END
```

**OUTPUT:-**

| Register            | Value       |
|---------------------|-------------|
| <b>Current</b>      |             |
| R0                  | 0x00000006  |
| R1                  | 0x00000004  |
| R2                  | 0x0000000A  |
| R3                  | 0x0000000A  |
| R4                  | 0xFFFFFFFFE |
| R5                  | 0xFFFFFFFFD |
| R6                  | 0x00000002  |
| R7                  | 0x00000001  |
| R8                  | 0x00000004  |
| R9                  | 0x00000006  |
| R10                 | 0x00000002  |
| R11                 | 0x00000000  |
| R12                 | 0x00000000  |
| R13 (SP)            | 0x00000000  |
| R14 (LR)            | 0x00000000  |
| R15 (PC)            | 0x00000030  |
| + CPSR              | 0x000000D3  |
| + SPSR              | 0x00000000  |
| + User/System       |             |
| + Fast Interrupt    |             |
| + Interrupt         |             |
| + <b>Supervisor</b> |             |
| + Abort             |             |
| + Undefined         |             |
| - Internal          |             |
| PC \$               | 0x00000030  |
| Mode                | Supervisor  |
| States              | 12          |
| Sec                 | 0.00000100  |



3. Write a program to find the sum of the first 10 integer numbers.

### SUM OF THE FIRST 10 INTEGER NUMBERS:-

AREA ADD, CODE, READONLY

ENTRY

START

MOV R0,#10;

MOV R1,#0;

LOOP

ADD R1,R0,R1;

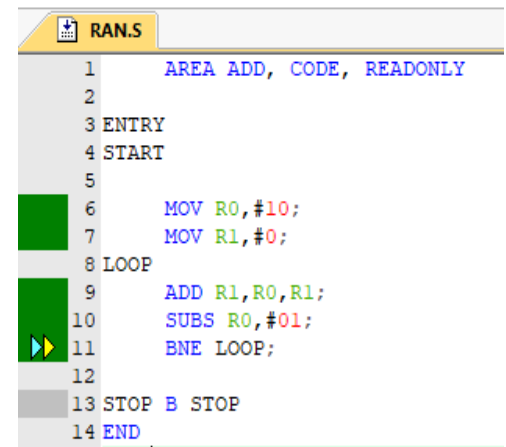
SUBS R0,#01;

BNE LOOP;

STOP B STOP

END

### OUTPUT:-



```

1  AREA ADD, CODE, READONLY
2
3  ENTRY
4  START
5
6  MOV R0,#10;
7  MOV R1,#0;
8  LOOP
9  ADD R1,R0,R1;
10 SUBS R0,#01;
11 BNE LOOP;
12
13 STOP B STOP
14 END

```

| Register          | Value      |
|-------------------|------------|
| <b>Current</b>    |            |
| R0                | 0x00000000 |
| R1                | 0x00000037 |
| R2                | 0x00000000 |
| R3                | 0x00000000 |
| R4                | 0x00000000 |
| R5                | 0x00000000 |
| R6                | 0x00000000 |
| R7                | 0x00000000 |
| R8                | 0x00000000 |
| R9                | 0x00000000 |
| R10               | 0x00000000 |
| R11               | 0x00000000 |
| R12               | 0x00000000 |
| R13 (SP)          | 0x00000000 |
| R14 (LR)          | 0x00000000 |
| R15 (PC)          | 0x00000010 |
| CPSR              | 0x600000D3 |
| N                 | 0          |
| Z                 | 1          |
| C                 | 1          |
| V                 | 0          |
| I                 | 1          |
| F                 | 1          |
| T                 | 0          |
| M                 | 0x13       |
| SPSR              | 0x00000000 |
| User/System       |            |
| Fast Interrupt    |            |
| Interrupt         |            |
| <b>Supervisor</b> |            |
| Abort             |            |
| Undefined         |            |
| <b>Internal</b>   |            |
| PC \$             | 0x00000010 |
| Mode              | Supervisor |
| States            | 49         |
| Sec               | 0.00000408 |

