

# **EE22016 : Microprocessor Theory and Lab**

## **Lab Experiment # 5**

### **ARM Assembly - Computations in ARM**

Batch 35

VIKAS GOLLAPALLI, EE21B051 ,  
SARATH CHANDRA ,EE21B073,

October 19, 2022

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Aim of the Experiment .....	1
1.2	Equipments .....	1
1.3	Concepts Required .....	1
1.4	Overview of KEIL software .....	2
1.5	Problems .....	2
<b>2</b>	<b>Computing the factorial of a given number</b>	<b>2</b>
2.1	Flow Chart .....	2
2.2	Code.....	2
2.3	Result.....	3
<b>3</b>	<b>Combine the last nibble of 4 number to create a 16 bit half word</b>	<b>4</b>
3.1	Flow Chart .....	4
3.2	Code.....	4
3.3	Results Images .....	5
<b>4</b>	<b>Figuring whether a given number is even or odd</b>	<b>6</b>
4.1	Flow Chart .....	6
4.2	Code.....	6
4.3	Result Images.....	7
<b>5</b>	<b>Inference</b>	<b>7</b>
<b>6</b>	<b>Learning Outcomes</b>	<b>7</b>

## List of Figures

1	Flow Chart for the program to find the factorial of a given number .....	2
2	The image of the result.....	3
3	Flow chart to create a halfword from the LSB nibbles.....	4
4	Result Image .....	5
5	Flow Chart for 4 finding whether a given number is even or odd .....	6
6	Result for odd number.....	7
7	Result for even number .....	7

# 1 Introduction

## 1.1 Aim of the Experiment

The aim of the experiment is:

1. Learn the architecture of ARM processor
2. Learn basics of ARM instruction set, in particular the ARM instructions pertaining to computations
3. Go through example programs
4. Write assembly language programs for the given set of (computational) problems

## 1.2 Equipments

To perform this experiment the following components are required

- KEIL 5 IDE for ARM
- Flashmagic software for programming flash memory
- ARM 7 Hardware Kit
- USB to serial converter
- Serial Cross cable

## 1.3 Concepts Required

The contents of the book Welsh especially from the chapter 6. And the structure of ARM architecture

- A large array of uniform register
- A load/store model of data-processing where operations can only operate on registers and not directly on memory. This requires that all data be loaded into registers before an operation can be performed, the result can then be used for further processing or stored back into memory
- A small number of addressing modes with all load/store addresses being determined from register and instruction fields only.
- A uniform fixed length instruction (32 bit)

In addition to these traditional features of a RISC system the ARM provides a number of additional features

1. Separate arithmetic logic unit and shifter giving additional control over dataprocessing to maximise execution speed.
2. Auto increment and auto decrement addressing modes to improve the operation of program loops.
3. Conditional execution of instructions to reduce pipeline flushing and thus increase execution speed.

## 1.4 Overview of KEIL software

Keil u Vision is an IDE directed towards code development for multiple platforms like AVR, ARM, CORTEX-M, C166, C251, C51 and 8051 based MCU architectures manufactured by various companies. Whereas Atmel Studio is a Visual Basic and .NET Framework based IDE which only supports AVR and ARM architecture based MCU's only by Atmel.

## 1.5 Problems

The problems we need to solve in the experiment are:

1. Compute the factorial of a given number using ARM processor through assembly programming
2. Combine the low four bits of each of the four consecutive bytes beginning at LIST into one 16-bit halfword. The value at LIST goes into the most significant nibble of the result. Store the result in the 32-bit variable RESULT.
3. Given a 32 bit number. identify whether it is even or odd.

## 2 Computing the factorial of a given number

### 2.1 Code

```
AREA factorial, CODE
ENTRY

MOV R0, #1          ;program to find factorial
MOV R1, #6          ;int c =1
MOV R3, #1          ;int n=1
BL loop            ;int fact=6
B STOP
loop
    MUL R4, R3, R0
    MOV R3, R4
    ADD R0, R0, #1
    CMP R0, R1
    BLE loop
    BX LR
STOP B STOP        ;R4 IS FINAL ANSWER

END
```

Listing 1: Code for giving out factorial of a given number

## 2.2 RESULTS

C:\Users\ielab1\Documents\batch35\uvproj - uVision

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Registers

Register	Value
R0	0x0000000A
R1	0x00000009
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)	0x00000010
R15 (PC)	0x0000002C
CPSR	0x200000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC 3	0x0000002C
Mode	Supervisor
States	100
Sec	0.00002500

Disassembly

```

17: STOP B STOP ;R4 IS FINAL ANSWER
0x0000002C EAffffff B 0x0000002C
0x00000030 00000000 ANDEQ R0,R0,R0
0x00000034 00000000 ANDEQ R0,R0,R0
0x00000038 00000000 ANDEQ R0,R0,R0

sarath facts
1 AREA factorial, CODE
2 ENTRY
3 ;program to find factorial
4 MOV R0,#1 ;int c =1
5 MOV R1,#9 ;int m=1
6 MOV R3,#1 ;int fact=6
7 BL loop
8 B STOP
9
10 loop
11 MUL R4,R3,R0
12 MOV R3,R4
13 ADD R0,R0,#1
14 CMP R0,R1
15 BLZ loop
16 BX LR
17 STOP B STOP ;R4 IS FINAL ANSWER
18
19 END

```

Command

```

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 48 Bytes (0%)

```

Call Stack - Locals

Name	Location...	Type
void f0	0x00000000	void f0

ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR

Real-Time Agent: Target Reset Simulation t1: 0.00002500 sec L17 C:1 CAP NUM SCRL DIVR R/W

ENG 3:21 PM 10/14/2022

### 3 Combine the last nibble of 4 number to create a 16 bit halfword

#### 3.1 Flow Chart

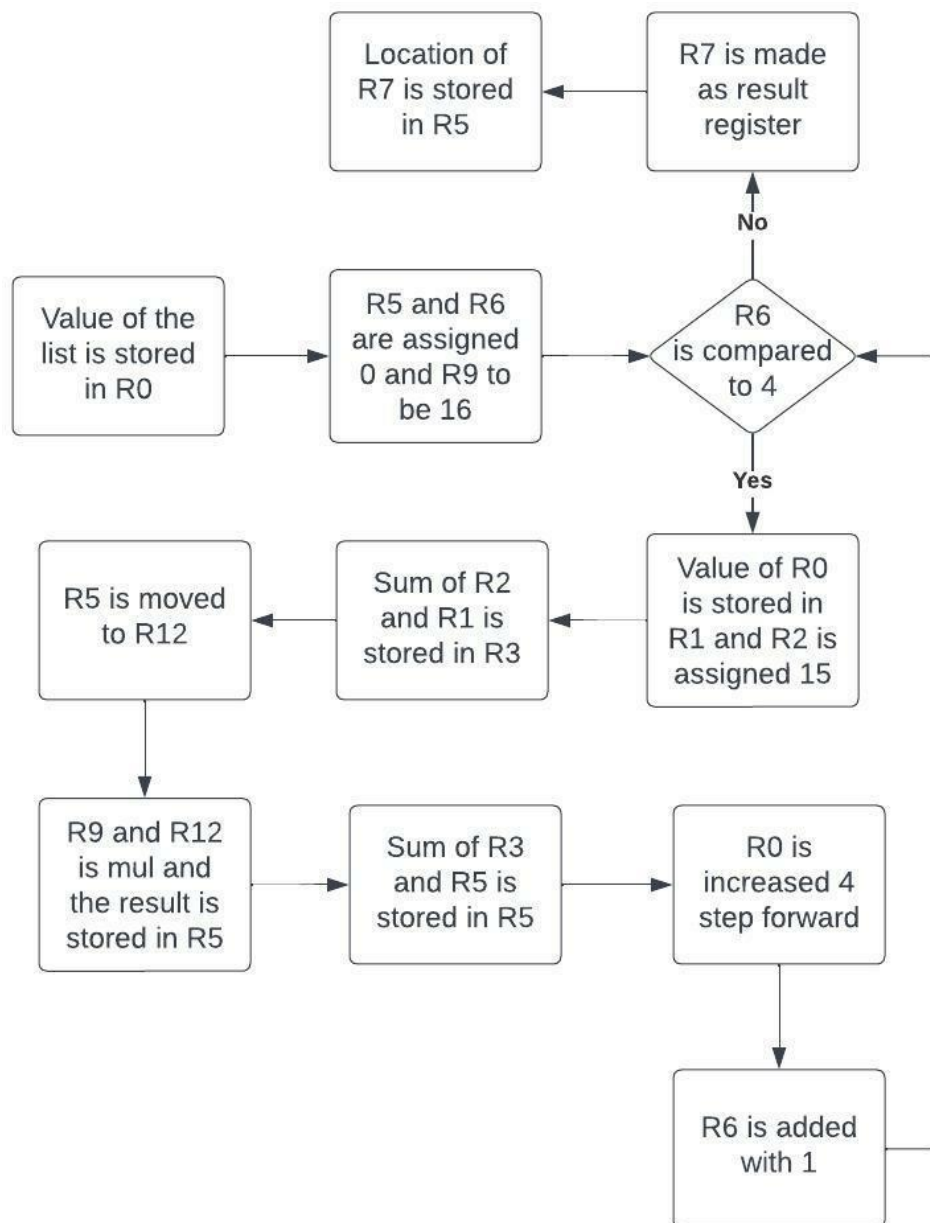


Figure 3: Flow chart to create a halfword from the LSB nibbles

#### 3.2 CODE

```

AREA Program, CODE, READONLY
Entry
    LDR R0, =LIST
    MOV R5, #0
    MOV R6, #0
    MOV R9, #16
Loop
    CMP R6, #4
    BEQ done
    LDR R1, [R0]
    MOV R2, #15
    AND R3, R2, R1
    MOV R12, R5
    MUL R5, R12, R9
    ADD R5, R5, R3
    ADD R0, R0, #4
    ADD R6, R6, #1
    B Loop
done
    LDR R7, =Result
    STR R5, [R7]

    SWI &11

LIST DCD &2D3F, &5F53, &1234, &0987
    align

    AREA DataRam, DATA, READWRITE
Result DCD 0
    END

```

Listing2:code for the question

## 3.2 Results Images

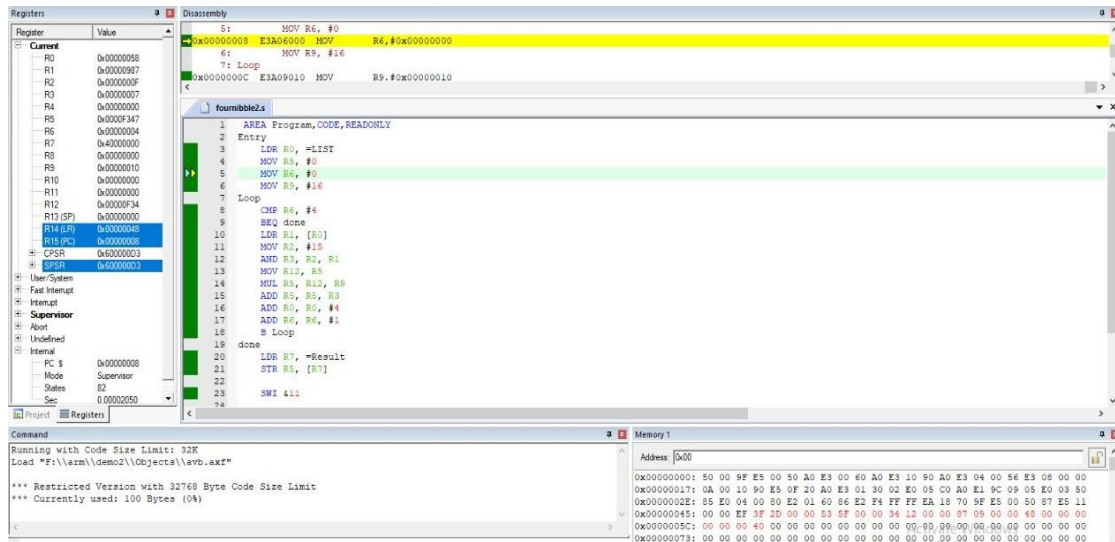


Figure 4: Result Image



## 4 Figuring whether a given number is even or odd

### 4.1 Flow Chart

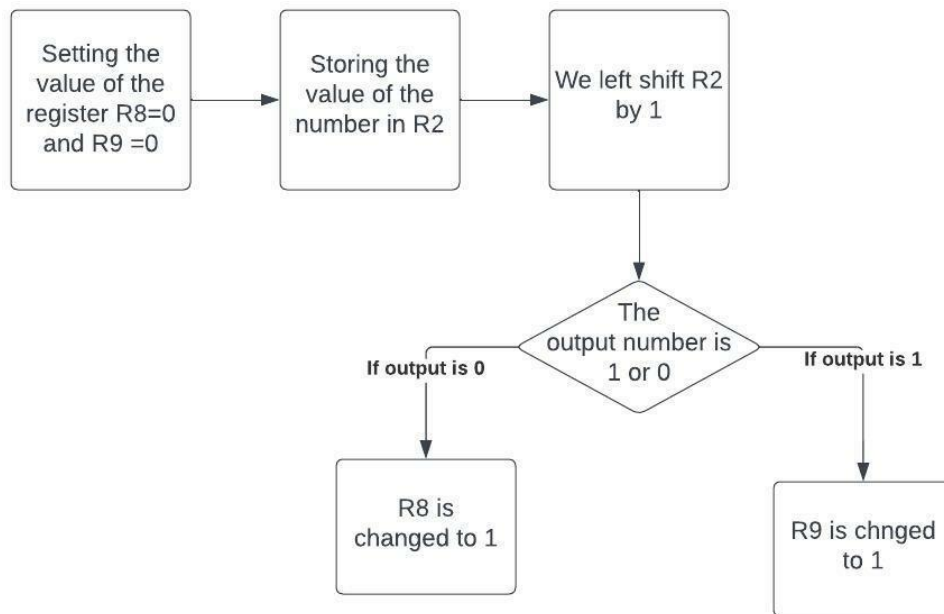


Figure 5: Flow Chart for 4 finding whether a given number is even or odd

### 4.2 Code

```
AREA evenODD, CODE, READONLY
ENTRY
MOV R8,#0 ; For storing even numbers
MOV R9,#0 ; For storing odd numbers
NEXT
LDR R2,=0x44444444 ; Take 1st number
LSRS R2,#1 ;Shift number to right
BCC EVEN ; If C=0 go to even
ADD R9,#1 ; IF C=1 R9=R9+1
BAL XX ;Always Jump

EVEN
ADD R8,#1
END
```

## 4.3 Result Images

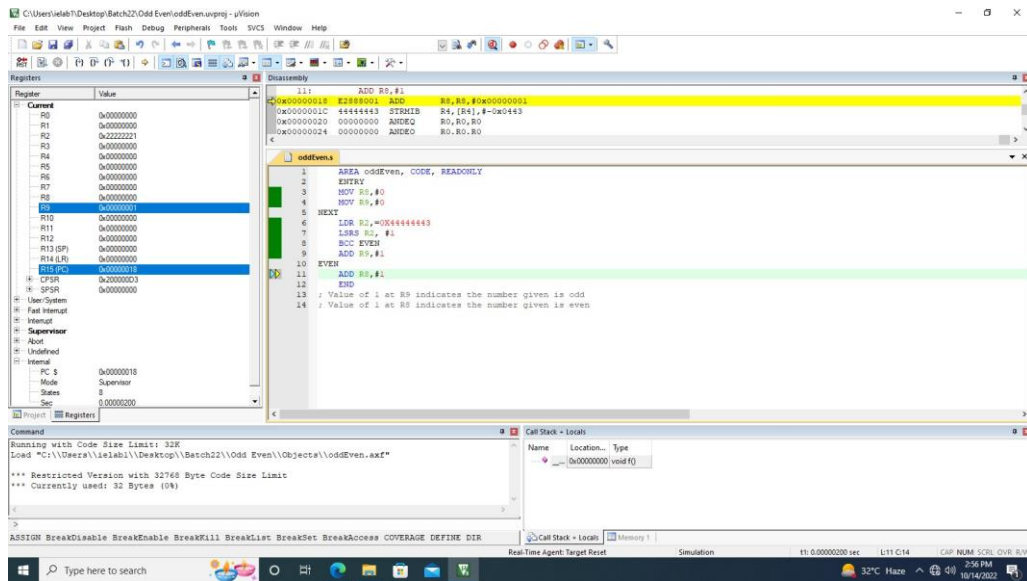


Figure 6: Result for odd number

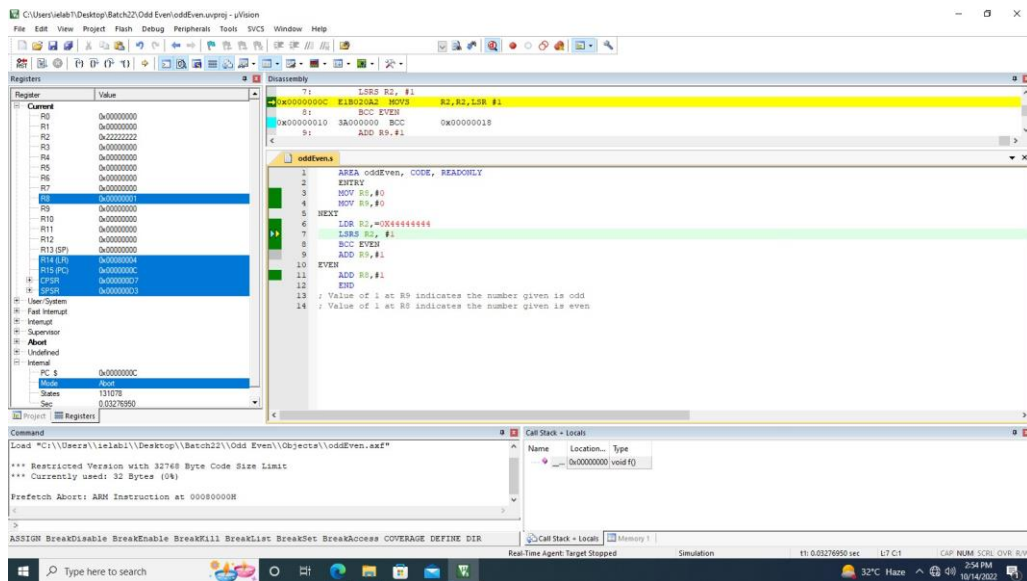


Figure 7: Result for even number

## 5 Inference

In the above experiment we wrote the code for the given program in the Assembly Program which can be used to directly manipulate hardware, access specialized processor instructions, or evaluate critical performance issues.

## 6 Learning Outcomes

- We learnt to write code in the assembly language

- We used the KEIL software to run the program
- We also learnt about how to debug the instruction
- We also learnt about various instruction which we can use in the ARM AssemblyLanguage Program