ECE 2510

Lab 5

Objectives:

• Multiprecision Addition/Subtraction.
• Understand Binary Coded Decimal (BCD) representation of numbers.
• Use branches to implement control statements and loops.
• Understand ARM Control instructions.
• Practice loops.

Introduction:

In this lab we will expand our knowledge of binary arithmetic by looking into multiplication/division, larger data sizes, and binary coded decimal representation of numbers.

Task 1

We can combine two of the tasks together to create a program that adds multiprecision BCD numbers together: Write an assembly program that adds the two BCD numbers: 0x694272 + 0x378925

• The result should be 3 bytes long (each byte represents two binary coded digits).
• store the result starting at address 0x20000000.

Task 2

The easiest way to understand branch instructions is to step through a program in the simulator and observe their behavior. Use Keil to simulate the following program. Step through the code one line at a time and observe what happens. For the lab report, describe what the program does; Specifically, watch the program counter and describe its behavior.
Hand calculations: For each branch instruction, calculate the branch offset.

```
back:
        B forward
        NOP
        NOP
        NOP
forward:
```

```
NOP
B back
```

Task 3

We can use conditional branches or conditional execution to implement if/else statements. Implement the following pseudocode in ARM assembly.

- First, draw a flowchart representing the program.
- Write the ARM assembly program.
- Calculate the execution time with hand calculations, showing the clock cycles for each instruction. Then compare the total cycles to that from the simulator.

```
var1 = 10

if var1 > 0:
    var2 = 100

else:
    var2 = -50
```

Task 4

We can use control instructions to execute a block of code multiple times, in loops. For this task, implement a loop that only executes a certain number of times. To do this, we must use conditional branches. Write an assembly program that increments A 100 times.
• Draw a flowchart for your program.
• Write the assembly code

Task 5

Write an ARM assembly program that reads an array of one-byte elements and increments each element by one. The array consists of 10 elements. Use a for loop.
• Draw a flow chart for your program.
• Write the assembly program. Define all variables and the array

Task 6

In a previous lab, we wrote a program that added up the first 10 natural numbers. Now we can write a better version of that program using branch instructions: write an assembly program that uses a loop to find the sum of the first 10 natural numbers.
• Draw a flowchart of your program.
• Write the assembly code.

Lab Report

• For all tasks, comment on your code and mention any challenges.

Task 1:

• Assembly Code
• Screenshot showing the final results.

Task 2:

• For each branch instruction, calculate the branch offset.

Task 3:

• First, draw a flowchart representing the program.
• Write the ARM assembly program.
• Calculate the execution time with hand calculations.
• Snapshot for the execution window showing the execution time.

Task 4:

• Draw a flowchart for your program.
• Write the assembly code.

Task 5:

• Draw a flow chart for your program.
• Write the assembly program. Define all variables and the array.

Task 6:

• Draw a flowchart of your program.
• Write the assembly code.