ECE 2510

Lab 3

Introduction:

In this lab, you will learn about ARM assembly and practice writing programs using basic instructions. It is important to know the effect each instruction has on the registers and memory. You will simulate the programs and observe how each instruction changes the data in memory and in the registers including the condition code register

Addressing modes and Assembler directives can be found in the class slides and should be referenced for this lab

Task 1

Create a new ASM project

• Type the code into Keil and simulate it.

• Step through each instruction and watch the destination register(s) change. If the instruction writes to a register, take a screenshot of the register window. If the instruction modifies memory, take a screenshot of the memory window at the correct address

```
MOV R0, #0

MOV R0, #0x27

MOV R1, #5

ADD R0, R0, R1

LDR R2, =0x20000000

STRB R0, [R2, #0]

STRB R1, [R2, #1]

LDR R3, =0x20000000

LDRB R0, [R3, #0]

LDRB R0, [R3, #1]
```

Task 2

The ASPR is a 32-bit register that records specific events during program execution. For example, if an instruction causes an overflow, the V flag will be set high. If the result of an instruction is zero, the Z flag will be set, etc

Create a new ASM project
• Write a program that adds the following 32-bit integers together:
0x00000000 + 0xFFFFFFFF + 0x00000001 + 0x12345678 + 0x89ABCDEF + 0x7FFFFFFF + 0x80000000 + 0x00000001 + 0xFFFFFFFF
• Simulate the program step-by-step and take a screenshot of the register window after each step.

Task 3

In this task, write an ASM code that will access memory using different addressing modes. Before starting this task, make sure you understand how the load and store instructions work,as well as the different addressing modes.

Write an ASM program that writes the following bytes to memory starting at address 0x20000020:
{10, 20, 30, 40, 50}

- Now re-write the program to use the register offset addressing mode.
    - Step through the program and screenshot the final values in memory.
- Now re-write the program to use the post indexed addressing mode.
    - Step through the program and screenshot the final values in memory

Task 4

For each of the following tasks, label the values initialized by DCB or DCD as array/var1, 2, etc. For example, if asked to define 2 separate arrays, the first array would be labeled array1, and the second array would be labeled array2.

- Use **DCB** to define **0xB5**
- Use **DCB** to define the following: **{0x15, 0x1A, 0x15, 0x2C}**
- Use **DCD** to define {**0x230D, 0x203F, 0x02D0, 0x00A2}**
- Use **SPACE** to define **100 bytes**
- Place the following directive directly after the first (where you defined **0xB5**): **DCB 0x47**
- For the report, take snapshots of each memory location used. Contrast the differences between **DCB** and **SPACE**.

Task 5

- Declare the first 10 natural numbers starting at address **0x20000000** using **DCB**.

- Write an assembly program that adds these 10 numbers using the **Register Offset (Indexed)** addressing mode. (Hint: start by setting register **R1** to **0x20000000** and use a second register as an index).
- Rewrite the program using the **Post-Indexed** addressing mode.
- For the lab report: take a snapshot of memory and the final state of the registers. Discuss the difference between **Register Offset** and **Post-Indexed** addressing modes.

Lab Report

Task 1:

- **ARM Assembly** code with **Register Transfer Level (RTL)** and addressing modes for each instruction.
- Screenshots of each step of the program:
    - If the instruction modifies the register, screenshot the **Register Window**.
    - If the instruction modifies memory, screenshot the **Memory Window** displaying the correct address.

Task 2:

- **ARM Assembly** code.
- Screenshot of each step of the program.

Task 3:

- ASM code version 1 (**Immediate Offset**).
- ASM code version 2 (**Register Offset**).
- ASM code version 3 (**Post-Index**).
- Screenshot showing the values in memory at the correct address.

Task 4:

- Code.
- Memory snapshots.
- Comment on the difference between the assembler directives used (**DCB, DCD, SPACE**).

Task 5:

- Code.
- Memory snapshots.
- Comment on the difference between the assembler directives used.
- Discusses the differences between the different addressing modes used (**Immediate Offset vs. Register Offset vs. Post-Indexed**).