Lab 3: The Arithmetic Logic Unit

This lab covers new territory, with a new set of resources to become familiar with before coming to lab. Many of these resources are large documents; learning to scan for the needed information is a valuable skill that takes practice.

Purpose/Scope: The arithmetic logic unit (ALU) represents the heart of the processor. This lab will focus on the operating aspect of instructions, by observing how the ALU processes data and sets the condition codes to allow software branches to be made based on them.

Concepts: Assembly instructions performed by the ALU
How the Program Status Register Flags are set by the ALU

References:

- Microprocessors Course Reference Chapters 6-8
- Data on assembler directives and assembly language instructions (links shown on the following page)
- Signed and unsigned numbers

Deliverables

Answers to these questions are to be completed before coming to your scheduled lab period, for submittal at the beginning of the lab. Use the guidelines for homework. Also keep a record of your answers to use in the lab.

Part A: Assembler Directives

Using the link below, obtain descriptions for each of the following assembler directives.

https://sourceware.org/binutils/docs/as/index.html

1. .data
2. .text
3. .align
4. .type
5. .global
6. .end
7. .macro
8. .size

Part B: ARM Instructions and PSR Condition Code Flags

The ALU is where the processor does its work and modifies the core registers and special purpose (C)PSR (Program Status Register) depending on the operation performed. Each time an instruction is executed, general purpose and special registers may be modified. General purpose registers (R0-12) are modified if they are the destination register for the operation (adding, subtracting, shifting, etc.).

A special purpose register that may be affected is the PSR. In particular, the following bits may be affected by instructions that are allowed to affect the PSR. Only assembly instructions with an s-suffix can affect the PSR condition codes. These status bits are used in assembly programs to indicate various important conditions. The PSR is a word (32 bits) and the part we care about are the top 4 bits (most significant hex digit in the PSR word!).

Program Status Register Condition Code Flags

PSR bit # 31 - N Negative condition code flag - set when result is negative
PSR bit # 30 - Z Zero condition code flag - set when result is zero
PSR bit # 29 - C Carry condition code flag - set when carry/borrow is caused
PSR bit # 28 - V Overflow condition code flag - set when result overflows

It is important to understand that these flags indicate conditions that may require your assembly program to make decisions based on their value. They may indicate an error or a special case that must be handled.

Using the Cortex-M4 Instruction Set section of the Cortex M-4 Generic User Guide (found in the Blackboard "Resources – Lab Resources" tab) explain how the core registers are affected, and whether or not the condition code flags may be affected by the following instructions. Use "Find" or just look up the instruction.

Example Question and Answer:

Q: ADD R2, R1, R3

A: Add contents of R3 to R1 and place in R2. R1, R3, and condition code flags are not affected.

1. ADDS R2, R1, R3 // you must understand how this is different than the example above!
2. SUB R3, R5, R6 // you must understand which register is subtracted from which!
3. SUBS R1,#1
4. MOV R8, SP // SP is stack pointer - one of the core registers (aka R13)
5. MVN R2, #0
6. MVN R4, #0xFFFFFFFF
7. MVNS R6,#0x1
8. LSLS R1, R2, #3
9. LSR R4, R5, #12
10. AND R9, R2, #0x00FF
11. NOP

Part C: Data Calculations (signed and unsigned)

1. Calculate the sum of 27 and 35, with the answer in both decimal and hex.
2. Assume 8-bit values for the following questions:
    a. What is the range of values for unsigned representation in hex and decimal?
    b. What is the range of values for signed representation in hex and decimal?
3. Assume 8-bit values for the following questions. Be careful and use your answers from above in order to calculate these numbers. What I am looking for is what the 8-bit value will be even if it is incorrect because the number it is supposed to hold exceeds the maximum value that it can contain.
    a. 252+7 = ? (unsigned - hex and decimal)
    b. 125+5 = ? (unsigned - hex and decimal)
    c. 125+5 = ? (signed - hex and decimal)
    d. 5 - 8 = ? (signed - hex and decimal)
    e. 5 - 8 = ? (unsigned - hex and decimal)