L01GYKUV — COMPUTER ARCHITECTURE AND OPERATING SYSTEMS
Lab Practice 3

Stefano Di Carlo

November 6, 2023 - November 10, 2023

1 Exercise 1:

Compose an ARM assembly program to carry out the following operations:

Table 1: Flags and Register Values

Please report the hexadecimal representation of the values				
Updated Flags	R0 + R3		R4 - R2	
	R0	R3	R4	R2
Carry = 1				
Carry = 0				
Overflow				
Negative				
Zero				

- Initialize registers R3 and R4 with randomly generated signed values.
- Sum R0 and R3 (R0 + R3) and store the result in R2.
- Subtract R4 from R2 (R4 - R2) and store the result in R5.
- Configure the debug register window to set specific values for the program to update the following flags one at a time (whenever possible) to 1:
  - Carry
  - Overflow
  - Negative
  - Zero
- Document the chosen values in (Table 1).

2 Exercise 2:

Write two versions of a program to perform the following operations:

- Initialize registers R2 and R3 with randomly generated signed values.
- Compare the two registers:
  - If they differ, store the maximum of R2 and R3 in register R5.
  - Otherwise, perform a logical right shift of R3, sum R2, and store the result in R4.

First, solve it using a traditional assembly programming approach with conditional branches and then compare the execution times. For the second program, use conditional instruction execution. Record the execution times for both cases in the table below (Table 2).

NOTE: Report the number of clock cycles (cc) considering a CPU clock (clk) frequency of 16 MHz, as well as the simulation time in milliseconds (ms). Notice that the processor clock frequency is set up in the menu ”Options for Target: ’Target 1’”.

Table 2: Execution Time Report

	R2 == R3 [cc]	R2 == R3 [ms]	R2 != R3 [cc]	R2 != R3 [ms]
Traditional Approach				
Conditional Instruction Execution				

3 Exercise 3:

Develop a program to determine the number of leading zeros in a variable. The leading zeros are computed by counting the number of zeros starting from the most significant bit and stopping at the first 1 encountered. For example, in 0b**00000**101, there are 5 leading zeros. The variable to check is in R1. After the count, if the number of leading zeros is odd, perform the sum between R2 and R3. If the number of leading zeros is even, perform the difference between R2 and R3. In both cases, the result is placed in R4. Implement the ASM code for these operations:

- Determine whether the number of leading zeros of R1 is odd or even.
- As a result, compute the value of R4 as follows: - If the leading zeros are even, R4 is the difference between R2 and R3. - Otherwise, R4 is the sum of R2 and R3.

- Provide details on code size and execution time (with a 15 MHz clock) in the table below (Table 3).

Table 3: Execution Time Report

	Code Size [Bytes]	Execution Time [use the proper measurement unit]	
		IF R2 is even	Otherwise
Exercise 3			