

Lab #2

Condition Code Flags and Basic Flow Control

Sections 1, 2 and 3 of this exercise will count towards your final coursework mark for CS1021. Submit your solutions using Blackboard no later than 23:59pm on Tuesday 31st October 2017.

1 Condition Code Flags

- (a) In Tutorial #3, you were asked to consider each of the highlighted instructions in the ARM Assembly Language program below. Use the Flags project in μ Vision to verify the state of the N (Negative), Z (Zero), C (Carry) and V (oVerflow) flags after the execution of each of the highlighted instructions. Do the results concur with your prediction from the tutorial? You will need to use the μ Vision debugger to step through the program and inspect the state of the condition code flags immediately after executing each of the highlighted instructions. [Hint: This is a very typical exam question!]

1	LDR	R0, =0xC0001000
2	LDR	R1, =0x51004000
3	ADDS	R2, R0, R1 ; result? flags?
4	LDR	R3, =0x92004000
5	SUBS	R4, R3, R3 ; result? flags?
6	LDR	R5, =0x74000100
7	LDR	R6, =0x40004000
8	ADDS	R7, R5, R6 ; result? flags?
9	LDR	R1, =0x6E0074F2
10	LDR	R2, =0x211D6000
11	ADDS	R0, R1, R2 ; result? flags?
12	LDR	R1, =0xBF2FDD1E
13	LDR	R2, =0x40D022E2
14	ADDS	R0, R1, R2 ; result? flags?

- (b) In Tutorial #3, you were also asked to find pairs of 32-bit values which, when added together using the ADDS instruction, cause the following combinations of the conditions code flags to be set or cleared (1 or 0). Verify your answers from the tutorial by using the Flags2 project in μ Vision to load test values into R0 and R1 and add them using an ADDS R2, R0, R1 instruction. Do your test values give you the results you expected? [Hint: This is also a very typical exam question!]

- | | |
|----------------------------------|-----------------------------------|
| (i) N = 0; Z = 0; C = 0; V = 0 | (vi) N = 0; Z = 1; C = 0; V = 0 |
| (ii) N = 1; Z = 0; C = 0; V = 0 | (vii) N = 1; Z = 0; C = 0; V = 1 |
| (iii) N = 0; Z = 0; C = 1; V = 0 | (viii) N = 0; Z = 0; C = 1; V = 1 |
| (iv) N = 1; Z = 0; C = 1; V = 0 | (ix) N = 0; Z = 1; C = 1; V = 1 |
| (v) N = 0; Z = 1; C = 1; V = 0 | |

2 Division

- (a) Write and test an ARM Assembly Language program that will compute the quotient (“whole” part) and remainder of $a \div b$, where a and b are non-negative integers stored in R2 and R3 respectively. Your program should store the quotient in R0 and the remainder in R1.

A simple (but very inefficient) way to compute the quotient and remainder of $a \div b$ is to repeatedly subtract b from a until b is greater than a . Each time you subtract b from a , add 1 to the quotient. The value remaining in a at the end is the remainder. The following pseudo-code illustrates the approach:

```
remainder = a;
while (remainder >= b)
{
    quotient = quotient + 1;
    remainder = remainder - b;
}
```

- (b) What happens if you run your program with $b = 0$? If you have not already done so, try to extend your program so that it “works” when b is equal to zero.

3 Greatest Common Divisor

The greatest common divisor of two positive integers a and b is the largest integer that divides both a and b without a remainder. For example, the largest common divisor of 24 and 32 is 8.

Write and test an ARM Assembly Language program that will compute the greatest common divisor of two unsigned word-size values stored in R2 and R3. Your program must store the result in R0.

Rather than developing your own algorithm to compute the GCD, you may use the pseudo-code below as a starting point. After executing the following algorithm, the value in a will be the GCD.

```
while (a != b)
{
    if (a > b)
    {
        a = a - b;
    }
    else
    {
        b = b - a;
    }
}
```