CS1021 Lab 4

Logic and Shift Instructions

Q1 Write and ARM assembly language program to set R0 = 1 if R1 has an odd number of one bits (otherwise R0 = 0). For example, if R1 = 0x12345678, R0 = 1 as there are 13 one bits. Test your program with (i) 0x12345678 (ii) 0xF0F0F0F0 and (iii) 42.

Hint: One approach is to keep shifting R1 right one place and count the number of times the carry bit is set. Bit 0 of the count indicates if there is an odd or even number of one bits. An alternative approach is to EOR the top 16 bits of R1 with the low 16 bits, then EOR the second group of 8 bits with the low 8 bits and so on until bit 1 has been EORed with bit 0. Bit 0 then indicates if there is an odd or even number of one bits.

Q2 The ARM CPU does not have a divide instruction. A google search for "division algorithm" returns a Wikipedia entry with a number of division algorithms. The object of this question is to study the pseudo code for two of these algorithms and implement them in ARM Assembly language. The pseudo-code uses N for the numerator (dividend), D for the denominator (divisor), Q for the quotient and R for the remainder.

Algorithm 1


```
Q := 0; R := N
while R ≥ D do
    Q := Q + 1
    R := R − D
end
```


Algorithm 2

```
Q := 0                  -- initialize quotient and remainder to zero
R := 0
for i := n − 1 .. 0 do  -- where n is number of bits in N
    R := R << 1         -- left-shift R by 1 bit
    R(0) := N(i)        -- set the least-significant bit of R equal to bit i of the numerator
    if R ≥ D then
        R := R − D
        Q(i) := 1
    end
end
```

For both algorithms, assume that at entry N is n R0, D in R1 and at exit Q is in R0 and R in R1 and that unsigned arithmetic is used. Test your algorithms by calculating (i) 27 / 7 (ii) 444,444 / 23 and (iii) 33,554,432 / 506.

Algorithm 1 is very inefficient especially when D = 1. Estimate how long the algorithm would take (when run on the Keil simulator) to divide 0xFFFFFFFF by 1. For example, time (with a watch) how long it takes to divide 0xFFFFFF by 1 and then extrapolate. Why is algorithm 2 normally faster than algorithm 1.

You should also figure out how Algorithm 2 works. It follows the long division method you were taught at School. The algorithm calculates each digit of the quotient, in turn, starting with the most significant digit. In the case of decimal division, the digits are 0 t0 9 while in binary division the digits are 0 or 1 which are easier to determine.

This lab will count towards your final CS1021 coursework mark. Submit your solution via Blackboard no later than **9am on Fri 2-Nov-2018**. You should submit lab4.s and one .pdf file which gives a brief summary of your work and a clear statement as to whether your code works or not. The .pdf file must include evidence that your program works (eg. use the Windows Snipping tool to get screen shots of the Keil IDE which show that your program generates the correct results).