# UNIVERSITY OF DUBLIN
# TRINITY COLLEGE

## Faculty of Engineering, Mathematics and Science

### School of Computer Science & Statistics

**Integrated Computer Science, Year 1**
**Computer Science & Business, Year 1**
**Computer Science & Language, Year 1**

**Trinity Term 2014**

## CS1021 – Introduction to Computing I

**Thursday, 8 May 2014** **RDS, MAIN HALL** **09:30–11:30**

**Dr Jonathan Dukes**

---

**Instructions to Candidates**

Answer any **TWO out of THREE questions**.

All questions are marked out of 25.

Where you are asked to write an assembly language program you must provide suitable comments to explain your program, for example, in the form of pseudo-code comments.

**Permitted Materials**

An **ARM Instruction Set and Addressing Mode Summary** booklet is available on request.

Non-programmable calculators are permitted for this examination. You must indicate the make and model of your calculator on the front of your first answer book.

1. (a) Consider the following sequence of ARM Assembly Language instructions. For each highlighted instruction, **give the final value in the destination register** and **state whether each of the N (negative), Z (zero), C (carry) and V (overflow) flags is set or clear** after the execution of the instruction. Answers without a brief supporting explanation or calculation will receive zero marks. [6 marks]

```
1         LDR     R1, =0x00001000
2         LDR     R2, =0xC0000200
3         ADDS    R0, R1, R2              ; Condition Code Flags ?
4         LDR     R4, =0xD0000000
5         LDR     R5, =0x40000000
6         ADDS    R3, R4, R5              ; Condition Code Flags ?
7         LDR     R7, =0x80000000
8         LDR     R8, =0x80000000
9         ADDS    R6, R7, R8              ; Condition Code Flags ?
10        LDR     R9, =0x08000000
11        LDR     R10, =0x00000005
12        MOVS    R11, R9, LSL R10        ; Condition Code Flags ?
```

(b) Find pairs of 32-bit values which, when added together using the ADDS instruction, result in each of the following Condition Code flag states (0=clear, 1=set). Answers without a brief supporting explanation or calculation will receive zero marks.

(i) N = 0; Z = 0; V = 0; C = 0

(ii) N = 0; Z = 1; V = 0; C = 1

(iii) N = 1; Z = 0; V = 1; C = 0

(iv) N = 1; Z = 0; V = 0; C = 1

[6 marks]

(c) Provide an ARM Assembly Language program that will multiply the value in R1 by 17, storing the result back in R1. Your program may only use the following instructions: `MOV`, `ADD`, `SUB`, `RSB`, `B` and `B`*xx* (branch on condition *xx*). **More marks will be awarded for programs with fewer instructions. You may not use the `MUL` instruction or any other ARM multiplication instruction.** You must briefly explain your approach. [3 marks]

*question 1 continued on next page ...*

*... question 1 continued from previous page*

(d) Translate the pseudo-code algorithm below into ARM Assembly Language.

Assume that `a`, `b`, `s`, `t`, and `r` are unsigned word-size values stored in R0, R1, R2, R3 and R4 respectively.

Recall that | is the bitwise OR operation (`ORR`), & is the bitwise AND operation (`AND`), << is the logical shift left operation (`LSL`) and >> is the logical shift right operation (`LSR`).

```
for (s = 0; ((a | b) & 1) == 0); s = s + 1)
{
  a = a << 1;
  b = b << 1;
}

while ((a & 1) == 0)
{
  a = a >> 1;
}

do {
  while ((b & 1) == 0)
  {
    b = b >> 1;
  }

  if (a > b)
  {
    t = b;
    b = a;
    a = t;
  }

  b = b - a;
} while (b != 0);

r = a << s;
```

[10 marks]

2. (a) Provide an ARM Assembly Language program that will add two 64-bit integer values stored in memory. Assume the values are stored in memory at the addresses contained in R1 and R2. Assume that the result is also to be stored in memory, at the address contained in R0. State any assumptions you make about the way the values are stored in memory. [3 marks]

   (b) Write an ARM Assembly Language program that will compute the maximum, minimum and average of a sequence of values stored in memory. Assume that R3 contains the address in memory of the first value in the sequence and R4 contains the number of values in the sequence. Your program should store the maximum value in R0, the minimum value in R1 and the average of the values in R2. [7 marks]

   (c) Design and write an ARM Assembly Language program that will count the number of **non-unique** values (i.e. values that occur more than once) in a sequence of word-size values stored in memory. For example, in the sequence below there are three non-unique values: 2, 4 and 7 so your program should store the result 3.

   4,1,4,4,2,5,2,7,3,7,7,8,7,9

   Assume that R1 contains the address in memory of the first value in the sequence and R2 contains the number of values in the sequence. Your program should store its result in R0.

   Your answer must include:

   (i) an explanation of your approach and [4 marks]

   (ii) your ARM Assembly Language program. [11 marks]

3. (a) Design and write an ARM Assembly Language program that will determine whether a NULL-terminated ASCII string stored in memory is a palindrome. A palindrome is a string that reads the same in either direction. For example, **"Able was I ere I saw Elba"** is a well-known palindrome:

Your program should be **case insensitive** (i.e. Your program should interpret 'e' as being equal to 'E' when determining whether the string is a palindrome). Your program should store the value 1 in R0 if the string is a palindrome and 0 if it is not. Assume that R1 contains the address in memory of the first character in the string.

Your answer must include:

(i) an explanation of your approach and [3 marks]

(ii) your ARM Assembly Language program. [9 marks]

(b) Design and write an ARM Assembly Language program that will count the number of occurrences of a string, *A*, in another string, *B*. The occurrences may overlap. In the example below there are three occurrences of *A* in *B*.

string *A* **aba**

string *B* **dababacabadaba**

Assume that both strings are NULL-terminated ASCII strings, R1 contains the address in memory of the first character in *A* and R2 contains the address in memory of the first character in *B*. Your program should store the number of occurrences of *A* in *B* in R0.

Your answer must include:

(i) an explanation of your approach and [3 marks]

(ii) your ARM Assembly Language program. [10 marks]