CS1021-1

# TRINITY COLLEGE DUBLIN
# THE UNIVERSITY OF DUBLIN

## Faculty of Engineering, Mathematics and Science

## School of Computer Science and Statistics

**Integrated Computer Science**
**Computer Science & Business**
**Computer Science & Language**
**Year 1 Annual Examination**

**Trinity Term 2015**

## CS1021 – Introduction to Computing I

**Wednesday, 29 April 2015** **LUCE LOWER** **14:00 – 16:00**

**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) Write an ARM Assembly Language program to evaluate the expression $2x^3 - x^2 + 5x - 12$. Assume $x$ is stored in R1. Store your result in R0. [3 marks]

(b) 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]

```
        LDR     R7, =0xFFFFFFFE
        LDR     R8, =0x00000004
        ADDS    R6, R7, R8          ; Condition Code Flags? Value in R6?

        LDR     R10, =0x70000000
        SUBS    R9, R10, R10        ; Condition Code Flags? Value in R9?

        LDR     R1, =0x80000000
        LDR     R2, =0x90000000
        ADDS    R0, R1, R2          ; Condition Code Flags? Value in R0?

        LDR     R4, =0x70000000
        LDR     R5, =0x90000000
        ADDS    R3, R4, R5          ; Condition Code Flags? Value in R3?
```

(c) Show how you would perform each of the following bit-manipulation operations using ARM Assembly Language:

(i) Clear (i.e. change to 0) the middle two bytes of the value in R1

(ii) Invert (i.e. change 0s to 1s and 1s to 0s) every second bit of the value in R2, beginning with the least significant bit.

(iii) Swap the position of the middle two bytes of the value in R3.

[4 marks]

*question 1 continued on next page ...*

*... question 1 continued from previous page*

(d) Translate the pseudo-code shown below into ARM Assembly Language. Assume that a, b, c, i and r are stored in R0, R1, R2, R3 and R4 respectively. [6 marks]

```
c = 1;
r = 1;
for (i = 1; i < 32; i++) {
  b = a & 1;
  a = a >> 1;
  if (b == (a & 1)) {
    c = c + 1;
  } else {
    if (c > r) {
      r = c;
    }
    c = 1;
  }
}
if (c > r) {
  r = c;
}
```

(e) Provide an ARM Assembly Language program that will multiply the value in R1 by the value in R2, storing the result in R0. Your program may only use the following instructions: `MOV`, `ADD`, `SUB`, `B`, `B`*xx* (branch on condition *xx*) and `CMP`. (You are allowed to use the "with Status" variant of these instructions, e.g. MOVS.) **More marks will be awarded for more efficient programs.** You may not use the `MUL` instruction or any other ARM multiplication instruction. [6 marks]

2. (a) Design and write an ARM Assembly Language program that will determine the most frequently occurring letter in a NULL-terminated ASCII string. Your program should be **case insensitive**. If there is a tie for the most frequently occurring letter, the letter closest to the start of the alphabet wins.

   Your program should store the ASCII code for the most frequently occurring letter in R0 and the number of times that it occurs in R1. Assume that R2 contains the start address of the string and that R3 contains the address of a region of memory that you may use for your own purposes.

   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 convert a NULL-terminated string to "CamelCase". For the purpose of this exercise, a CamelCase string is a string that contains only alphabetic characters, no spaces and each word begins with a capital letter. "IntroductionToComputing" is an example of a CamelCase string.

   Assume that R1 contains the start address of the original string. Store the new CamelCase string in memory beginning at the address contained in R0. You may also assume that the original string contains only alphabetic characters and spaces.

   Your answer must include:

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

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

3. (a) Translate the following pseudo-code program into ARM Assembly Language. Assume the assignment of registers to variables shown in the table below. The syntax Memory.Halfword[address] represents a halfword (2-byte) memory access at the specified address. START is a constant memory address that you may refer to using the label START. [10 marks]

```
found == FALSE;
while (i < j && found == FALSE) {
  k = i + ((j - i) >> 1);
  test = Memory.Halfword[START + (k * 2)];
  if (test == search) {
    found = TRUE;
  } else if (test < search) {
    i = k + 1;
  } else {
    j = k - 1;
  }
}
```

| **variable** | found | i | j | k | test | search |
|---|---|---|---|---|---|---|
| **register** | R0 | R1 | R2 | R3 | R4 | R5 |

(b) Assume two sequences of 32-bit values are stored in memory. The following ARM Assembler directives illustrate how the sequences are arranged in memory.

```
ASize DCD 10                                ; Size of Sequence A
ASeq  DCD 2, 8, 3, 4, 1, 5, 7, 10, 4, 2     ; Sequence A
BSize DCD 10                                ; Size of Sequence B
BSeq  DCD 5, 1, 10, 7, 4, 8, 3, 4, 2, 2     ; Sequence B
```

Design and write an ARM Assembly Language program that will determine whether one of the sequences could be formed by rearranging the elements from the other sequence (i.e. the two sequences must include not only the same elements but also the same number of each element). Your answer must include:

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

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