UNIVERSITY OF DUBLIN
TRINITY COLLEGE

Faculty of Engineering, Mathematics and Science

School of Computer Science and Statistics

BA (Mod) Business and Computing
BA (Mod) Computer Science, Linguistics and a Language
Junior Freshman Examination

Trinity Term 2012

CS1021 – Introduction to Computing I

2nd May 2012 **RDS MAIN** **14:00 – 16:00**

Dr Jonathan Dukes

Instructions to candidates

- Answer any **THREE out of FOUR questions**.
- Each question is worth 40 marks.
- The result for this examination will be calculated as a percentage of 120 marks.
- 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 will be provided with this examination paper.
- Non-programmable calculators are permitted for this examination. Please indicate the make and model of your calculator on the front of your first answer book.

1 (a) The ARM Assembly Language program shown below results in three errors when assembled. The resulting assembler output is also shown below. Briefly explain each of the errors and show how to correct them.

(6 marks)

Program:

```
1               AREA    BadCode, CODE, READONLY
2               IMPORT  main
3               EXPORT  start
4
5     start
6               MOV     R1, #511
7               MUL     R0, R1, #5
8               MOV     R2, #20
9               MUL     R0, R0, R2
10    stop      B       stop
11
12              END
```

Assembler output:

```
Build target 'Simulator'
assembling BadCode.s...

BadCode.s(6): error: A1510E: Immediate 0x000001FF cannot
              be represented by 0-255 and a rotation
BadCode.s:     6 00000000   MOV  R1, #511

BadCode.s(7): error: A1647E: Bad register name symbol,
              expected Integer register
BadCode.s:     7 00000004   MUL  R0, R1, #5
BadCode.s:                                ^

BadCode.s(9): error: A1477E: This register combination
              results in UNPREDICTABLE behaviour
BadCode.s:     9 00000008   MUL  R0, R0, R2

BadCode.s:    3 Errors, 0 Warnings
Target not created
```

(b) Convert the following signed decimal values to their binary equivalents using an 8-bit 2's Complement representation. (**Note:** marks will be awarded for showing **how** you have performed the conversion. Answers consisting of just the final result will receive zero marks.)

(i) +10 **(1 mark)**

(ii) -14 **(2 marks)**

continued on next page ...

1 ... continued from previous page

(c) Consider the following sequence of ARM Assembly Language instructions. For each of the highlighted **ADDS** and **SUBS** instructions, **give the final value in the destination register in binary or hexadecimal form 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. **You must explain your answers.** Assume the flags are all clear before the execution of the first instruction.

(9 marks)

```
        LDR     R0, =0xC0001000
        LDR     R1, =0x51004000
        ADDS    R2, R0, R1              ; result? flags?

        LDR     R3, =0x92004000
        SUBS    R4, R3, R3              ; result? flags?

        LDR     R5, =0x74000100
        LDR     R6, =0x40004000
        ADDS    R7, R5, R6              ; result? flags?
```

(d) Design and write an ARM Assembly Language program that will determine the number of contiguous sequences of 1s (set bits) in an arbitrary 32-bit value stored in **R1**. Your program should store the result in **R0**. For example, given the following 32-bit value in **R1**, your program should store the result **5** in **R0** as there are five contiguous sequences of 1s.

11100110000000001000011000000111

Your answer must include:

(i) an explanation of your approach to solving the problem and

(6 marks)

(ii) an ARM Assembly Language listing for your program with adequate comments.

(16 marks)

2 (a) Translate the following pseudo-code extract into ARM Assembly Language. Assume that **a**, **b**, **c**, **N** and **address** are unsigned values stored in **R0**, **R1**, **R2**, **R3** and **R4** respectively. The syntax **Memory.Word[address]** is intended to represent loading the word-size value from memory at the specified **address**.

```
a = 0
c = 0
while (a < N)
{
     address = b + (a * 4)
     c = c + Memory.Word[address]
     a = a + 1;
}
```

(5 marks)

(b) *Pilish* is a form of constrained writing in which the lengths of consecutive words correspond to the digits of the value π (pi). The following sentence is a well-known example of Pilish, as the lengths of consecutive words are 3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5, 8, 9, 7 and 9, which are the same as the first fifteen digits of π.

> *How I need a drink, alcoholic in nature, after the heavy lectures involving quantum mechanics!*

Design and write an ARM Assembly Language program that will determine whether a NULL-terminated ASCII string is valid Pilish. Assume that the string can contain any ASCII characters but that words contain only alphabetic characters (i.e. **a** ... **z** and **A** ... **Z**). Words may be separated by any number of non-alphabetic characters. Words with ten or more letters correspond to the digit 0.

Assume that **R1** contains the address of the start of the string, **R2** contains the address of the start of a sequence of byte-size values containing a sufficient number of the digits of π to check any given string. Store 1 in **R0** if the string is valid Pilish or zero otherwise.

Your answer must include:

(i) an explanation of your approach to solving the problem and

(9 marks)

(ii) an ARM Assembly Language listing for your program with adequate comments.

(26 marks)

3 (a) Consider the ARM Assembly Language listing below, which also shows the address at which each assembled instruction will be stored in memory.

line	address	label	instruction	
1	00000000	wh1	CMP	r2, #1
2	00000004		BLS	endwh1
3	00000008		MUL	r0, r2, r0
4	0000000C		SUB	r2, r2, #1
5	00000010		B	wh1
6		endwh1		
7	00000014	stop	B	stop

(i) Calculate the branch target offset that would be encoded in each of the three highlighted branch instructions and, in each case, show how you computed your result.

(6 marks)

(ii) Referring to the *"ARM Instruction Set and Addressing Mode Summary"* booklet that accompanies this examination paper, provide the 32-bit machine code instruction that the assembler would produce for each of the three highlighted branch instructions.

(6 marks)

(b) Design and write an ARM Assembly Language program that will accept an ASCII string representing a simple mathematical expression and return the result of the evaluation of the expression as a binary value (i.e. the result does not need to be converted to an ASCII string).

Valid mathematical expressions consist of a sequence of ASCII characters representing a decimal value using the digits '0' ... '9', followed by '+' or '-', followed by a second sequence of ASCII characters representing a second decimal value. The strings are NULL (zero) terminated.

The following are examples of valid strings:

`"25+15"` `"199+0"` `"300-50"`

Your program should return the results 40, 199 and 250 respectively for the examples above. Assume that the start address of the string is stored in **R1**. Your program should store the result in **R0**.

Your answer must include:

(i) an explanation of your approach to solving the problem and

(7 marks)

(ii) an ARM Assembly Language listing for your program with adequate comments.

(21 marks)

4 The *Imaginary Lottery Company* sells lottery tickets for a weekly draw. Players choose six numbers between 1 and 32 when purchasing a ticket. There are prizes for tickets that match any four, five or six of the numbers drawn.

Design and write an ARM Assembly Language program that will determine the number of tickets that match four numbers, five numbers and six numbers. (i.e. Your program should produce three result values for "match four" tickets, "match five" tickets and "match six" tickets.) You may store these results in a location (register or memory) of your choosing.

Assume that the numbers chosen for each ticket are stored consecutively in memory, beginning at the address contained in **R0**. The number of tickets sold is stored in **R1**. The lottery numbers drawn are stored in memory beginning at the address contained in **R2**. All lottery numbers are stored as simple byte-size values (i.e. they are **not** ASCII characters). The ARM Assembly Language extract below illustrates the structure of the data in memory.

```
            AREA    Lotto, CODE, READONLY
            ...     ...

start       LDR     R0, =tickets
            LDR     R1, =3
            LDR     R2, =draw
            ...     ...

            AREA    LottoData, DATA, READWRITE

tickets     DCB     4,17,21,24,25,29        ; first ticket
            DCB     2,4,5,19,20,30          ; second ticket
            DCB     10,12,24,27,29,31       ; third ticket

draw        DCB     3,4,5,19,25,30          ; draw result

            END
```

Note: Although the numbers for each lottery ticket are shown on a separate line above, the first number of the second ticket is stored in the byte immediately after the last byte of the first ticket, and so on for successive tickets. There is no need to separate the tickets with any value, as we know there are exactly six numbers on every ticket.

Your answer must include:

(i) an explanation of your approach to solving the problem and **(10 marks)**

(ii) an ARM Assembly Language listing for your program with adequate comments. **(30 marks)**