UNIVERSITY OF DUBLIN
TRINITY COLLEGE

Faculty of Engineering, Mathematics and Science

School of Computer Science and Statistics

Integrated Computer Science Programme **Trinity Term 2012**
Junior Freshman Examination

CS1021 & CS1022 – Introduction to Computing I & II

2nd May 2012 **SPORTS CENTRE** **14:00 – 17:00**

Dr Jonathan Dukes

Instructions to candidates

- Answer any **FOUR out of SIX questions**.
- Each question is worth 25 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 and fix each of the errors.

(3 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 **(1 mark)**

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.

(8 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?

        LDR     R8, =0xA4001000
        LDR     R9, =0x61000000
        SUBS    R10, R8, R9             ; 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

(3 marks)

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

(9 marks)

2 (a) Translate the following pseudo-code extract into ARM Assembly Language. Assume that **a**, **b**, **c** and **N** are unsigned values and that **M** is an array of word-size values beginning at an address in memory that may be referred to using the label **arrM**.

```
a = 0
while (a < N)
{
     c = M[a];
     if (c & 0x00000001 == 0x00000001)
     {
          b = b + c
     }
     a = a + 1;
}
```

(4 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 subroutine 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.

Your answer must include:

(i) an explanation of your approach to solving the problem, **(5 marks)**

(ii) a description of the interface to your subroutine and **(1 mark)**

(iii) an ARM Assembly Language listing for your subroutine with adequate comments.

(15 marks)

3 (a) Consider each of the following independent ARM Assembly Language instructions. State precisely what effect executing each of the instructions would have and the memory address that would be accessed. Assume that `R1=0x40002000` and `R2=0x00000080`.

(i) `LDR R0, [R1, R2, LSL #2]` **(1 mark)**

(ii) `LDR R0, [R1, #4]!` **(1 mark)**

(iii) `LDRSH R0, [R1, R2]` **(1 mark)**

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

Design and write an ARM Assembly Language subroutine that will determine the number of tickets that match four numbers, five numbers and six numbers. (i.e. Your subroutine should return three result values for "match four" tickets, "match five" tickets and "match six" tickets.).

Assume that the numbers chosen for each ticket, the total number of tickets sold and the six lottery numbers drawn are stored in memory as shown below. The data should be passed to your subroutine using an appropriate interface.

```
            AREA    LottoData, DATA, READWRITE

Ntickets    DCW     3                   ; number of tickets

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:

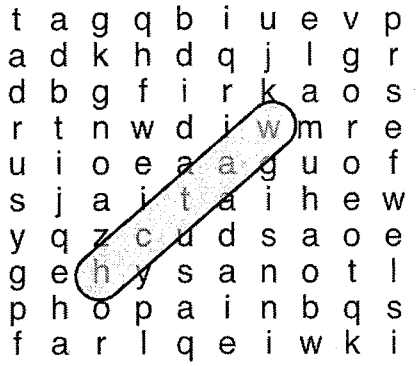
(i) an explanation of your approach to solving the problem, **(5 marks)**

(ii) a description of the interface to your subroutine and **(1 mark)**

(iii) an ARM Assembly Language listing for your subroutine with adequate comments.

(16 marks)

4 A word search puzzle consists of a two-dimensional grid of letters and a list of words contained within the grid. Each word may be contained in the grid vertically, horizontally or diagonally in any direction (upwards, downwards, left-to-right, right-to-left, etc.). The example below shows the location of the word "watch" in a word search grid.



Given a square two-dimensional array of byte-size ASCII characters representing a Word Search grid and a NULL-terminated ASCII string containing a single word, design and write an ARM Assembly Language subroutine that will determine whether the word is contained in the grid in any direction and at any location.

You may assume that both the grid and the word contain only lower-case alphabetic characters. You should include the dimension of the grid as one of the parameters passed to your subroutine.

Your answer must include:

(i) an explanation of your approach to solving the problem, **(7 marks)**

(ii) a description of the interface to your subroutine and **(1 mark)**

(iii) an ARM Assembly Language listing for your subroutine with adequate comments. **(17 marks)**

5 (a) Show how you would represent each of the following two floating point values using the IEEE754 single-precision format.

(i) 24.25_{10} **(1½ marks)**
(ii) 9.625_{10} **(1½ marks)**

(b) Show how the two floating point numbers from part (a) above would be added. Explain each step of the addition. You do not need to consider rounding.

(4 marks)

(c) Write an ARM Assembly Language subroutine to **normalize** and then **encode** a binary floating-point value. The sign, exponent and fraction elements of the value should be passed as three parameters to the subroutine, which should return a single-precision IEEE-754-encoded value. You may ignore rounding.

Your answer must include:

(i) an explanation of your approach to solving the problem, **(4 marks)**

(ii) a description of the interface to your subroutine and **(1 mark)**

(iii) an ARM Assembly Language listing for your subroutine with adequate comments.

(13 marks)

6 (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.

(3½ 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.

(3½ marks)

(b) Suppose you are required to implement a simple stopwatch based on an NXP LPC2468 microcontroller. The stopwatch has two push-buttons, BUTTON1 and BUTTON2, connected to the EINT0 and EINT1 external interrupt lines.

The stopwatch starts when BUTTON1 is pressed. Pressing BUTTON1 again causes the stopwatch to pause counting. A further press of BUTTON1 should cause the stopwatch to resume counting.

Pressing BUTTON2 while the stopwatch is paused should cause the time to be reset to zero. Pressing BUTTON2 while the stopwatch is counting should cause the current time to be recorded in a "lap times" log.

Design and write an ARM Assembly Language program to implement the stopwatch. The lap times log should be stored as an array in memory. The base address of the array is `0xA1000000`. The first time should be stored at the start of the array and further times should be stored consecutively in the array.

Your solution must include (i) an explanation of how you would initialise the system and configure any timer devices or external interrupts required and (ii) an assembly language listing for any interrupt handlers required by your design. You must provide adequate comments for any assembly language code that you write.

(18 marks)