# UNIVERSITY OF DUBLIN
# TRINITY COLLEGE

## Faculty of Engineering, Mathematics and Science

### School of Computer Science & Statistics

**JF Computer Science & Business**
**JF Computer Science & Language**

**Trinity Term 2013**

## CS1021 – Introduction to Computing I

**Thursday, 9 May 2013** **Sports Centre** **9:30–11:00 (1½ Hours)**

**Dr Jonathan Dukes**

---

**Instructions to Candidates**

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

Each question is worth 50 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**

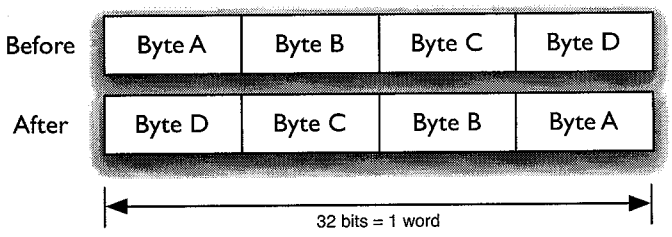
An **ARM Instruction Set and Addressing Mode Summary** booklet will be provided with this examination paper.

Non-programmable calculators are permitted for this examination. 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 of the highlighted 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. Answers without an explanation will receive zero marks. Assume the flags are all clear before the execution of the first instruction. [12 marks]

```
1        LDR     R0, =0x80004040
2        LDR     R1, =0xB0000200
3        ADDS    R2, R1, R0      ; Condition Code Flags?
4
5        LDR     R4, =0x40000000
6        LDR     R5, =0x50000000
7        SUBS    R6, R4, R5      ; Condition Code Flags?
8
9        LDR     R3, =0x40000000
10       MOVS    R3, R3, LSL #2  ; Condition Code Flags?
```

(b) Provide an ARM Assembly Language program to reverse the order of the bytes of the word in R0, as illustrated below.

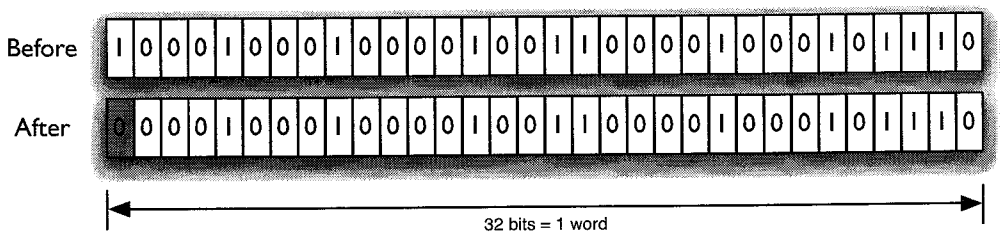



[14 marks]

*question continued on next page ...*

*... question continued from previous page*

(c) Design and write an ARM Assembly Language program that will modify the most-significant bit of the word in R0 such that the number of set bits (1's) in the word is even. For example, the word shown below has an odd number of set bits before executing the program. After executing the program and inverting the most-significant bit, the number of set bits is even. (**Hint:** Begin by counting the number of set bits in the word.)




[24 marks]

2. (a) Design and write an ARM Assembly Language program that will convert an ASCII string to Proper Case. In a Proper Case string, the first letter of each word is capitalised and all other letters are lower case.

"This Is An Example Of A Proper Case String"

The string is stored in memory beginning at the address contained in R0. The string contains only alphabetic characters and spaces and is NULL-terminated. Assume that a word is defined to be a contiguous sequence of alphabetic characters and **words may be separated by any number of spaces**.

Your answer must include:

(i) an explanation of your approach and [8 marks]

(ii) your ARM Assembly Language program. [24 marks]

(b) In a Scrabble®-like game, players form words and each word is awarded a score that is the sum of the "values" of the letters used to form the word. For example, the word "banana" would be awarded a score of 8 because "b" has the value 3, "a" has the value 1 and appears three times and "n" has the value 1 and appears twice. (i.e. $3+1+1+1+1+1=8$).

Design and write an ARM Assembly Language program that will compute the score for a single word. The word is stored in memory at the address contained in R1. The word is stored as a NULL-terminated ASCII string containing only UPPER CASE alphabetic characters. The score value of each letter in the alphabet is also stored in memory as a sequence of 26 byte size values beginning at the address contained in R2. The first byte is the score value of "A", the second byte is the score value of "B", etc. Your program should store the score for the word in R0.

Your answer must include:

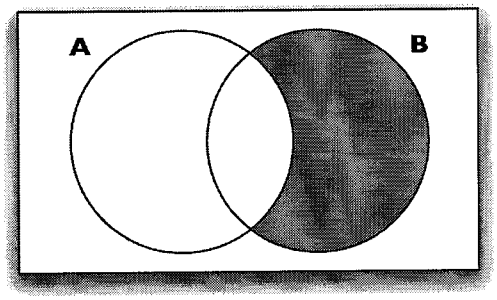
(i) an explanation of your approach and [4 marks]

(ii) your ARM Assembly Language program. [14 marks]

3. (a) Assume two mathematical *sets*, $A$ and $B$, containing 32-bit unsigned integers are stored in memory. Assume also that space in memory has been set aside to store a third set, $C$. The following ARM Assembler directives illustrate how the sets are arranged in memory.

```
1 ASize   DCD    8                  ; Number of elements in Set A
2 AElems  DCD    7,20,9,17,3,2,23,13 ; Elements in Set A
3
4 Bsize   DCD    6                  ; Number of elements in Set B
5 Belems  DCD    6,13,11,2,25,10    ; Elements of Set B
6
7 Csize   DCD    0                  ; Number of elements in Set C
8 Celems  SPACE  14*4               ; Space for elements of Set C
```

Design and write an ARM Assembly Language program to populate the third set, $C$, such that $C$ is the *relative complement* of $A$ in $B$. The *relative complement* of $A$ in $B$ contains all the elements of $B$ that are not also contained in $A$. In the Venn diagram below, the relative complement of $B$ in $A$ has been shaded.




Your program must also store the size of $C$ in memory at the location labelled `Csize` above.

Your answer must include:

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

(ii) your ARM Assembly Language program. [20 marks]

*question continued on next page ...*

*... question continued from previous page*

(b) Assume a sequence of word-size signed integers is stored in memory at an address contained in R1. The number of integers in the sequence is stored in R2. Design and write an ARM Assembly Language program that will count the number of sequences of contiguously increasing values. For example, the sequence below contains two sequences of contiguously increasing values, which have been underlined.

12, 9, 7, 5, 6, 10, 17, 19, 16, 12, 11, 11, 19, 21

Your answer must include:

(i) an explanation of your approach and [6 marks]

(ii) your ARM Assembly Language program. [20 marks]

# UNIVERSITY OF DUBLIN
## TRINITY COLLEGE

### Faculty of Engineering, Mathematics and Science

### School of Computer Science & Statistics

**Integrated Computer Science, Year 1** **Trinity Term 2013**

## CS1021 & CS1022 – Introduction to Computing I & II

**Thursday, 9 May 2013** **Sports Centre** **9:30–12:30 (3 hours)**

**Dr Jonathan Dukes**

---

**Instructions to Candidates**

**Answer TWO questions from Section A and TWO questions from Section B.**
Section A is the examination for CS1021 and Section B is the examination for CS1022.
It is recommended that you spend no more than 90 minutes answering each Section.
Use separate answer booklets for each Section.
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. You must indicate the make and model of your calculator on the front of your first answer book.

# Section A

## CS1021 – Introduction to Computing I

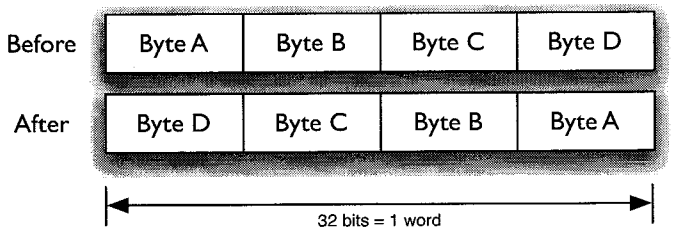
**Answer any TWO out of THREE questions from this Section**

It is recommended that you spend no more than 90 minutes answering this Section

1. (a) Consider the following sequence of ARM Assembly Language instructions. For each of the highlighted 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. Answers without an explanation will receive zero marks. Assume the flags are all clear before the execution of the first instruction. [6 marks]

```
1        LDR     R0, =0x80004040
2        LDR     R1, =0xB0000200
3        ADDS    R2, R1, R0      ; Condition Code Flags?
4
5        LDR     R4, =0x40000000
6        LDR     R5, =0x50000000
7        SUBS    R6, R4, R5      ; Condition Code Flags?
8
9        LDR     R3, =0x40000000
10       MOVS    R3, R3, LSL #2  ; Condition Code Flags?
```

(b) Provide an ARM Assembly Language program to reverse the order of the bytes of the word in R0, as illustrated below.

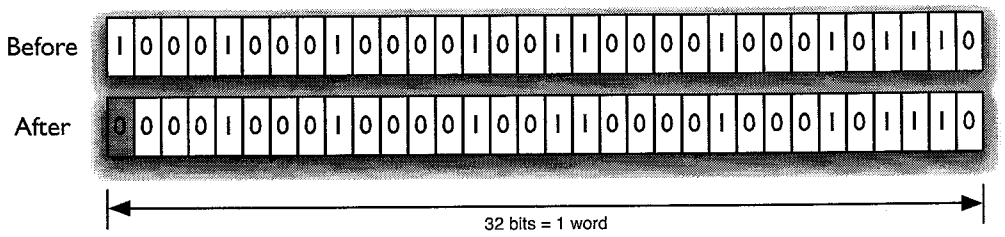



[7 marks]

*question continued on next page ...*

*... question continued from previous page*

(c) Design and write an ARM Assembly Language program that will modify the most-significant bit of the word in R0 such that the number of set bits (1's) in the word is even. For example, the word shown below has an odd number of set bits before executing the program. After executing the program and inverting the most-significant bit, the number of set bits is even. (**Hint:** Begin by counting the number of set bits in the word.)




[12 marks]

2. (a) Design and write an ARM Assembly Language program that will convert an ASCII string to Proper Case. In a Proper Case string, the first letter of each word is capitalised and all other letters are lower case.

"This Is An Example Of A Proper Case String"

The string is stored in memory beginning at the address contained in R0. The string contains only alphabetic characters and spaces and is NULL-terminated. Assume that a word is defined to be a contiguous sequence of alphabetic characters and **words may be separated by any number of spaces.**

Your answer must include:

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

(ii) your ARM Assembly Language program. [12 marks]

(b) In a Scrabble®-like game, players form words and each word is awarded a score that is the sum of the "values" of the letters used to form the word. For example, the word "banana" would be awarded a score of 8 because "b" has the value 3, "a" has the value 1 and appears three times and "n" has the value 1 and appears twice. (i.e. $3+1+1+1+1+1=8$).

Design and write an ARM Assembly Language program that will compute the score for a single word. The word is stored in memory at the address contained in R1. The word is stored as a NULL-terminated ASCII string containing only UPPER CASE alphabetic characters. The score value of each letter in the alphabet is also stored in memory as a sequence of 26 byte size values beginning at the address contained in R2. The first byte is the score value of "A", the second byte is the score value of "B", etc. Your program should store the score for the word in R0.

Your answer must include:

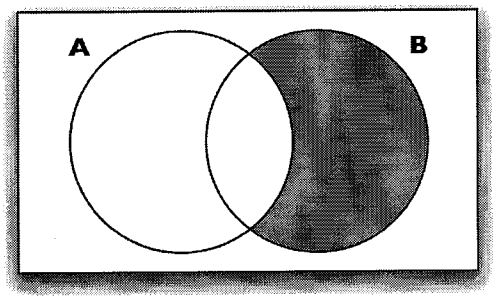
(i) an explanation of your approach and [2 marks]

(ii) your ARM Assembly Language program. [7 marks]

3. (a) Assume two mathematical *sets*, $A$ and $B$, containing 32-bit unsigned integers are stored in memory. Assume also that space in memory has been set aside to store a third set, $C$. The following ARM Assembler directives illustrate how the sets are arranged in memory.

```
1 ASize   DCD    8                  ; Number of elements in Set A
2 AElems  DCD    7,20,9,17,3,2,23,13 ; Elements in Set A
3
4 Bsize   DCD    6                  ; Number of elements in Set B
5 Belems  DCD    6,13,11,2,25,10    ; Elements of Set B
6
7 Csize   DCD    0                  ; Number of elements in Set C
8 Celems  SPACE  14*4               ; Space for elements of Set C
```

Design and write an ARM Assembly Language program to populate the third set, $C$, such that $C$ is the *relative complement* of $A$ in $B$. The *relative complement* of $A$ in $B$ contains all the elements of $B$ that are not also contained in $A$. In the Venn diagram below, the relative complement of $B$ in $A$ has been shaded.




Your program must also store the size of $C$ in memory at the location labelled `Csize` above.

Your answer must include:

(i) an explanation of your approach and [2 marks]

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

*question continued on next page ...*

*... question continued from previous page*

(b) Assume a sequence of word-size signed integers is stored in memory at an address contained in R1. The number of integers in the sequence is stored in R2. Design and write an ARM Assembly Language program that will count the number of sequences of contiguously increasing values. For example, the sequence below contains two sequences of contiguously increasing values, which have been underlined.

12, 9, 7, 5, 6, 10, 17, 19, 16, 12, 11, 11, 19, 21

Your answer must include:

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

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

# Section B

## CS1022 – Introduction to Computing II

**Answer any TWO out of THREE questions from this Section**

It is recommended that you spend no more than 90 minutes answering this Section

4. (a) Consider the ARM Assembly Language extract below. Describe in detail the operation performed by each of the instructions. In the case of each instruction, you must state the memory address that is accessed and the new value contained in any register that is modified. You must also provide a memory diagram to illustrate the effect of any instruction that modifies the system stack (or system stack pointer).

Assume the following initial values in R1, R2 and SP:

`R1=0xA1001000, R2=0x00000008, SP=0xA1000700`

```
1       STMFD   SP!, {R4-R6, LR}
2       LDR     R4, [R1, R2, LSL #2]
3       STR     R4, [SP, #-4]!
```

[5 marks]

(b) Write an ARM Assembly Language subroutine, **swap(array, i, j)** that will swap two elements in a 1-dimensional array of word-size integers. Your subroutine should accept the address of the array and the indices of the two elements to be swapped as parameters.

Your answer must include:

(i) a description of an appropriate interface for your subroutine and [1 marks]

(ii) an ARM Assembly Language listing for your subroutine [4 marks]

*question continued on next page ...*

*... question continued from previous page*

(c) Translate the pseudo-code shown below into an ARM Assembly Language subroutine. You must make use of your **swap(array, i, j)** subroutine from part (b).

```
sort(array, N)
{
    do
    {
        swapped = false;
        for (i = 1; i < N; i++)
        {
            if (array[i-1] > array[i])
            {
                swap(array, i-1, i);
                swapped = true;
            }
        }
    } while (swapped);
}
```

Your answer must include:

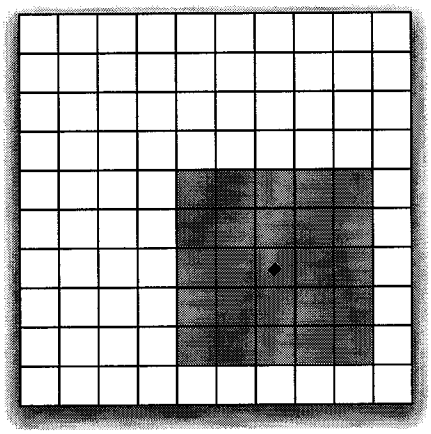
(i) a description of an appropriate interface for your subroutine and [1 marks]

(ii) an ARM Assembly Language listing for your subroutine [9 marks]

(d) Subroutines usually save the contents of the Link Register on the system stack at the start of the subroutine and restore it at the end. Explain, using the **sort()** and **swap()** subroutines from parts (b) and (c) as an example, why this is necessary. Your answer must also explain in detail how the *Branch and Link* (BL) instruction is used to invoke a subroutine and how a subroutine returns to the instruction immediately following the instruction that originally invoked it. [5 marks]

5. Consider a grey-scale image stored in memory as a two-dimensional array of half-word unsigned integers. Each unsigned integer represents the brightness of the corresponding pixel in the image.

   A "blurring" effect can be applied to the image in a simple way by replacing the brightness of each pixel with the average brightness of the pixels surrounding it. For example, in the figure below, the brightness of the pixel marked ♦ is replaced with the average brightness of the shaded pixels. (Note that the original brightness of the pixel marked ♦ is also included in the average.)



   The number of pixels included in the average is determined by the "radius", $r$, of the blur operation. The example above illustrates a radius of $r = 2$ pixels. The total number of pixels used to compute the average is $(2r + 1)^2$, which is 25 pixels in the example above.

   (a) Provide a pseudo-code description of an algorithm to apply the blurring effect to an image with $N \times N$ pixels. [8 marks]

   (b) Provide an ARM Assembly Language implementation of your blurring algorithm. (**Note:** You may assume the existence of a **divide** subroutine that divides an integer, $a$, in R1 by another integer, $b$, in R2 and returns the quotient in R0.) [17 marks]

6. Suppose you are required to develop an ARM Assembly Language program that implements a simple buzzer. Your program is to run on an LPC2468 Development Board, identical to those that you have used previously. The buzzer should sound for five seconds when a push-button is pressed. Your program must make appropriate use of TIMER device interrupts. Assume that the push-button is connected to pin P2.10 of the LPC2468 and the speaker is connected to the analog-out (AOUT) pin of the LPC2468's digital-to-analog converter.

   Your solution must include:

   (i) a description of your approach [6 marks]

   (ii) an explanation of how you would initialise the system and configure any timer devices or external interrupts required and [7 marks]

   (iii) 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. [12 marks]

   **Note:** When answering the above question, it is not necessary to provide detailed assembly language to control TIMERs, the configuration of the push-button pin or the DAC. Pseudo-code or a detailed written description will suffice.