UNIVERSITY OF DUBLIN
TRINITY COLLEGE

Faculty of Engineering, Mathematics and Science

School of Computer Science and Statistics

JF BA (Mod) Business and Computing
JF BA (Mod) Computer Science, Lingustics and a Language

Trinity Term 2010

CS1021 – Introduction to Computing I

Wednesday, 12 May 2010 **EXAM HALL** **14:00 – 16:00**

Dr Jonathan Dukes

You **MUST** Answer
Question 1 from **Section A** and
TWO out of **THREE Questions** from **Section B**

Non-programmable calculators are permitted for this examination

Please indicate the make and model of your calculator
on each answer book used

To be accompanied by an **ARM Instruction Set and Addressing Mode Summary** booklet

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

SECTION A (20 Marks)

You MUST answer Question 1 from this section

Suggested time allocation: 25-30 minutes

1. (a) How many binary digits (bits) are required to store the binary equivalent of the following decimal values?

(i) 320

(ii) 1045

(1 marks)

(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) +15

(ii) -100

(2 marks)

(c) Write an ARM Assembly Language program to compute the following mathematical expression, assuming x is in R1 and y is in R2. Your program should store the result in R0.

$$3x^2+9xy$$

(3 marks)

1. ... continued ...

(d) Consider each of the following independent instruction sequences. In each case, calculate the final value in R0 and state whether each of the **N** (negative), **Z** (zero), **C** (Carry) and **V** (oVerflow) flags would be set or clear after the execution of the final instruction in each sequence. Assume the flags are all clear before the execution of the first instruction in each sequence.

(i)
```
LDR     R1, =0x6E0074F2
LDR     R2, =0x211D6000
ADDS    R0, R1, R2
```

(3 marks)

(ii)
```
LDR     R1, =0xBF2FDC1E
LDR     R2, =0x40D032E2
ADDS    R0, R1, R2
```

(3 marks)

(e) Translate each of the following pseudo-code extracts into ARM Assembly Language. Assume all values are unsigned word-size values, a, b and c are stored in R0, R1 and R2 respectively.

(i)
```
if (a >= 60 && a < 70) // && is logical AND
{
     b = b + 1;
}
```

(2 marks)

(ii)
```
if ((a & 0xFFFFFFDF) == 'n') // & is bitwise AND
{
     b = 0;
}
```

(2 marks)

(iii)
```
if (a > 0)
{
     while (b >= a)
     {
          b = b - a;
          c = c + 1;
     }
}
```

(4 marks)

SECTION B (80 marks)

You MUST answer TWO out of THREE questions from this Section
All questions are worth 40 marks

Suggested time allocation: 40-45 minutes per question

2 (a) Show how you would perform each of the following operations using ARM Assembly Language:

(i) clear the least significant 5 bits of the word in R1

(1½ marks)

(ii) invert the most significant byte of the word in R2

(1½ marks)

(iii) change the sign of the word in R3, assuming a signed 2's complement number system

(1½ marks)

(iv) multiply the value in R4 by 16 using only the MOV instruction

(1½ marks)

(b) Design and write an ARM Assembly Language program that will generate an ASCII string representation, in hexadecimal form, of a word-size (binary) value stored in register R1. The string generated by your program must be prefixed with the characters '0x' and enough leading '0' characters so that the overall length of the string, excluding NULL termination, is ten characters. Store your generated string in memory starting at the address contained in R2.

For example, if register R1 contains the word-size value 0x00004F1E, your program should generate a string containing the following sequence of ASCII characters:

'0', 'x', '0', '0', '0', '0', '4', 'F', '1', 'E', 0 (NULL)

Your answer must include:

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

(10 marks)

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

(24 marks)

3. Write ARM Assembly Language programs to perform the string operations described below. Assume all strings are composed of byte-size ASCII characters and are NULL (zero) terminated. Provide adequate comments to explain your programs.

(a) Determine the number of ASCII characters in a string, excluding the NULL character. Assume that the start address of the string in memory is stored in R1. Store the computed length of the string in R0.

(8 marks)

(b) Convert each alphabetic character in a string to uppercase. Assume that the string may contain non-alphabetic characters.

(12 marks)

(c) Determine whether one string (with starting address in R1) is a substring of a second string (with starting address in R2). Your comparison should be case sensitive. Store the value 1 in R0 if the first string is a substring of the second and 0 if it is not.

(20 marks)

4. Design and write an ARM Assembly Language program that will determine whether each word-size value in a list of ten word-size values in memory is unique (i.e. each value occurs only once in the list). If every value in the list is unique, your program should store the value 1 in R0, otherwise you should store 0 in R0.

Your answer must include:

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

(ii) an ARM Assembly Language listing with adequate comments (30 marks)

For example, given the list below, your program should store a 1 in R0 because each value only occurs once.

5, 2, 7, 4, 13, 30, 18, 8, 9, 12

However, given the list below, your program should store a 0 in R0 because the value 4 occurs twice in the list.

5, 2, 7, 4, 13, 4, 18, 8, 9, 12

Assume that R1 contains the address of the first value in the list, that there are always ten values in the list, each value is one word in size and the values are stored contiguously in memory.

You may use the template program below for your solution.

```
          AREA     Unique, CODE, READONLY
          IMPORT   main
          EXPORT   start

start
          LDR      r1, =VALUES   ; list_addr = address of values

          <your program goes here>

stop      B        stop


          AREA     TestData, DATA, READWRITE

COUNT     EQU      10
VALUES    DCD      5, 2, 7, 4, 13, 4, 18, 8, 9, 12

          END
```