# TRINITY COLLEGE DUBLIN
# THE UNIVERSITY OF DUBLIN

## Faculty of Engineering, Mathematics and Science

## School of Computer Science & Statistics

**Integrated Computer Science**
**Year 1 Annual Examination**

**Trinity Term 2015**

## CS1022 – Introduction to Computing II

**Monday, 11 May 2015** **EXAM HALL** **09:30–11:30**

**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) Describe in detail the operation performed by each of the instructions below. For each instruction, **state the memory address that is accessed and the new value contained in any register that is modified**. Provide a diagram to illustrate the effect of any instruction that modifies the system stack. [4 marks]

(i) `STR R12, [SP, #-4]!`

(ii) `LDR R0, [SP, #32]`

(iii) `LDMFD SP!, {R4-R6, R12, LR}`

(iv) `STR R0, [R1, R2, LSL #4]`

Assume the following initial values in R1, R2 and SP (R13):

`R1=0xA1000080, R2=0x00000042, SP=0xA0000800`

(b) Translate the pseudo-code below into ARM Assembly Language. (A is an array of 32-bit signed values with start address labelled `A`. SIZE is the number of elements in array and is declared using an `EQU` directive.) [6 marks]

```
n = SIZE;
do {
    for (i = 1; i < n; i++) {
        if (A[i-1] > A[i]) {
            tmp = A[i-1];
            A[i-1] = A[i];
            A[i] = tmp;
            swapped = true;
        }
    }
    n = n - 1;
} while (swapped = false);
```

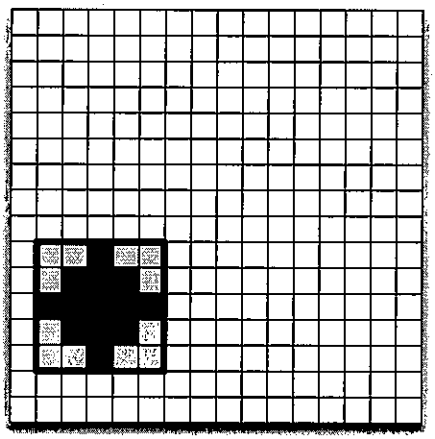
(c) Design and write an ARM Assembly Language **subroutine** that will reverse the order of the elements in an array. The reversed array must overwrite the original array in memory. Document a suitable interface for your subroutine. (Higher marks will be awarded if you can avoid using a stack or other memory to store a temporary copy of the array.) [7 marks]

(d) Design and write an ARM Assembly Language **subroutine** that will merge two arrays of unsigned 32-bit values **stored in ascending order** into a new single array **that is also sorted in ascending order**. Document a suitable interface for your subroutine. [8 marks]

2. (a) Explain the difference between the *row-major* and *column-major* approaches for storing a 2-D array in memory. Use a diagram to illustrate your answer. [2 marks]

   (b) Show how you would use ARM Assembly Language to load an element of a 2-D array of **8-byte** values into registers R0 and R1. Assume the start address of the array is stored in R2 and the row and column indices are stored in R3 and R4 respectively. The number of elements in each row is stored in R5. Load the least significant word into R0 and the most significant word into R1. [3 marks]

   (c) Consider an image stored in memory as a two-dimensional array of 32-bit values, with each value representing the colour of a single pixel. A logo is to be "stamped" on the image. The logo is a second image stored in memory in the same format. (The logo image will usually be smaller than the main image.) The figure below illustrates a $5 \times 5$ pixel diamond logo "stamped" on an original image.



   (i) Provide a pseudo-code description of a subroutine for "stamping" the logo on the original image. In other words, your subroutine should replace part of the original image with the logo image. Your subroutine should allow the logo to be placed at any location within the original image. State any assumptions that you make. [6 marks]

   (ii) Provide an ARM Assembly Language implementation of your subroutine from part (i). Document the interface for your subroutine. [14 marks]

3. A *game clock* is sometimes used in two-player games, such as Chess or Scrabble®, to limit the aggregate amount of time that each player can use to make all of their moves. Each player begins with a timer that counts down from an initial value towards zero while they take their turn in the game. When a player ends their turn, they press a button, causing their timer to be paused and their opponent's timer to resume counting down. If a player's timer reaches zero, they forfeit the game.

   Describe how you would develop an ARM Assembly Language program to implement the game clock described above using an LPC2468 development board identical to those that you have used previously. The remaining time for each player should be stored in memory. You must use TIMER interrupts to decrement the remaining time every second. Use the push-button attached to pin P2.10 to indicate a change of turn. If a player runs out of time, your program should generate a sound using the speaker connected to the analog-out (AOUT) pin of the LPC2468's digital-to-analog converter.

   Your answer must include:

   (i) a detailed description of your approach, using pseudo-code to support your explanation, [8 marks]

   (ii) a detailed explanation of how you would initialise the system and configure any timer devices or external interrupts required and [8 marks]

   (iii) an assembly language listing for any interrupt handlers required by your design. [9 marks]

   **Note:** When answering the above question, you may assume the existence of appropriate labels to refer to the addresses of memory-mapped device registers and the values used to perform actions such as stopping a TIMER device or enabling a VIC channel.