UNIVERSITY OF DUBLIN
TRINITY COLLEGE

Faculty of Engineering, Mathematics and Science

School of Computer Science & Statistics

Integrated Computer Science, Year 1 **Trinity Term 2014**

CS1022 – Introduction to Computing II

Thursday, 8 May 2014 **RDS, MAIN HALL** **14:00–16:00**

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. You must also provide a diagram to illustrate the effect of any instruction that modifies the system stack.

```
1       LDRB    R0, [R1, #4]!
2       STR     R0, [R1, R2, LSL #2]
3       LDMFD   SP!, {R4-R8, LR}
4       LDR     R0, [SP], #4
```

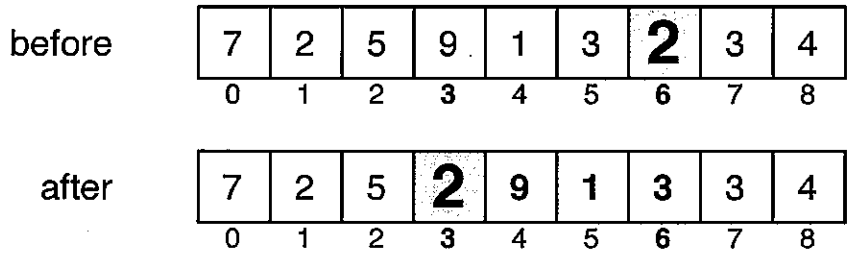
Assume the following initial values in R1, R2 and SP (R13):

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

[5 marks]

(b) Write an ARM Assembly Language subroutine that will move an array element from an old index to a new index in an array of word-size values. Your subroutine should accept the address of the array and the old and new index of the element as parameters.

The figure below illustrates an array in which an element is moved from old index 6 to new index 3. **Note how the elements between the old and new indices have been moved to fill the "gap" that was left in the array.**



Your answer must include:

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

(ii) an ARM Assembly Language listing for your subroutine. [7 marks]

question 1 continued on next page ...

... question 1 continued from previous page

(c) Explain how the ARM `BL` instruction allows you to invoke and subsequently return from a subroutine. What problem arises when you invoke one subroutine from another subroutine? How would you avoid this problem? [3 marks]

(d) Translate the pseudo-code shown below into an ARM Assembly Language subroutine. You must make use of your `move(...)` subroutine from part (b). Assume that A is an array of word-size unsigned values, N is the number of elements in array A and i and j are indices into array A.

```
sort(array, N)
{
    for (i = 1; i < N; i++)
    {
        j = 0;
        while (j < i && A[j] < A[i])
        {
            j = j + 1;
        }
        move(A, i, j);
    }
}
```

Your answer must include:

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

(ii) an ARM Assembly Language listing for your subroutine. [8 marks]

2. (a) Write an ARM Assembly Language **subroutine** to retrieve the value of an element from a two-dimensional array of word-size values stored in row-major order. Your subroutine must accept as parameters the start address of the array in memory, the row and column indices of the element to retrieve and the number of elements in each row. Your subroutine must implement the interface shown below. [2 marks]

```
1 ; get2D subroutine
2 ; Retrieve the value of an element from a 2D array
3 ; Parameters:   R0 - array, start address of the array in memory
4 ;               R1 - i, row number of element to retrieve
5 ;               R2 - j, column number of element to retrieve
6 ;               R3 - N, number of elements in each row
7 ; Return:       R0 - value of element array[i,j]
```

(b) Write a second ARM Assembly Language **subroutine** to set the value of an element in a two-dimensional array of word-size values stored in row-major order. Your subroutine must accept as parameters the start address of the array in memory, the row and column indices of the element to retrieve, the number of elements in each row and the new value to be stored in the array. Your subroutine must implement the interface shown below. [3 marks]

```
1 ; set2D subroutine
2 ; Set the value of an element in a 2D array
3 ; Parameters:   R0 - array, start address of the array in memory
4 ;               R1 - i, row number of element to retrieve
5 ;               R2 - j, column number of element to retrieve
6 ;               R3 - N, number of elements in each row
7 ;               [SP] - the new value to be stored in the array
8 ;                      (passed on the top of the stack)
```

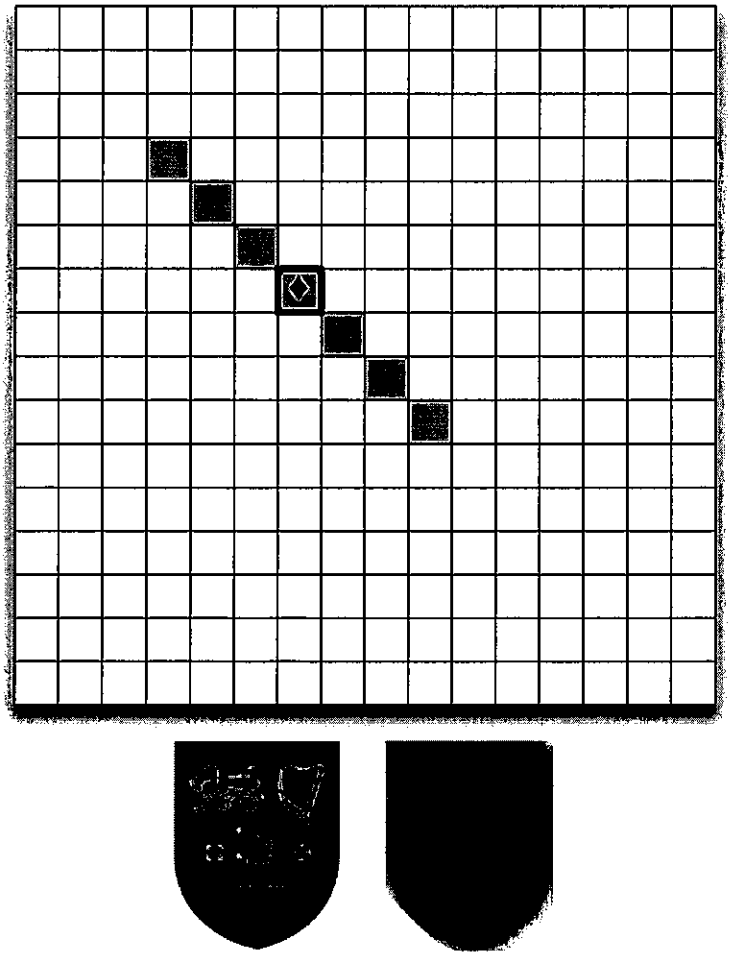
question 2 continued on next page ...

... *question 2 continued from previous page*

(c) Consider a **grey-scale** image stored in memory as a two-dimensional array of word-size unsigned integers. Each unsigned integer represents the brightness of the corresponding pixel in the image.

A "motion blur" effect can be applied to the image by replacing the brightness of each pixel with the average brightness of the pixels in a line through the current pixel. (Note that the original brightness of the current pixel is also included in the average.)

For example, in the figure below, the brightness of the pixel marked ♦ is replaced with the average brightness of the shaded pixels to produce a diagonal motion blur effect, also illustrated below.



question 2 continued on next page ...

... question 2 continued from previous page

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 $r = 3$ pixels.

(i) Provide a pseudo-code description of an algorithm to apply the motion blur effect, with a radius of r, to a grey-scale image, with $N \times N$ pixels, in the diagonal direction shown in the figure above. (**Note:** To simplify your answer, you may assume that the motion blur effect is only applied to the pixels at a distance of at least r from any edge.) [6 marks]

(ii) 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.) [14 marks]

3. (a) Consider a timer device used to measure the elapsed time in a rugby match. A single push-button is used to control the timer. The first time the button is pressed, the timer begins counting down from 40 minutes. If the button is pressed while the timer is counting down, the timer is paused. It resumes the next time the button is pressed. When the timer reaches zero, a buzzer should sound for five seconds.

Describe how you would develop an ARM Assembly Language program to implement the above timer using an LPC2468 development board identical to those that you have used previously. The remaining time should be stored in memory. You must use TIMER interrupts to decrement the remaining time every second. The push-button is connected to pin P2.10 of the LPC2468 and the speaker used to signal the end of the 40 minutes is 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, [9 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. [8 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).