UNIVERSITY OF DUBLIN
TRINITY COLLEGE

Faculty of Engineering, Mathematics & Science

School of Computer Science and Statistics

BA (Mod) Computer Science
Junior Freshman Examination

Trinity Term 2008

1BA3 – Introduction to Computing

Saturday 24 May 2008 **Regent House** **14:00 – 17:00**

Dr Jonathan Dukes

Attempt **FIVE** questions

All questions carry equal marks

To be accompanied by a **1BA3 Exam Handout** containing
MC68332 Instruction Set Summary and **Robot Technical Summary**

Non-programmable calculators are permitted for this examination

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

1. (a) Consider the execution of the sequence of instructions shown below. State the value of **d0**, **d1** and **each of the flags in the condition code register** (X, N, Z, V, C) after the execution of **each** instruction. (Assume X, N, Z, V and C are initially all clear.)

```
move.b   #8, d0
move.b   #-1, d1
add.b    d0, d1
sub.b    #9, d1
move.b   #126, d1
add.b    d0, d1
and.b    #127, d1
sub.b    #6, d1
```

(8 marks)

(b) Write a commented 68332 assembly language subroutine that will add two unsigned 128-bit values. You must use the stack to pass the two 128-bit parameters to the subroutine and to return the result.

(10 marks)

(c) Use an appropriate diagram to illustrate how the invocation and execution of your subroutine from part (b) would affect the system stack

(2 marks)

2. Design and write an assembly language program that will read a string entered by a user on the console, "echoing" each character as it is entered. When the user presses the "." (period) key, your program should display the string "PALINDROME" if the string entered by the user was a palindrome. Otherwise, your program should display the string "NOT PALINDROME". (A palindrome is a string that reads the same in either direction, e.g. "Able was I ere I saw Elba".) You should ignore case when determining whether the string is a palindrome.

You may only use the following trap routines to read from and write to the console:

trap #1	perform a blocking read of a single character from the console and store the read character in d0
trap #2	write the character in d0 to the console

Your solution must demonstrate problem decomposition by making appropriate use of subroutines and must include a commented assembly language listing.

(20 marks)

3. (a) For each of the pseudo-code extracts shown below:

(i) Translate the pseudo-code extract into 68332 assembly language (assume all variables are word-size)

(ii) For each conditional PC-modifying instruction that you use, explain why you chose that instruction, referring to the evaluation of the condition using one or more condition code register (CCR) flags (i.e. X, N, Z, V and/or C).

```
if (x > 5) {
    x = 5;
}
```

x is a signed value stored in d0

```
while (i <= 16) {
    i = i + 1;
}
```

i is an unsigned value stored in d1

```
if (j >= 100 && j < 200) {
    k = k + 1;
}
```

j is an unsigned value stored in d2, k is stored in d3

(10 marks)

(b) Translate the following pseudo-code into 68332 assembly language:

```
for (int i = 1; i < n; i++) {
    t = arr[i];
    j = i - 1;
    while (j >= 0 && arr[j] > t) {
        arr[j + 1] = arr[j];
        j = j - 1;
    }
    arr[j+1] = t;
}
```

Assume that the start address of the array, `arr`, is stored in address register `a0`, that `arr` is an array of word-size values and that `n` is the number of elements in the array and is stored in `d1`.

(10 marks)

4. (a) Show how you would perform each of the following bit manipulation operations using 68332 assembly language:

(i) clear all bits in the least significant byte of the word in `d0`

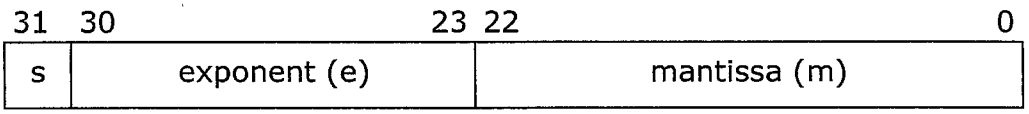
(ii) set bits 2, 3 and 4 of the word in `d1`

(iii) invert the most significant bit in the byte value in `d2`

(6 marks)

(b) Write a commented 68332 assembly language program that will convert the 32-bit floating point value stored in `d0` into an integer, storing the result in `d1`. For example, if the floating-point value stored in `d0` is 17.25, your program should store 17 in `d1`. Alternatively, if the floating-point value stored in `d0` is 0.625, your program should store 0 in `d1`.

Assume that a floating point value in the form $(-1)^s \times 1.m \times 2^{(e-127)}$ is stored as a 32-bit value as shown below, where s is the sign, e is the exponent and m is the mantissa (or significand). The exponent is stored with a bias of 127 and there is a "hidden" 1 to the left of the radix point.



You may ignore cases where the floating-point value cannot be represented as a signed 32-bit integer and you do not need to consider rounding.

(14 marks)

5. Design and write a 68332 assembly language subroutine to convert a signed word-size (16-bit) value to a null-terminated string representation in decimal form. The string generated by your subroutine should be stored in memory beginning at an address provided by the caller of the subroutine.

For example, given the value `123` and the address `$4200`, your subroutine should store the following sequence of ASCII characters in memory beginning at address `$4200`: `'1'`, `'2'`, `'3'`, `0` (NULL). Alternatively, given the value `-83` and the address `$6420`, your subroutine should store the following sequence of ASCII characters in memory beginning at address `$6420`: `'-'`, `'8'`, `'3'`, `0` (NULL).

Your solution must include the following:

(i) a description of the parameter-passing interface used by your subroutine

(ii) a commented assembly language listing for your subroutine

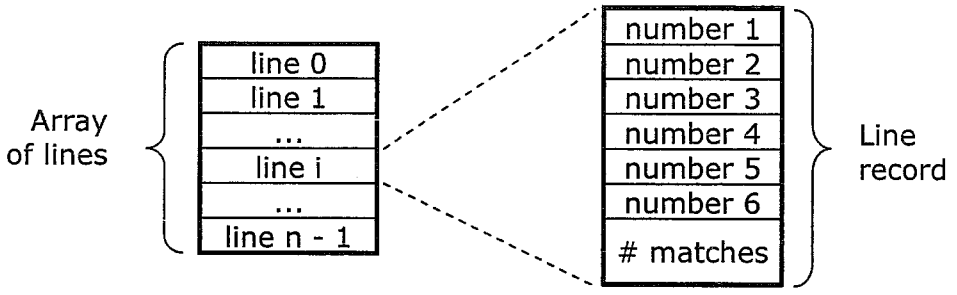
(20 marks)

6. Design and write a 68332 assembly language program which, given a list of "lines" entered into a lottery draw and the result of the draw, will:

(i) determine how many matching numbers there are in each line

(ii) count the number of lines with (a) 4 matching numbers, (b) 5 matching numbers and (c) 6 matching numbers.

Each line entered in the draw is represented as a record containing six single-byte values, followed by a word-size value to store the number of "matches" in the line. All of the lines entered in a draw are stored as an array of line records.



Your program should iterate over each line record in the array, compare the numbers in the line with the result of the draw and store the number of matches in the last word of the line record.

For example, if the numbers drawn in the lottery were 3, 7, 15, 19, 27, 29 and one of the line records contained the numbers 7, 14, 21, 22, 27, 30, then the number of matches for that line should be set to 2.

Your program should also store the total number of "match 4" lines at address $4200, the total number of "match 5" lines at address $4202 and the total number of "match 6" lines at address $4204.

Assume that the array of line records begins at the address stored in address register `a0` and that the result of the draw is stored as an array of six single byte values beginning at the address stored in `a1`. Assume also that data register `d0` contains the number of lines entered in the draw.

Your solution must include a commented assembly language listing.

(20 marks)

7. (a) Briefly explain, with the aid of an appropriate diagram, how a two-dimensional array can be stored in one-dimensional memory.

(2 marks)

(b) Given a two-dimensional array of word-size elements with *row_dim* rows and *col_dim* columns, calculate the byte offset used to access the element in the 3rd column and 4th row. State any assumptions that you make.

(2 marks)

(b) A colour digital image (e.g. a colour photograph) can be converted to a monochrome (or "grayscale") image in a simple manner by setting the colour of each pixel to a shade of gray with a "brightness" determined by the brightness of the original pixel colour.

The colour of a pixel can be stored as three values, representing the red, green and blue components of the colour. If the colour of a pixel with coordinates i and j is $(r_{i,j}, g_{i,j}, b_{i,j})$, then we can convert a colour image to monochrome by setting the colour of the pixel to $(y_{i,j}, y_{i,j}, y_{i,j})$, where $y_{i,j} = [(r_{i,j} + g_{i,j} + b_{i,j}) / 3]$.

Design and write a 68332 assembly language program that will convert a colour digital image to grayscale. Assume the image is stored in memory as a two-dimensional array of three-byte records. The three bytes in each record represent the red, green and blue colour components of a single pixel. Assume that address register `a0` contains the start address of the image, `d0` contains the width of the image in pixels and `d1` contains the height of the image in pixels.

Your solution must include a commented assembly language listing.

(16 marks)

8. (a) Briefly explain how memory-mapped I/O works and use examples from the 68332 computers you have been using to illustrate your answer.

(4 marks)

(b) Design an appropriate chip-select logic function to map a device with eight 32-bit registers into the memory space beginning at address $32000.

(4 marks)

(c) Write a 68332 assembly language subroutine that will display two NULL-terminated strings on the LCD display of the 68332 computers that you have been using. The first string should be displayed in the centre of the top line of the LCD display and the second string should be displayed in the centre of the bottom line of the LCD display.

Your solution must include the following:

(i) a description of the parameter-passing interface used by your subroutine

(ii) a commented assembly language listing for your subroutine

(12 marks)