PAPER CODE NO. COMP103	EXAMINER : **Dr Malcolm J Taylor** DEPARTMENT : **Computer Science** Tel. No. **43682/43649**

THE UNIVERSITY of LIVERPOOL

JANUARY 2001 EXAMINATIONS

Bachelor of Engineering : Year 1
Bachelor of Science : Foundation Year
Bachelor of Science : Year 1

COMPUTER SYSTEMS

TIME ALLOWED : Two Hours

INSTRUCTIONS TO CANDIDATES

Answer **ALL** questions in section **A** and **THREE** questions in section **B**.

If you attempt to answer more than the required number of questions (in any section), the marks awarded for the excess questions will be discarded (starting with your lowest mark).

THE UNIVERSITY of LIVERPOOL

SECTION A

Answer **ALL** Questions in this section.

A1. Outline the principal steps carried out by the computer processor in the execution of a single typical machine code instruction. Explain the role of the program counter and of other processor registers in this cycle.

(5 marks)

A2. Describe briefly how positive and negative integers are represented in **twos complement** binary notation. Illustrate your answer by showing the representation of the numbers +5 and –5 as 8-bit signed binary numbers.

(5 marks)

A3. In the context of computer hardware, give brief definitions of the following terms:

i) bus,
ii) half adder,
iii) flip flop.

(5 marks)

A4. Briefly explain the principle of a **stack** for storing information. Describe the features of the 68000 assembly language which provide for the implementation of the **stack**.

(5 marks)

A5. Write a sequence of institutions in the 68000 assembly language which will perform the equivalent of the Java conditional statement:

```
if ( p > = q )
    max = p ;
else
    max = q ;
```

(5 marks)

A6. What is the function of the **status register** in the 68000 processor? Explain how it is used in the construction of conditional sequences of instructions.

(5 marks)

A7. Give brief definitions of the following modes of data transmission:

i) simplex,
ii) half duplex,
iii) full duplex.

(5 marks)

A8. Distinguish between a memory address of a word in memory and the contents of a word in memory by explaining the function of the following 68000 assembly language instruction:

```
add.w memloc,do
```

(5 marks)

SECTION B

Answer **THREE** Questions from this section

B1. The piece of program below, in the 68000 assembly language, is written with the address of each instruction shown as a decimal number in the left-hand column.

(a) Draw up an execution history of the program, tabulating the changes in the values of the PC and the other registers used throughout an execution of the program.

(14 marks)

(b) For each of the instructions at addresses 2000, 2004 and 2016, give a clear explanation of the use and effect of the addressing modes used for each operation.

(6 marks)

```
2000                move #4, do
2004                lea array, a0
2008                move #0, d2
2012                move #0, d3
2016   loop:        move (a0)+, d1
2018                bgt positive
2022                add d1, d3
2024                jmp endloop
2028   positive:    add d1, d2
2030   endloop:     sub #1, d0
2034                bne loop

3000   array:       dc.w 26,-70,5,-9
```

B2.

(a) Write a 68000 assembly language subroutine called **sum** which takes as parameters two integers in **twos complement form** in the d0 and d1 registers, and evaluates the sum of the two integers which is to be returned in the d2 register.

As the two integers are in **twos complement form** then it is possible that on evaluation of the sum arithmetic overflow could take place. To accommodate for this, the subroutine should also return in the d3 register **either** integer 0 if the sum is correct **or** integer 1 if arithmetic overflow has taken place.

(NB: minor errors in the form of the instructions you write will not be penalised)

(10 marks)

(b) Write a UNIX shell script which will exchange the contents of two files, the names of which are passed as parameters. What action is required to make the shell script executable?

(10 marks)

B3

(a) By constructing an extended truth table for the expression, show that the logic expression:

f(A,B) = (A **and** B) **or** ((**not** A) **and** (**not** B))

will be true (or logic 1) **if and only if** A=B

(7 marks)

(b) Draw a logic diagram to represent this logic expression.

(5 marks)

(c) Two bytes are equivalent if all respective bits in the two bytes are equal. That is, bit 0 in the first byte is equal to bit 0 in the second byte, etc. For example, byte 11010011 is equivalent to byte 11010011 but not to byte 11010010 or 11010000, where one or more bits differ respectively.

Using as a building block (module) a component to represent the circuit described in (a) and (b), show how a number of such modules can be connected together with a simple logic gate to produce a logic 1 **if and only if** the two bytes are equivalent.

(8 marks)

B4.

(a) Describe the main steps carried out by a typical High-Level Language compiler.

(10 marks)

(b) Explain the terms even and odd parity.

What is the function of a parity bit?

Bytes of information in terms of ASCII characters with even parity are transmitted to a computer. Explain how this would enable ASCII characters with one bit in error to be detected but not two bits in error. In your explanation give an example to show how a character could be modified by both one and two bits in error respectively.

(10 marks)