UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 1996

BEng Honours Degree in Computing Part I
MEng Honours Degrees in Computing Part I
BSc Honours Degree in Mathematics and Computer Science Part I
MSci Honours Degree in Mathematics and Computer Science Part I
for Internal Students of the Imperial College of Science, Technology and Medicine

This paper is also taken for the relevant examinations for the
Associateship of the Royal College of Science
Associateship of the City and Guilds of London Institute

PAPER 1.4 / MC1.4

ARCHITECTURE I

Friday, May 10th 1996, 2.00 - 3.30

Answer THREE questions

For admin. only: paper contains
4 questions
4 pages (excluding cover page)

1 A Turing-like program has the following data and code:

```
type Person =
        record
        male  : boolean    % 8-bit, 0=FALSE, 1=TRUE
        age   : int        % 16-bit two's complement
        name  : array 1..4 of char  % ASCII chars
        score : real       % 32-bit IEEE floating-point
        end

var Fred : Person

begin
    Fred.male  := true
    Fred.age   := -21
    Fred.name  := "NOEL"   % NOEL is 4E4F454C in hex
    Fred.score := 64.75
end
```

For this program:

a Show the memory layout of the variable `Fred` at the end of the program if run on:

i) a 68000 architecture (BIG) with `Fred` at address \$1000.

ii) a modified 68000 architecture (LITTLE) with `Fred` at address \$1000. LITTLE is a 68000 architecture but one that employs a little-endian main-memory organisation.

Use 16-bit hex words when listing memory contents.

b Show the memory layout of the variable `Fred` on LITTLE if it is copied from `Fred` on BIG using:

i) a byte-by-byte transfer

ii) a word-by-word transfer

Use 16-bit hex words when listing memory contents.

c Comment on why the results in part b are incorrect, and outline a solution to correctly transfer the variable between BIG and LITTLE.

The three parts carry, respectively, 50%, 20% and 30% of the marks.

2a For the 68000 architecture define the following terms:

i) vector number

ii) interrupt vector table

iii) CPU exception

iv) trap

b Identify 4 differences between a procedure and an interrupt handler.

c Develop a simple stopwatch driver for an interrupt-driven timer device. The device has a byte-sized control port at address $FF80 and uses vector number 80.

Writing a 1 to the device's control port causes an interrupt to occur in 100 milliseconds time. Writing a 0 to the device's control port cancels any outstanding timer request.

For this device write the following routines in 68000 assembly code:

```
procedure InitWatch          % Initialises software

procedure TimerHandler       % Interrupt handler

procedure StartTiming        % Restarts timing from 0

function Seconds:longword    % Seconds since timing started

procedure StopTiming         % Stops timing
```

The three parts carry, respectively, 20%, 20% and 60% of the marks.

Turn over ...

3 You are required to develop a 68000 version of the following function:

```
function Position (ch : char,
                   var str : array 1..* of char,
                   len : int) : int
```

`Position` returns an integer indicating the position of the character `ch` in an array of characters `str`. The elements of `str` are indexed from 1 to `len`. If the character is not in the array or `len` is less than 1 then `Position` returns zero. Note that `str` is declared as a **var** parameter.

For this problem assume characters are 8-bit and integers are 16-bit. State any additional assumptions that you make.

a Provide commented 68000 assembly code for the assignment statement:

```
pos := Position ('G', Alphabet, 26)
```

where `pos` and `Alphabet` are global variables.

b Show the stack frame for the call `Position('G', Alphabet, 26)` where `Alphabet` is a global variable. Indicate clearly the offset (from SP or A6) and the size of each item on the stack frame.

c Provide commented 68000 assembly code for the `Position` function.

The three parts carry, respectively, 20%, 30% and 50% of the marks.

4a Suppose that the IEEE defines a new 8-bit floating point format called Tiny Precision that follows the same general rules as the IEEE Single Precision format except that the Exponent field is 3 bits and the Significand field is 4 bits.

	1 bit	3 bits	4 bits
Tiny Precision Format	Sign	Exponent	Significand

For this format calculate:

i) the appropriate excess for a 3-bit exponent field

ii) the largest normalised positive value that can be represented

iii) the smallest normalised positive value that can be represented

iv) the largest denormalised positive value that can be represented

v) the smallest denormalised positive value that can be represented

In each of the last four cases above, convert the positive values to decimal.

b Discuss what happens if you attempt to sum the series:

$$\sum_{n=1}^{N} \frac{1}{2^n}$$

using floating point arithmetic for some large N with

i) n increasing in value from 1 to N

ii) n decreasing in value from N down to 1

Explain which method is better.

c Explain why it is unsafe to compare two floating point values for equality. Show how two floating point values should be compared for equality.

The three parts carry, respectively, 30%, 35% and 35% of the marks.

End of paper