UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2002

BEng Honours Degree in Computing Part I
MEng Honours Degrees in Computing 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 City and Guilds of London Institute

PAPER C113

COMPUTER SYSTEMS

Friday 26 April 2002, 10:00
Duration: 120 minutes

Answer FOUR questions, at least one from each section

Paper contains 6 questions
Calculators required

Section A *(Use a separate answer book for this section)*

1a How does the ASCII code represent a character in a computer? What is the meaning and function of the parity bit? How many distinct characters can the core ASCII code represent?

Why was the UNICODE standard defined and how does it achieve its purpose?

b i/ Express the octal number 051515 as a 16 bit binary pattern.

ii/ What is the value of this pattern when interpreted as

1) a BCD number?

2) an ASCII character string?

Note the character "A" is represented by 65_{10} in ASCII

3) What parity are the characters in sub-part 2) above?

c Suppose the 10 least significant bits of the octal number in part b) i/ are taken. What is the value of this new bit pattern when interpreted as:

i/ a sign and magnitude number?

ii/ a one's complement number?

iii/ an excess - n number with the usual excess for the field length?

d Discuss the advantages and disadvantages of the three mappings in part c) above.

Each part carries equal marks.

2a Give the steps of the "fetch-execute" cycle of a computer executing an ADD instruction between a register and memory. Describe with an example the main activities which take place in each step.

b For a modern computer architecture identify the different types of hardware registers. Give examples of each type and describe their function. Carefully distinguish between those which are programmer accessible and those which are not.

c Suppose that an architecture has 8 general purpose registers and a maximum addressable range of 8 Megabytes. Main memory is byte-addressable and the architecture has 60 instructions each of the form:

OPCODE REGISTER ADDRESS

For this architecture determine: showing your working:

i) the minimum width, in bits, of the instruction register.

ii) the minimum width, in bits, of the program counter register.

iii) the maximum number of instructions in any program.

d Show how a zero-address instruction design might work and discuss the advantages and disadvantages of such an architecture.

Each part carries, equal marks.

Section B *(Use a separate answer book for this Section)*

3a State one advantage and one disadvantage of writing an operating system in a high-level language, such as C or C++.

b Using pseudocode, describe the functionality of the two main components of a *PRINTER* device driver.

c In virtually all systems that include a DMA (direct memory access module) its access to memory runs at a *higher priority* than the processor. In one paragraph, briefly explain operation of the DMA. What does '*higher priority*' mean in this context? Why should this be the policy setting for the DMA?

The three parts carry, respectively, 15%, 55%, 30% of the marks.

4a Give an example which shows why testing and setting a lock must be an indivisible operation.

b Two concurrent processes P1 and P2 each have a critical region. P1 *increments* the shared integer X by one while in its critical region P2 *decrements* X by 1. Write pseudocode to protect these critical regions using Semaphores. Include data declarations and appropriate initialisations.

c i Illustrate with an example what is meant by *deadlock.*

ii Describe what facilities the operating system provides to detect deadlock

iii What would be a good policy for choosing which process to terminate once deadlock has been found? If a system did not have deadlock detection facilities how would deadlock manifest itself to the user?

The three parts carry, respectively, 20%, 45%, 35% of the marks.

Section C (Use a separate answer book for this Section)

5a Translate the following high-level language class into a **commented** Pentium assembly language version:

```
class comparator {

    int sign [20];    // Assume int's are 32-bit

    int compare (int x, int y) {
        if    (x < y)      return -1
        else if (x == y)   return  0
        else               return +1;
    }

    void arraycompare (int [20] a, int [20] b) {

        int k;
        for (k=0; k<20; k++) {
            sign [k] = compare (a[k], b[k]);
        }
    }

}
```

b Show the contents of the stack just before the **if** statement in method `compare` is executed. You should assume that `compare` is called from `arraycompare` and your stack should show the stackframes for both `arraycompare` and `compare`.

The two parts carry, respectively, 80%, 20% of the marks.

6a Suppose that the IEEE defines a new 9-bit floating point format called Tiny Precision that follows the same general rules as IEEE Single Precision format except that the Exponent is 4 bits and the Significand is 4-bits

	1 bit	4 bits	4 bits
Tiny Precision Format	Sign S	Exponent E	Significand F

For this format determine:

i) the next number after 200 that can be represented exactly.

ii) the largest positive normalised number that can be represented.

iii) the smallest positive normalised number that can be represented.

iv) the largest positive de-normalised number that can be represented.

v) the smallest positive de-normalised number that can be represented.

For parts (i) to (v) above, give your answer in both tiny precision format, and as a binary value in the form **1.bbbb x 2^N**. For parts (i) and (ii) only, give your answer in decimal also.

b Multiply the tiny precision number `0 0101 1000` with itself (i.e. square the number). Give the result both in tiny precision format and hexadecimal.

c Divide the tiny precision number `1 0110 1110` by the tiny precision number `0 0101 0100`. Give the result both in tiny precision format and hexadecimal.

d On some computers floating point multiplication is performed more quickly than floating point addition. Since multiplication by hand is more complex than addition, this is surprising. Why might it be faster?

The four parts carry, respectively, 25%, 25%, 25%, 25% of the marks.