IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2011

MSc in Computing Science
for Internal Students of the Imperial College of Science, Technology and Medicine

PAPER M2

COMPUTER SYSTEMS

Tuesday 10 May 2011, 14:30
Duration: 120 minutes

Answer THREE questions

Paper contains 4 questions
Calculators required

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

1 This question concerns logic circuits, memory organisation and number representation

a i) Provide truth tables for A OR B and A NAND B

ii) State De Morgan's laws.

iii) Explain how an OR gate can be built out of NAND Gates. Sketch the circuit, showing how the inputs are connected via the NAND gates to the output.

b 16 Megabytes of byte-addressable memory is built using 256K x 8-bit RAM chips aligned on 32-bit words.

i) How many RAM chips are rquired in total?

ii) How many memory modules will there be?

iii) If the memory is organised with low-order interleaving, which module will contain the 32-bit word at memory address 003CB4?

c The memory word described in part b(iii) contains a floating point number.

i) Describe the components of the IEEE single precision floating point representation.

ii) The hexadecimal value in the location is C49A5000. Convert this to a decimal floating point representation.

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

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

2a i) Name three possible types of operands in an instruction according to their addressing modes.

ii) Explain briefly the terms programmed I/O and interrupt-driven I/O. Name one advantage of interrupt-driven I/O over programmed I/O.

iii) When an interrupt occurs and before it is handled on the Pentium CPU, which two registers' contents will be saved and where so that execution of the original program can be resumed after handling the interrupt. Why are the saved contents sufficient to resume the original program execution?

b Consider a simple CPU with four general purpose registers (R0, R1, R2 and R3) and the following instructions:

Opcode	**Instruction**	**Action**
0000	STOP	Stop Program Execution
0001	LOAD Rn, [Addr]	Rn = Memory [Addr]
0010	STORE Rn, [Addr]	Memory [Addr] = Rn
0011	ADD Rn, [Addr]	Rn = Rn + Memory [Addr]
0100	SUB Rn, [Addr]	Rn = Rn - Memory [Addr]
0101	GOTO Addr	PC = Addr
0110	IFZER Rn, Addr	IF Rn == 0 THEN PC = Addr
0111	IFNEG Rn, Addr	IF Rn < 0 THEN PC = Addr
1000	not used	
1001	LOAD Rn, [Rm]	Rn = Memory [Rm]
1010	STORE Rn, [Rm]	Memory [Rm] = Rn
1011	ADD Rn, [Rm]	Rn = Rn + Memory [Rm]
1100	SUB Rn, [Rm]	Rn = Rn - Memory [Rm]

Let a vector of n variables $A[0], A[1], \ldots, A[n\text{-}1]$ be stored at consecutive memory locations. Each memory location can store one single variable or instruction.

Write a pseudo code program to sort the variables in ascending order and store them in the same consecutive memory locations. That is, the smallest and largest variables are placed in memory locations with the smallest and largest addresses, respectively. Using the above instructions, write the corresponding assembly program to perform the sorting.

Specify the memory locations for variables $A[0], A[1], \ldots, A[n\text{-}1]$ and n in hexadecimal numbers. Define necessary constants and temporary variables, and provide their memory locations. Also specify the memory locations of your assembly instructions.

The two parts carry, respectively, 45% and 55% of the marks.

3 a Discuss the (dis)advantages of *Compile Time*, *Load Time* and *Execution Time* address binding.

b Discuss the *Buddy System*.

c Show, in a diagram, the use of 128Mb of memory, managed via the buddy system, that receives the following instructions:

i) A: Request 9Mb,

ii) B: Request 6Mb,

iii) C: Request 13Mb,

iv) Release A,

v) D: Request 5Mb,

vi) Release B,

vii) Release D,

viii) Release C.

d Discuss *paging*; what does it solve, and how? What is a *translation look-aside buffer*?

The four parts carry equal marks.

4 This question has 4 parts and is concerned with I/O and inter-process communication.

a What is DMA and why is it useful?

b Consider the psuedo-code for an interrupt-handler shown below (`PC` is the program counter and `Mem` is the main memory).

```
void InterruptHandler ()
{
  saveProcessorState();
  dealWithInterrupt();
  restoreProcessorState();
  InterruptEnabled = true;
  PC = Mem[0];
}
```

Is this always going to work? If not, why and how can you fix it?

c Briefly describe the functions of the bottom-half and the top-half of a device driver.

d Consider a barber shop with one *barber* and many *customers*. The barber and customers can each be thought of as a process. The barber has one barber chair and a waiting room with a number of chairs in it. The following rules apply:

- The barber can only cut one customer's hair at any time.
- When the barber has finished with a customer, he/she dismisses that customer and and checks for any waiting customers. If no customers are available, the barber goes to sleep.
- A customer on arrival checks on the barber. If the barber is sleeping, the customer wakes up the barber and sits in the barber's chair.
- If the barber is busy, the customer goes to wait in the waiting room.
- If there are not chairs available in the waiting room, the customer leaves.

Write pseudo-code that implements the above protocol. Please make sure that you state your assumptions clearly.

The four parts carry, respectively, 20%, 20%, 25%, and 35% of the marks.