IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2012

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

PAPER M2

COMPUTER SYSTEMS

Tuesday 8 May 2012, 10:00
Duration: 120 minutes

Answer THREE questions

Paper contains 4 questions
Calculators required

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

1 a A machine has 2M×16-bit memory made up from 256K×4-bit chips

i) How many chips are there per memory module? How many modules are required?

ii) How many address bits are required to access this memory if it is byte addressable? How many address bits are required if it is word addressable?

iii) The memory is byte addressable and the memory at address 100_H contains the following values

100_H	101_H	101_H	101_H
03_H	$F7_H$	80_H	01_H

If the architecture is Big Endian and the memory contains two two's complement integers, what are the decimal values of these two integers? What is their sum? Perform your calculation in two's complement.

iv) What is the answer for part iii if the memory is Little Endian? What is the sum of these two numbers?

b Suppose that the IEEE defined a new 16-bit floating point format called *Tiny Precision* that is the same as IEEE Single Precision format but with the following assignment of bits:

Sign S	Exponent E	Significand F
1-bit	7 bits	8 bits

i) In what format should the Exponent be held? Interpret the 16-bit hexadecimal value 4180_H as a Tiny Precision number and then convert this Tiny Precision number to decimal.

ii) What is the Tiny Precision representation of the decimal number -17.5? Give your answer as binary bits and hexadeximal digits.

iii) Caclulate the sum and the product of the two numbers from parts i and ii together. Do all your working in binary - stating clearly how you handle each of the different parts of the format. Give the result in Tiny Precision representation (binary and hexadecimal) and finally check the result by conversion to decimal.

The two parts carry equal marks.

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

2a i) What are the two fields of the instruction that fully specify an "unconditional jump" instruction? Describe in a few steps how the jump instruction is executed in a CPU.

ii) Name three types of operands associated with an instruction according to the places where the operands are stored. Identify the operand type that supports the maximum range of data representation and quick access time, and explain why.

iii) What is the access order for stack memory? Explain how stack memory is used to support procedure or function calls.

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

Opcode	Assembly 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

Using these instructions, write an assembly program to compute the product of two numbers *most efficiently*:

$$A = B * C$$

where B and C are known to be positive integers.

To start, write a pseudo code to achieve the computation and then convert it into the assembly instructions.

Define necessary constants and provide their memory locations. Also specify the memory locations of your variables and assembly instructions.

The two parts carry 50% and 50% of the marks, respectively.

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

3 a Give the design (pseudo code) for both the *synchronous send* and the *receive* procedure as used in a message passing mechanism.

b Discuss the differences between *synchronous* and *asynchronous* send, and their implications for the kernel. The answer should not be longer than a few sentences (about a 100 words).

c Is the code in the OS that does the actual scheduling part of a task/process or not? Explain your answer.

d Why could it be useful to introduce the distinction between user processes and system tasks (give at least three reasons).

e **New.** Consider the situation where we want to create a link `/homes/guest/` to the existing file `/tmp/files/`. In this case, `/homes/guest/` is the link name and `/tmp/files/` is the file to which it is linked.

i) Describe the steps required to read the first byte of `/tmp/files/dummy` starting with the soft link `/homes/guest/dummy`. How many disk reads will it require?

ii) If the file `/tmp/files` is removed, is the hard link still valid? Is the soft link still valid?

The five parts carry, respectively, 30%, 20%, 10%, 20%, and 20% of the marks.

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

4a The DMA is considered the "processor's helper". Briefly describe what the DMA is and how it operates. In your description, mention the three main arguments (*parameters*) the software will pass to the DMA to carry out its task.

b Briefly explain the terms ***synchronous*** and ***asynchronous*** in terms of an operating system. Explain why **asynchronous** systems require buffering?

c A house of improvised students eats communal dinners from a large pot that can hold *M* servings of soup. When a student wants to eat, he/she helps themselves from the pot, unless it is empty. If the pot is empty, the student wakes up the student-cook and then waits until the cook has refilled the pot.

Listings 1 and 2 show *pseudocode* representing the student and cook processes.

```
while True:
   putServingsInPot(M)
```

Listing 1: Unsynchronized cook code

```
while True:
  getServingFromPot()
eat()
```

Listing 2: Unsynchronized student code

The synchronization constraints are:

- Students cannot invoke **`getServingFromPot`** if the pot is empty.
- The cook can invoke **`putServingsInPot`** only if the pot is empty.

Using only semaphores, write *pseudocode* for the students and the cook that satisfies the synchronization constraints, clearly stating all assumptions made.

d In many current architectures, the operating system controls DMA operation via *parameters* rather then using direct addresses. There are two advantages of this; Atomicity and Protection (from programming errors and malicious users).

i) What do we mean by Atomicity in terms of the main parameters you give in section 4a?

ii) What do we mean by Protection in terms of the main parameters you give in section 4a?

The four parts carry, respectively, 20%, 20% 40%, and 20% of the marks.