IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2013

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

PAPER M2

COMPUTER SYSTEMS

Tuesday 7 May 2013, 10:00
Duration: 120 minutes

Answer THREE questions

Paper contains 4 questions
Calculators required

1 This question has 4 parts.

a Briefly describe the *Big Endian* and *Little Endian* formats for storing data in memory. Please show how the string `TESTING` is stored in memory in both of the above formats.

b Assume that you have N-bits in memory to represent a number. Briefly describe the 2's complement number representation for N-bits. Calculate the 2's complement value of the decimal number **-5** when you have **4-bits**.

c Express the following decimal number **-139.325** in binary and hexadecimal using the IEEE Single Precision Format. Show your working clearly. If you use extra bits in your calculation, use rounding rather than truncation when calculating the result.

d Simplify the following Binary expression to its simplest form:
$E = A' \bullet (A + B) + (B + A \bullet A) \bullet (A + B')$, where $\bullet$, $+$, and $'$ represent "AND", "OR" and "NOT" operations respectively. Please show your working clearly and state the reduction rules used.

The four parts carry, respectively, 15%, 20%, 30%, and 35% of the marks.

2a i) Name a special register within a CPU that contains the address of the next instruction to be executed. Explain how the register is updated normally (i.e., to execute the next instruction in sequence). Name one type of instructions which can change the content of that register explicitly, other than by the normal way.

ii) Explain what the "Call" and "Return" instructions do for initiating and ending the execution of a procedure (function) call, respectively. In your explanation, name the register and memory segment that are typically involved in executing the instructions.

iii) Following part ii) above, what should an assembly programmer do to ensure that after the execution of the called procedure, the program execution resumes with the CPU state identical to that immediately before the procedure call?

iv) What are two main differences between a procedure (function) call and a hardware interrupt?

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

By using the above instructions, write an assembly program to compute

$$B = A[0] - A[1] + A[2] - A[3] + ... + A[n-2] - A[n-1]$$

where n is a positive, even integer and the variables $A[0]$, $A[1]$, …, $A[n\text{-}1]$ are stored in consecutive memory locations. We can assume that the computation does not cause any overflow or underflow.

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

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

3 a Explain why, when running a multi-user system, interrupts are essential when dealing with IO.

b Describe the classic instruction cycle, and what happens when an interrupt occurs.

c Explain the *vectored interrupt* scheme.

d Give the *process state diagram*. Give an extension that deals with the notion of a process being *swapped out*, i.e., who's process descriptor exists, but no longer has its code in memory.

e Describe the flow-of-control in MINIX of a system call like `READ` that reads a block from disk. Make sure that you discuss the role of the interrupt handler, the system task, the user process and the scheduler.

The five parts carry, respectively, 15%, 25%, 15%, 25%, and 20% of the marks.

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

b Explain why the kernel of a system implementing *asynchronous message passing* needs to provide *buffering*, whereas this is not needed when *synchronous* message passing is used.

c In the context of processes that run concurrently, when protecting access to critical regions, the following instruction can be used: `tsl reg, lock`. What are '`ref`', '`tsl`' and '`lock`' here?

Give the implementation of protection of critical regions, i.e. give assembler specifications of both '`enter_cr`' and '`leave_cr`', using `lock`s.

d Describe in a few lines the effect of the Unix system calls `fork` and `exec` and how they interact.

The four parts carry, respectively, 30%, 25%, 25%, and 20% of the marks.