IMPERIAL COLLEGE LONDON EE1-09

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
EXAMINATIONS 2017

EIE PART I: MEng, BEng and ACGI

Introduction to Computer Architecture and Systems

Monday, 22nd May 2017, 10:00AM

Time allowed: 1:00 hour

Corrected copy

There is ONE Question on this paper

Answer ALL questions.

Any special instructions for invigilators and information for candidates are on page 1.

Examiners responsible	First marker:	T.J.W. Clarke
	Second marker:	C. Bouganis

The Questions

The question is on the next two pages

1. a) Define a *context switch* to be whenever an unfinished job is suspended. Note that when one job finishes and another starts or restarts, it is not a context switch.

   i) In a system with n jobs, assuming no blocking, state what is the maximum number of context switches possible, or whether there is no upper bound, for each of the following scheduling methods: First-come-first-served scheduling, priority scheduling with pre-emption, round-robin scheduling.

   [3]

   ii) In an operating system scheduler design, the aim is to provide a waiting time of no more than 10ms to n interactive jobs from users. Where this 10ms limit cannot be met, the aim is to reduce the worst-case waiting time for any job. Describe, with reasons, the scheduling method you would use for interactive jobs, if context-switch time is 0.

   [2]

   iii) Suppose in the scheduler design described in (ii), it is known that no interactive job lasts for more than 5ms, and the context-switch time is also 5ms. You may assume that each interactive job overlaps with at most one interactive job from each of the $n-1$ other users. State, with reasons, what scheduing method would then be optimal.

   [2]

b) i) In an operating system the following *atomic* OS API calls are provided to user processes:

- *Load-and-add.* Return the value of a memory location, then add a fixed signed integer to the location.
- *Remove-from-queue-and-make-ready.* Remove the front process from a queue and make its state *ready-to-run.*
- *Add-to-queue-and-suspend.* Add the currently running process to the front of a queue and suspend it, making its state *waiting*.

Using these OS calls, define a semaphore data structure and write pseudocode for three user functions that correctly implement the semaphore *init*, *wait*, and *signal* operations.

[3]

ii) What change, if any, would be needed to make your implementation a *strong semaphore* where waiting processes are allowed to access the critical region in first-come-first-served order.

[2]

iii) Four processes share a printer resource which requires a mutually exclusive critical region of code to implement spooling. Explain how to use semaphore operations as in (i) to ensure that critical regions in all four processes are mutually exclusive. Detail what in your code would change if there were two available printers and therefore at most two processes could enter their critical regions simultaneously.

[2]

c) Describe page table and base-and-limit-register memory allocation methods. Define interior and exterior fragmentation, and state how these apply to each memory allocation method.

[2]

d) Contrast the merits of using threads or processes to implement concurrent processing needed by a single user.

[2]

e) State what factors determine the performance of a virtual memory system, and under what circumstances thrashing might occur.

[2]