Student Number:	
Family Name:	
Given Names:	
Signature:	

THE UNIVERSITY OF NEW SOUTH WALES

Final Examination

Sample

COMP3231/COMP9201/COMP3891/COMP9283
Operating Systems

- Time allowed: **2 hours**
- Reading time: **10 minutes**
- Total number of questions: **5**; Total number of pages: **7**
- Answer **all** questions. The questions are **not** of equal value
- Total marks available for the exam: **100**
- This paper may **not** be retained by the candidate
- **Answers must be written in ink**, with the exception of graphs and multiple-choice answers sheets.
- Electronic calculators approved by the University may be used.
- No other examination materials may be used.
- **Provide answers to Question 1 on the answer sheet provided.**
- **Use a *separate* answer book for *each* of the other questions.**
- **2 marks will be awarded if the above instructions are followed.**

Question 1 [40 Marks]

Answer this question *on the multiple-choice answer sheet* provided.

For each statement given below, classify each statement as *true* if the statement is generally true, or *false* if the statement (or a component of it) is false.

If the statement is *true*, mark box 'A' on the answer sheet provided; if the statement is *false*, mark box 'B'. You will receive one mark for each correct classification, and lose one mark for each incorrect classification. You gain zero marks for each answer left unclassified. The overall mark for this question will not be negative, i.e. the minimum mark is zero.

1) The role of an operating systems is to present all the low level details of the computer hardware without any abstraction.
2) The operating system runs in the privileged mode of the microprocessor.
3) An application running in user-mode shares the same stack with the operating system when it runs in kernel mode.
4) An operating system can enforce security by putting checks in the standard C-library.
5) Arguments to system calls are placed in registers (or on the stack) based on a mutually defined convention between the OS and user-level applications.
6) In the three-state process model, and common transition is from *ready* to *blocked*.
7) Co-operative (non-preemptive) multitasking can result in a non-responsive system if an application has an endless loop.
8) A threading library implementation at user-level (i.e. user-level threads) does not expose the concurrency available in the application to the operating system.
9) Application threading supported by kernel implemented threads (i.e. kernel-level threads) can take advantage of multiple processors if available.
10) Applications (i.e. user-level code) can synchronise to avoid race conditions by disabling and enabling interrupts.
11) Semaphores can be used to implement mutual exclusion primitives.
12) Condition variables are used together with semaphores to manage blocking and waking within the semaphore.
13) *Bankers algorithm* can avoid deadlock if the maximum resource requirements of threads are known in advance.
14) *Hold and wait* is a practical deadlock prevention strategy.
15) Sparse files save disk space by not storing the parts of the file that are not written to.
16) The buffer cache improves write performance by buffering writes. Without extra application effort, the performance increase comes at the expense of reliability and consistency in the presence of failures.
17) Contiguous file allocation is an allocation strategy suitable for read-only media.
18) Chained (link-list) file allocation is desirable for file systems that support random-access workloads.
19) Best-fit memory allocation gives the best result in terms of memory fragmentation.
20) Swapping allows applications larger than physical memory to execute.
21) Overlays allow applications larger than physical memory to execute.
22) With appropriate language and OS support, segmentation can be used for bounds checking array access.
23) Virtual memory thrashing is where virtual memory translation happens too fast.

24) *Optimal* is the best practical page replacement algorithm.

25) When choosing a victim for page replacement, a clean page is faster to replace than a dirty page.

26) Increasing the page size generally increases the working set size of an application.

27) Spatial locality contributes to VM system efficiency, but temporal locality does not.

28) Adding more levels of I/O buffering always improves performance.

29) *Shortest seek time first* is a disk scheduling algorithm that is always fair.

30) Compared to polled I/O, Interrupt driven I/O is preferable when there is a significant delay between a device request and the device response.

31) Favouring CPU-bound applications over I/O-bound applications generally improves overall system performance.

32) Rate-monotonic scheduling can always schedule a task set if the total utilisation is less or equals to 1.

33) Round robin scheduling can be tuned by varying the time-slice length to trade-off responsiveness for decreased CPU overhead.

34) Spinlocks can never be more efficient to use than blocking locks, even on a multiprocessor machine.

35) On a multiprocessor machine, a single ready queue provides automatic load balancing between CPUs.

36) A *read before test and set* implementation of a spinlock just adds extra overhead to the lock implementation by adding a superfluous read.

37) A multiprocessor machine only speeds up parallelisable workloads.

38) On a processor with a hardware-refilled TLB, the operating system is free to implement the most efficient page table data structure for the expected workload.

39) Memory compaction can be performed transparently to an application on a machine with *base* and *limit* registers.

40) Page sizes are always a power of 2.

Question 2 [18 Marks]

Answer this question in a *separate* book

A) **[6 marks]**

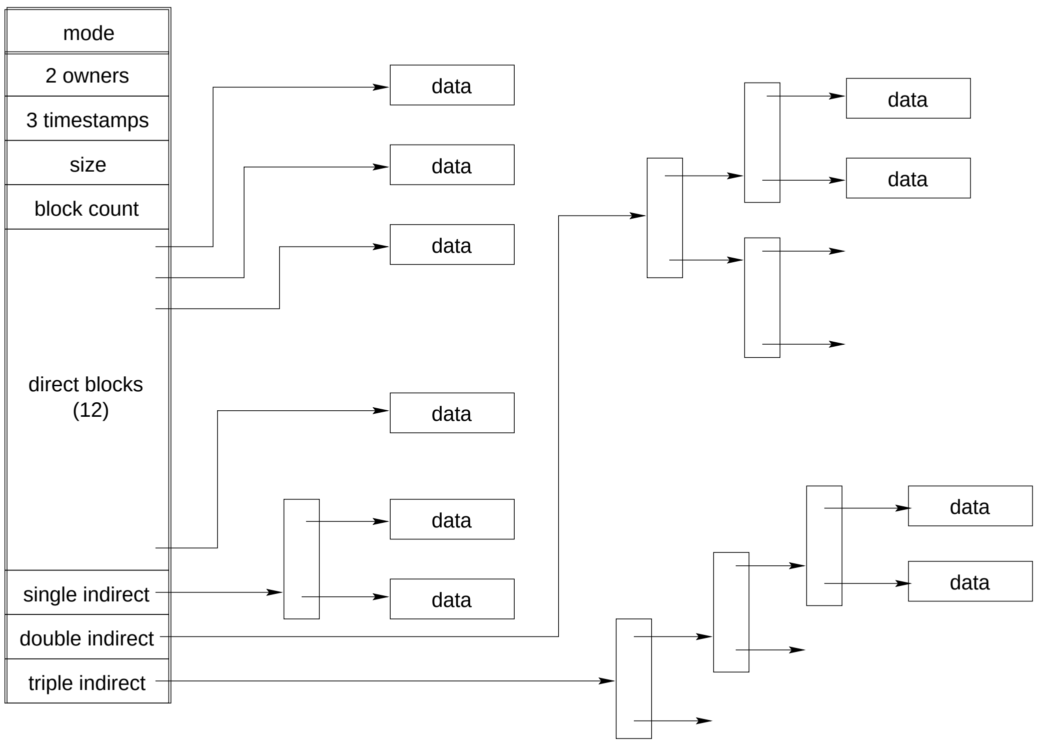
Consider a demand-paging system with a paging disk that has an average access and transfer time of 5 milliseconds for a single page. Addresses are translated through a page table in main memory, with an access time of 100 nanoseconds per memory access. Thus, each memory reference through the page table takes two accesses. The system has a 48-entry TLB to speed up memory accesses. Assume that 99% of memory accesses result in a TLB hit, and of the remaining 1%, 5 percent (or 0.05% of the total) cause page faults. What is the effective memory access time?

B) **[6 marks]**

Some versions of UNIX store the first part of each file in the same disk block as the inode. Discuss why this might be advantageous in practice.

C) **[6 marks]**

In most UNIX file systems, every file has an index block, called an *inode*, which is used to store information about the file as well as pointers to data blocks, as shown in the diagram below.



Assume a block size of 2 kilobytes (2048 bytes) and 4-byte block numbers. What is the largest possible file size support by this filesystem?

Question 3 [14 Marks]

Answer this question in a *separate* book

A) **[8 marks]**

Suppose that the head of a moving-head disk with 192 tracks, numbered 0 to 191, is currently serving a request at track 80 and has just finished a request at track 62. The queue of requests is kept in the FIFO order: 119, 58, 114, 28, 111, 55, 103, 30, 75. What is the total number of tracks traversed by head movements needed to satisfy these requests for the following disk-scheduling algorithms?

i) FCFS.

ii) SSTF.

iii) Elevator (SCAN).

iv) Modified Elevator (C-SCAN).

B) **[6 marks]**

User-level threads packages generally implement cooperative scheduling. What is cooperative scheduling, and why is it the common method used to schedule user-level threads?

Question 4 [18 Marks]

Answer this question in a *separate* book

A) **[8 marks]**

Explain how a 32-bit virtual address is translated into a physical address on a system using a two-level page table and 4kb pages. Your explanation, where it refers to parts of an address, **must** specifically state which bits of the address you are talking about. (Ignore any TLB). **State any assumptions you make!**

B) **[4 marks]**

Describe the difference between external and internal fragmentation. Indicate which of the two are most likely to be an issues on a) a simple memory memory mangement machine using base limit registers and static partitioning, and b) a similar machine using dynamic partitioning.

C) **[6 marks]**

Describe in detail why a Mellor-Crummey Scott (MCS) lock would be preferable to a simple test-and-set spinlock on a multiprocessor machine.

Question 5 [14 Marks]

Answer this question in a *separate* book

A) **[8 marks]**
Describe the four conditions required for deadlock to occur. Describe a common method for deadlock prevention that prevents one of the conditions occurring.

B) **[6 marks]**
What are the two main roles of an *Operating System*? For each of the two roles, give an example of a service that demonstrates the operating system fulfilling its role, and how the fulfillment of the role is beneficial.

End of Paper