COMP25111 MARKING SCHEME

Two hours

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Operating Systems

Date:

Time:

Please answer Questions A1 and B1
and
TWO other Questions from A2, B2 or B3

Use a SEPARATE answer book for each SECTION

This is a CLOSED book examination

The use of electronic calculators is NOT permitted

[PTO]
Section A

A1. **Compulsory**

a) A web browser in the UK accesses a database in New Zealand. How long does it take for the data to arrive back in the UK after the user presses the ‘access’ key on the web page? For simplicity, assume that: (1) the times to process the browser key, to retrieve the data from the database, and to send and receive messages are all negligible; and (2) the round trip data transfer takes the same time as light to travel round the circumference of the Earth (a distance of 40,000 kilometres at a speed of $3 \times 10^8$ metres/second). Assume the computer running the browser has a clock rate of 3GHz and can execute one instruction per clock cycle. How many instructions could that computer execute while waiting for the response from the database?

(2 marks)

Round trip for data transfer takes $\frac{4\times10^7}{3\times10^8} = 0.1333$ s = 133.3 ms. 3GHz clock does $3 \times 10^6$ cycles per ms, so number of clock cycles while waiting is $3\times133.3\times10^6 = 400$ million = number of instructions that could be executed. Last year, a similar question caused candidates a lot of trouble, so I am asking a minimum of them this year; simply getting the arithmetic right will get full marks.

b) Explain briefly the actions that must occur when a context switch happens in an operating system.

(2 marks)

A context switch is needed when one process is de-scheduled from the CPU and another process is selected to run. The entire state (CPU state and memory contents, plus anything else that is needed) of the old (de-scheduled) process needs to be saved somewhere (usually, but not necessarily, on disk) so that its execution can be resumed later, if appropriate. The entire state of the new process now needs to be (re-)instated to the CPU and memory (from wherever it has been saved) before execution of the new process can (re-)commence.
c) What does the term deadlock mean? How may deadlock occur?  

A (non-singleton) set of processes are all waiting, each one waiting for something to happen that can only be done by one of the other processes in the set. Since they are all waiting for this, none of them can progress and the situation can only persist. An example of how it can occur is two processes wanting to access two locks, A and B. One process tries to take lock A first and lock B second, the other process tries to take lock B first and lock A second. This may work okay, but if both start at the same time and each succeeds with its first lock take, neither second lock take is possible. This is an example that has been presented in a lecture, but any other valid example would be acceptable.

d) What is the key difference between a system call and a call to a library routine? Briefly explain why this is important.

A CPU can execute in one of two modes, namely user mode and system mode (a.k.a. privileged mode, supervisor mode or kernel mode); system mode allows instructions to do things that it would be unwise to let user code do. A call to a library routine is conducted entirely in user mode; at machine level, a simple subroutine call is made to start with, and the library call finishes with a simple subroutine return. A system call executes in system mode, so the initial call instruction must additionally change the mode from user mode to system mode; similarly, the return instruction must also change the mode, this time from system mode back to user mode. This is important in order to protect the OS (and other users) from misbehaviour in user code.

e) In Linux, how does a shell implement a pipe between commands?

A pipe links two distinct commands together, as follows; the standard output of the first command directly forms the standard input of the second command. This is relatively easy to do in Unix/Linux because any form of input or output looks like (and is treated as) a file. Some system store has to be assigned to buffer data in transit between the two processes executing the two commands, and the OS is responsible for managing this in an appropriate FIFO fashion. A good answer needs to explain what the terms “standard input” and “standard output” mean.
Section A

A2.  a) In the context of non-preemptive process scheduling for a single CPU, what are the three main states that a process may be in, and what are the three transitions amongst these states caused by?

A process may be ready, running or blocked. Incoming processes are put in the ready state and wait until they are selected to run. Only one process can be running at any one time; the process to run is selected by the scheduler by extracting its process descriptor from a list of waiting processes (if there are no waiting processes, the system idles until a new process arrives). The chosen process is dispatched (1st transition – process selected to run goes from ready state to running state). A process may exit the running state either because it terminates (and therefore leaves the system entirely) or because it blocks for I/O and goes into the blocked state (2nd transition – process goes from running state to blocked state). A blocked process remains in the blocked state until its I/O action has completed, whereupon the process again becomes ready, and is added to the list of waiting processes (3rd transition – process goes from blocked state to ready state). No other transition is possible in non-preemptive scheduling.

b) What additional transition between states is possible in preemptive process scheduling for a single CPU? Why is this new transition useful?

In preemptive scheduling, a process selected to run is given a maximum timeslice for which it may occupy the CPU. If the process is still running when the timeslice expires, the process is forcibly removed from the CPU (preempted) and set to the ready state at the tail of the ready queue; another process will be selected to run (unless the removed process is the sole ready process, in which case it will be given a new timeslice and set to the running state again). This is useful when a process would otherwise ‘hog’ the CPU by never blocking; if it cannot be preempted, it will cause all other ready processes to ‘starve’ until it deigns to free the CPU by terminating.
c) Three processes, A, B and C, have the following computational needs: Process A requires a 7 time-unit CPU burst followed by a 4 time-unit I/O burst then a 7 time-unit CPU burst before it terminates; Process B requires a 5 time-unit CPU burst followed by a 4 time-unit I/O burst then a 5 time-unit CPU burst before it terminates; Process C requires a 3 time-unit CPU burst followed by a 6 time-unit I/O burst then a 3 time-unit CPU burst before it terminates. Draw a diagram showing the states of these processes as they are run by a non-preemptive First Come First Served (FCFS) scheduler until they all terminate, assuming that they all start ready at time-unit 0 and are queued in the order A (first), B, C (last), and the time for a context switch is negligible. What is the average wait time (the time between entering the ready queue for the first time and starting to run on the CPU) for the three processes? What is the average turnaround time (the time between entering the ready queue for the first time and terminating) for the three processes?

Diagram should look something like the following (== is I/O burst (process is blocked), -- process is “ready but not running”, A, B, C means the named process is running):

```
AAAAAAA=========AAAAAAA
----------BBBBB==========BBBBB
----------CCCCCCCCCCCCCCCCC
123456789012345678901234567890
```

Wait for A is 0 time-units; for B is 7 time-units; for C is 12 time-units. Average wait is 6.333 time-units. Turnaround for A is 22 time-units; for B is 27 time-units; for C is 30 time-units. Average is 26.333 time-units.

d) The three processes in part c) are now subjected to a preemptive Round Robin (RR) scheduler with a time-slice of 3 time-units. Draw a diagram showing the states of the processes from start to finish. What is the average wait time for the three processes? What is the average turnaround time for the three processes?

(4 marks)
Diagram should look something like the following (== is I/O burst (process is blocked), -- is “ready but not running”, A, B, C means the named process is running):

AAA--AAAAA--=AAAAA
---BBB-----BBB==BBB--BB
-------CCC======CCC
123456789012345678901234567890

Wait for A is 0 time-units; for B is 3 time-units; for C is 6 time-units. Average wait is 3 time-units. Turnaround for A is 30 time-units; for B is 26 time-units; for C is 18 time-units. Average turnaround is 24.667 time-units.

e) In a hierarchical file system, a pathname defines how a file is to be located in the filestore. Describe in detail the algorithm used to locate a file when given its full (absolute) pathname.

A full (absolute) pathname consists of “components” separated by “separators” with a “separator” at the start. The full pathname has to be traversed, starting at the root directory. Each component in the pathname should be the name of a directory (apart from the final component, which can be the name of either a non-directory file or another directory). Each directory is itself a file, and the algorithm must read at least part of this file in order that it can check whether the next component is indeed a directory (or valid file if it is the final component) within the “current” directory. An illegal component will cause the algorithm to fail. If the algorithm is successful it will deliver a pointer to the final file (see below).

f) In a file system using index nodes (i-nodes), the value obtained by the algorithm in part e) is the identifier of the i-node that describes the target file. Explain how the i-node is found on disk.

Every file has a unique i-node identifier. The corresponding i-node has fixed size and is located in a special area of the disk. The i-node is accessed by using the i-node identifier as an index of the appropriate size into the special area.
g) Assume that each i-node contains up to 8 pointers to the first 8 blocks of the file on disk, then up to 4 pointers to *single indirect blocks* on disk that each contain up to 1024 pointers to the next blocks of the file on disk, and then up to one pointer to a *double indirect block* on disk that contains up to 1024 pointers to single indirect blocks on disk that each contain up to 1024 pointers to the next blocks of the file on disk. Any unused pointers contain a special ‘null’ value. What is the size (in blocks) of the largest possible file in this system?

(3 marks)

The i-node contains information about the file, including a list of the block numbers it uses. The first few blocks are listed explicitly. Then single-indirection is used (the given block number is used to access a block which contains another list of block numbers), and finally double-indirection (the given block number is used to access a block which contains a list of single-indirection block numbers which are used to access blocks as described above. In the system described, the sole double indirect block can contain up to $1024 \times 1024 = 1M$ pointers to distinct blocks. Each single indirect block can contain up to 1024 pointers to blocks, so 4 such blocks can point to up to 4096 more distinct blocks. Finally, the first 8 pointers point directly to up to 8 distinct blocks. The largest file can thus be of size $(1M + 4096 + 8)$ blocks.
Section B

B1. **Compulsory**

a) What is a page replacement algorithm?  

(2 marks)

B1(a)

Bookwork (2 marks):

a) What is a page replacement algorithm?  

(2 marks)

The following points should be covered to some degree in the answer:

In a virtual memory, the memory space is divided into pages. There are normally more pages than can fit into the real (RAM) memory of the system and thus some pages are held on background storage to be moved into main memory only when the CPU wants to access them. When a page needs to be moved into main memory, it is necessary to decide which page to reject to make space for it. The page replacement algorithm is used to make the decision of which page to reject.

2 marks for an answer that describes and contextualises all four and gives the salient facts in a sensible way; and provides a well-defined set of brief descriptions of each; 1 mark for some basic understanding (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s.): Lectures 13 Virtual Memory (3).

TOTAL marks (2 marks) [2]
b) In the context of operating systems, what is meant by the term *programmed I/O*? Give a brief answer with respect to a keyboard I/O device – flowcharts are not required.

(2 marks)

**B1.b**

Bookwork (2 marks):

b) .. what is meant by the term programmed I/O? (2 marks)

The following points should be covered to some degree in the answer:

The processor periodically *polls* the *status* of the I/O device *checking* on whether a data *transfer* should be made.

In this case, the *programmed I/O* code is periodically called by the *controlling* program.

If a character has been typed, it is *read* and *placed* in *memory*.

If character has not been typed, the programmed I/O code *exits* and the processor can get on with useful work.

**2 marks** for an answer that depicts all the salient facts in a sensible way; and correctly delineated and briefly described;

**1 mark** for some basic understanding (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s.): Lectures 14 Controlling Input and Output 1.

TOTAL marks (2 marks) [4]
c) An input/output (I/O) module in a computer system has an interface using Direct Memory Access (DMA). Explain what the DMA interface does.

(2 marks)

Example answer:- The following points should be covered to some degree in the answer:

- The DMA interface will transfer blocks of data between the I/O module and memory over the system bus without intervention from the processor.
- A DMA device is capable of reading from or writing directly to memory in the same way as a processor does.
- It is optimised for transferring blocks of data into the memory system.
- In our example of reading from a disk, the DMA will handle the transfer of a whole block of data without processor intervention.

2 marks for a totally correct; and explicit concise explanation, should mention: relevant information.
1 mark for some basic understanding (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s.): 15; Controlling Input and Output 2.

TOTAL marks (2 marks) [6]
d) One method for implementing virtual memory is *paged virtual memory*. The virtual memory translation procedure can be viewed as a series of steps. Explain the paged virtual memory translation procedure by outlining the sequence of steps for translating a virtual address to a physical address.

(2 marks)

B1.d.
Bookwork and analysis (2 marks).

d) … Explain the paged virtual memory procedure, by outlining the sequence of steps for translating a virtual address to a physical address. (2 marks)

Example answer: The following points should be covered to some degree in the answer:

1. The processor *generates* a logical address;
2. The *page number field* is used by the MMU to look to see whether the page is in memory or not;
3. If it is in memory, a *physical* address is computed by replacing the page number with the page frame number of where the page can be found;
4. Together with the offset this is used as a physical address to memory; and
5. If it is *not* in memory, the transfer is aborted (page fault) and the operating system will load the page from disk to memory.

2 marks for a totally correct explicit delineation of the process using keywords in context – plus a complete – well presented – description.
1 mark for some basic table (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s.): 11; Virtual Memory (1).
TOTAL marks (2 marks) [8]
e) Contrast what is meant by the terms multiprogramming and fixed partitions. (2 marks)

B1.e.
Bookwork (2 marks).

e) Differentiate between multiprogramming and fixed partitions. (2 marks)

Example answer:- The following points should be covered to some degree in the answer:

Multiprogramming:
Multiprogramming has been used in the past; to differentiate from an operating system (OS) running a single program [or Uniprogramming] and one that runs a number of programs concurrently (or multiprogramming).
The OS must first load the multiple programs (into memory [primary (physical) memory]).
The OS will then switch between them [the different programs]; this may be due to the program requiring I/O, or at regular intervals the OS will switch to another of the other programs.
When one of the programs is finished the OS bring in a new one.

Fixed partitions:
Fixed partition divides memory into fixed size blocks.
Fixed partitioning: involved partitioning the available primary memory into a number of regions with each region having a fixed size. The sum of the sizes of all regions [plus that used by the OS itself] equals the size of the primary memory.

2 marks for a totally correct using keywords in context and explicit description including good explanation of utilisation,
1 mark for some basic table (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s.): 10, Virtual Memory (1).

TOTAL marks (2 marks) [10]
Section B

B2. a) In the context of a segmented memory, name and briefly describe two alternative algorithms that can be used to determine where to place a new segment in a memory that has external fragmentation. (4 marks)

B2.a.
Application (4 marks).

a)  In the context of a segmented memory, name and briefly describe two alternative algorithms that can be used to determine where to place a new segment in a memory that has external fragmentation. (4 marks)

The following points should be covered to some degree in the answer:

An algorithm can be used to determine where to place the segment in a memory that has external fragmentation.

The operating system maintains a list of the addresses and sizes of all the holes and can use algorithms like the following two alternative algorithms:

i) Best Fit – scan the complete ‘list of holes’ and determine which best fits the segment; tends to produce a lot of small holes.

ii) First fit – scan the list until a [the first] hole is found that fits the segment.

4 marks for a concise description of what external fragmentation is and names [correctly] both algorithms and gives concise descriptions of them.

2 marks for half issues covered.

1 mark for some basic understanding (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s): 12; Virtual Memory (2), Segmented Virtual Memory.

TOTAL marks (4 marks) [4]
b) In the case of a computer system with real memory, explain why a
multiprogramming operating system would need to relocate code.

(4 marks)

B2.b.
Application (example re-coding) (4 marks):
The following points should be covered to some degree in the answer:

b) In the case of a computer system with real memory, explain why a multiprogramming
operating system would need to relocate code. (4 marks)

In a multiprogrammed system, it is necessary to enable more than one program [to run] in the
memory at one time. As it would be inefficient to save and load processes to and from disk
every time a process switch occurred. In general it is not possible to know which combination
of processes will be needed in memory; and it would be restrictive if a program had to be
loaded into the same set of addresses every time [it was swapped in]. Hence, a scheme is
necessary – where a program can be loaded starting at any convenient memory location, this is
“relocation”.

4 marks for majority of above; e.g. for an answer that orders all lines correctly.
3 marks for correct answer but not concise [enough].
2 marks for some information & for a right-lines approach.
1 mark for some basic understanding (or attempt).
Reference Learning Resources, Background Reading, and Lecture itself for detailed
information; Lecture(s) No(s.): Lectures 10, Memory Management (1).
TOTAL marks (4 marks) [8]
c) On a paged machine with three page frames available, a particular process makes accesses to the following virtual pages in the order shown below.

Page number: 3 7 1 3 2 1 3 7 0

Show the contents of the three page frames and the cumulative total number of page faults (PFs) after each memory access assuming that a least recently used (LRU) page replacement algorithm is in use and that the page frames are initially empty. The kind of diagram you should produce is depicted in Figure 1, below.

(4 marks)

<table>
<thead>
<tr>
<th>Page access:</th>
<th>3</th>
<th>7</th>
<th>1</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>3</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Most recent</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Second most</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Third most</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>PFs</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Figure 1. Typical diagram showing three page frames and the cumulative total number of page faults (PFs).
B2.c.
Application (4 marks).

c. ... Show the contents of the 3 page frames and the cumulative total number of page faults (PF) after each memory access assuming that an LRU page replacement algorithm is in use and that the page frames are initially empty... (4 marks)

Example answer: The following points should be covered to some degree in the answer:

<table>
<thead>
<tr>
<th>Access</th>
<th>3</th>
<th>7</th>
<th>1</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>3</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Most recent:</td>
<td>3</td>
<td>7</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>Second most:</td>
<td>-</td>
<td>3</td>
<td>7</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>Third most:</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>7</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Total PFs:</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

4 marks for a totally correct content, 3 marks for all 3 page frames totally correct and 1 mark for the cumulative total number of page faults totally correct.

2 marks for half issues covered,
1 mark for some basic calculations (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s.): 13; Virtual Memory (3).

TOTAL marks (4 marks) [12]
d) Given that the overall actions that a CPU takes to handle an interrupt can be summarised in six basic steps, list and briefly describe the six basic steps.

(4 marks)

Example answer:- The following points should be covered to some degree in the answer:

1. **External line interrupts processor**;
2. **Interrupt acknowledgement (IACK) cycle identifies the interrupting device**;
3. **Processor accepts interrupt after current instruction**;
4. The processor **stores the information necessary to restart the original program following the interrupt**;
5. **Interrupt Service Routine (ISR) is run for interrupting device until return from interrupt instruction is reached; then finally**…
6. **Stored information is reloaded into the processor, processor continues executing the original program as if nothing had happened**.

4 marks for a full description of the six steps.
2 marks for half issues covered.
1 mark for some basic understanding (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lectures 15: Controlling Input and Output 2.

TOTAL marks (4 marks) [16]
e) Given a 4GB virtual address space and an associated 512KB page size, calculate the number of pages in this virtual address space. **NOTE:** To gain full marks you must show full working.

(2 marks)

Example answer: The following points should be covered to some degree in the answer:

If the virtual address space is 4 GB and the block size is 512 KB there are:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>

2 marks for an answer that calculates the correct answer and is laid out correctly e.g.
2 marks for a correct answer and full working out,
1 mark for a ‘right lines’ approach. Moderate marks will be awarded in the case of correct application for a wrongly calculated value.

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s): 11; Virtual Memory (1).

TOTAL marks (2 marks) [18]
Given a physical address size of 1GB and an associated 256KB page size, calculate the number of page frames in the physical address space. **NOTE:** To gain full marks you must show full working. (2 marks)

Example answer: The following points should be covered to some degree in the answer:

Virtual address space 1GB and block size 256KB.
If the virtual address space is 1 GB and the page size is 256 KB there are:

_____ ________   

2 marks for an answer that calculates the correct answer and is laid out correctly e.g. 2 marks for a correct answer and full working out. 1 mark for a ‘right lines’ approach. Moderate marks will be awarded in the case of correct application for a wrongly calculated value. ½ marks for some basic understanding (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s.): 11; Virtual Memory (1).
TOTAL marks (2 marks) [20]
Section B

B3. a) Given the simple page table diagram in Figure 2 (below) and the 32-bit virtual address 0x00040003: Determine the physical address given the data in the simple page table.

Figure 2. Simple page table diagram.

B3.a

Simple problem solving (1 mark):

Determine the physical address given the data in the page table; (1 mark)

The following points should be covered to some degree in the answer:

a) The physical address (0x i o) produced from the page table is:
   0x i o = 0x 02 0003;

1 mark for fully accurate answer.

½ mark for some basic understanding (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s): Lectures 13: Virtual Memory (1).

TOTAL marks (5 marks) [5]
b) Demand paging can be viewed as a sequence of steps. Explain the demand paging procedure by outlining the sequence of steps.

(4 marks)

Demand paging can be viewed as a sequence of steps. Explain the demand paging procedure by outlining the sequence of steps. (4 marks)

1) The current process issues a memory request for a virtual (page) address that is not currently resident in the available page frames of the real memory.
2) The current process is halted and put in the blocked state.
3) The operating system will cause the page to be loaded from disk into a page frame. While waiting for the load to complete, the OS may run some other process(es) (multitasking).
4) After the page has been loaded and the original process has been reinstated by the process manager, the memory request that originally failed will be tried again.

4 marks for an answer that depicts all the salient facts in a sensible way; and the use of keywords in context; also describing all steps in detail;
2 marks for an answer that depicts a proportion of the salient facts in a sensible way; and the use of some keywords in context;
1 mark for some basic understanding (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s): Lectures 13: Virtual Memory (1).

TOTAL marks (5 marks) [5]
c) State and briefly describe the steps an operating system takes to load a single user program into memory and then execute it, given this is *uniprogramming*. (5 marks)

The following points should be covered to some degree in the answer:

OS like KOMODO or original MSDOS undertake the following steps when a program is loaded into memory:

1) **OS** given a *file name*;
2) **Loads** it from disc;
3) **Jumps** to start of program;
4) **Executes** the program; may do I/O etc. via OS;
5) **Returns** to **OS** when done.

**5 marks** for an answer that depicts all points, and the use of **keywords** in context;

**3 marks** for half of the facts;

**1 mark** for some basic understanding (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s.): 10; Memory Management (1).

TOTAL marks (5 marks)[10]
d) Explain what is meant by a critical region of a multi-threaded program. Explain what is meant by mutual exclusion for a critical region. Explain what a semaphore is and describe the P (procure) and V (vacate) operations that can be performed on it. Explain how a semaphore can be used to enforce mutual exclusion for a critical region of a multi-threaded program.

(4 marks)

Bookwork: A critical region is any part of one thread’s code which reads and then modifies some shared variables in a manner that is sensitive to interference by other threads trying to do a similar thing on the same shared variables; all potential read-modify-write actions have to be achieved as if atomically. Mutual exclusion means that a critical region of code can only be executed by a single thread at-a-time. A semaphore is a shared integer variable that can only be accessed by the special operations P (for procure) and V (for vacate). For semaphore S, P(S) will repeatedly try to find S in state S>0; if it ever succeeds, it will immediately and atomically reduce S by 1 then complete (thereby entering the critical region of code, which immediately follows the P(S) operation). When execution of the critical region is complete, a call to V(S) should be made. This simply increments the value in S (again atomically), thereby potentially making it possible for another thread’s P(S) operation to see it in state S>0. For a semaphore initialised to value 1 (a binary semaphore), this will ensure mutually exclusive access to critical regions in multiple threads.

e) In a certain system, the execution of three threads, A, B and C, is synchronised using two semaphores, S1 and S2, as shown below. Both semaphores are initialised to zero and are used only in the sections of code shown below. The threads share the variables x and y.

<table>
<thead>
<tr>
<th>Thread A</th>
<th>Thread B</th>
<th>Thread C</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>P(S1)</td>
<td>P(S2)</td>
<td>x = 0</td>
</tr>
<tr>
<td>x = x + 1</td>
<td>y = 2</td>
<td>V(S1)</td>
</tr>
<tr>
<td>x = x - 2</td>
<td>y = y - 1</td>
<td>V(S2)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>y = 3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

i) What happens to the variable x, and why?

(1 mark)

The only thread that can execute is C. Thread A blocks until S1 is vacated, which can only be done by C. Thread B blocks until S2 is vacated, which can also only be done by C. Thread C thus initialises x to 0 and then vacates S1. Thread A is now able to execute beyond P(S1). Further execution of C or B can only affect variable y, so what happens to x is simply what Thread A does to it. Since it has been initialised to 0, x = x + 1 sets x to 1, then x = x - 2 sets x to -1. No further changes are possible, so final value of x is -1.
ii) There can be more than one outcome for variable y. What can happen, and why?

(2 marks)

After C has vacated S1 and released A, it now vacates S2, thereby potentially releasing B. Threads B and C will execute in an arbitrarily interleaved order, and they both access y, so we have to consider all possible interleavings. If C gets in first, y will be set to 3, then only B makes further changes, causing y to be set to 2, then 1 (first possibility: y is 1). If B gets in first, then y will be set to 2; however, there are now two possible cases, depending whether B or C gets in next. If B continues, then y will be set to 1 first, then C will set it to 3 (second possibility: y is 3). If C gets in before B can continue, then y will be set to 3, then B will decrement it to 2 (third possibility: y is 2). No other orderings are possible, so y can be any one of 1, 2 or 3 at the finish.

iii) Rewrite the code above, keeping the assignments to x and y unchanged, but changing how and/or where S1 and S2 are used so that, when all three threads have finished, x has the same final value as before, and y has the value 1. Explain why your code achieves this.

(1 mark)

To achieve this, we want to force the first possibility above to happen. Thus we want to force C to act on y before B gets started. This can be achieved simply by moving the V(S2) in Thread C below the y = 3 assignment. B cannot start until C has finished with y. Everything else remains the same.

iv) Rewrite the code above, keeping the assignments to x and y unchanged, but using any number of semaphores you need in any way so that, when all three threads have finished, x has the same final value as before, and y has the value 3. Use the minimum number of extra semaphores needed to achieve this. Explain why your code works.

(2 marks)

Now we want to force the second possibility in part ii) above to happen. This requires B to execute to the end before C sets y to 3. There are many ways to force this, but using the minimum number (which is zero, in this case) of extra semaphores forces something like the following: A remains the same; B has no P(S2) at the start, but adds a V(S2) at the end; C has no V(S2), but instead does P(S2) at the same place (in the original code), so that it has to wait until B has finished before setting y to 3. Anything that works will get 1 mark, provided there is an adequate explanation.

END OF EXAMINATION