**COMP25111**

Two hours

**UNIVERSITY OF MANCHESTER**
**SCHOOL OF COMPUTER SCIENCE**

**Operating Systems**

**Date: Thursday 22nd January 2015?**

**Time: 14:00 – 16:00?**

---

**Please answer Question ONE and any TWO Questions of your choice from the other THREE questions provided**

**For full marks your answers should be concise as well as accurate. Marks will be awarded for reasoning and method as well as being correct**

---

This is a CLOSED book examination

The use of electronic calculators is NOT permitted

[PTO]

1. **Compulsory**

a) In the worst case, after making an access for data from a disk, the operating system has to wait for the disk head to move all the way across the disk (the maximum "seek time") and then for the disk to make a full rotation through 360 degrees (the maximum "rotation time") before the data starts arriving. Given a disk that makes 7200 revolutions per minute and has a maximum seek time of 20 milliseconds, what is the worst case wait time for data? How many clock cycles does this represent for a 3GHz CPU? What is the implication of this wait for the operating system.

(2 marks)

7200 rpm = 120 revolutions per second, so maximum rotation time is $1/120^{th}$ s = 8.333 ms. Worst case wait is thus 28.333 ms. 3GHz clock does 3x10E6 cycles per ms, so number of clock cycles per worst case access is 3x28.333 = 85 million. Implication is: do something else while you are waiting, otherwise you waste a substantial amount of computing power.

b) In the context of process scheduling, explain briefly what a *CPU burst* is and what an *I/O burst* is. What is a *CPU bound* process, and what is an *I/O bound* process? Why is it a good scheduling strategy to give higher priority to I/O bound processes?

(3 marks)

A CPU burst is where a process uses the CPU continuously for some (relatively long) period of time. An I/O burst is where a process is waiting for data transfer to occur between CPU and peripheral devices. A CPU-bound process is mostly executing in CPU bursts, while an I/O-bound process is mostly executing in I/O bursts. I/O-bound processes voluntarily yield the CPU more readily than CPU-bound processes, thereby ensuring that "someone else gets a go" more often. Progress towards the final goal of completing execution is thus quicker (and there are other benefits, such as reduced contention for resources).

c) What does the term *deadlock* mean? How may deadlock occur?

(2 marks)

A set of processes are all 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.

d) Briefly explain the difference between a *process* and a *program*.

(1 mark)

A process is a complete executing program, including its virtual memory space as defined by page tables, links to IO/files, and whatever else is needed. As execution proceeds, this changes dynamically. A program is the object code and constant initial data produced by the compiler/linker/loader; it does not contain any space for computed data, including any stack or heap. Nor does it contain 'in-flight' information such as file and device descriptors etc. It is therefore static.

e) Why is the size of the time-slice in pre-emptive scheduling algorithms chosen to be significantly higher than the time taken for a context switch?

(2 marks)

A context switch is essentially wasted time; nothing useful is being done. This overhead can be lessened by reducing both the number of, and the time taken by, context switches. Once the context switch time has been fixed, only reducing the number is possible. Increasing the size of the time slice automatically reduces the number of context switches.

f) What is meant by the term *programmed I/O* in the context of operating systems? Give a brief answer – flowcharts are not required.

(2 marks)

1.f

Bookwork (2 marks):

f) What is meant by the term "programmed I/O" in the context of operating systems? Give a brief answer – flowcharts are not required. (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. Programmed [or polled] I/O's **not** interrupt driven I/O. In this case, programmed I/O corresponding to the periodically interrogating the peripheral IO. For example, if a character has been typed, it is read and placed in memory utilising programmed IO. Basically the programmed IO routine reads the status register (of the peripheral IO device), then if it has the appropriate bit set [in the status reg.] the routine reads the data register of the device, and finally stores this data in memory.

**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 10 Memory Management (1).

TOTAL marks (2 marks) [2]

g) Briefly explain what is meant by the terms *fixed partitions* and *multiprogramming*.

(2 marks)

1.g

Bookwork (2 marks):

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

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

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.

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.

**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).: Lecture 11: Memory Management (1).

TOTAL marks (2 marks) [4]

h) A segmented memory, shown in Figure 1h, is to swap out segment 2 (of size 20kB) for segment 7 (of size 14kB). Draw a diagram depicting the segmented memory layout before and after the swap. Comment on any effect that swapping segment 2 for segment 7 has on the overall memory space.

(2 marks)

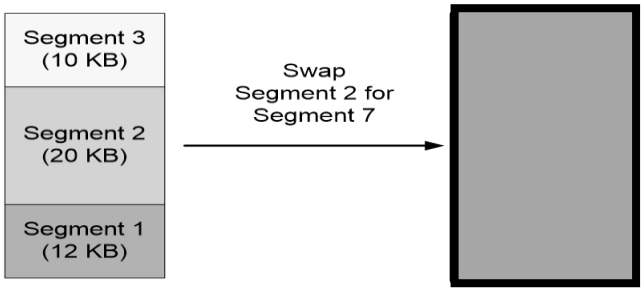


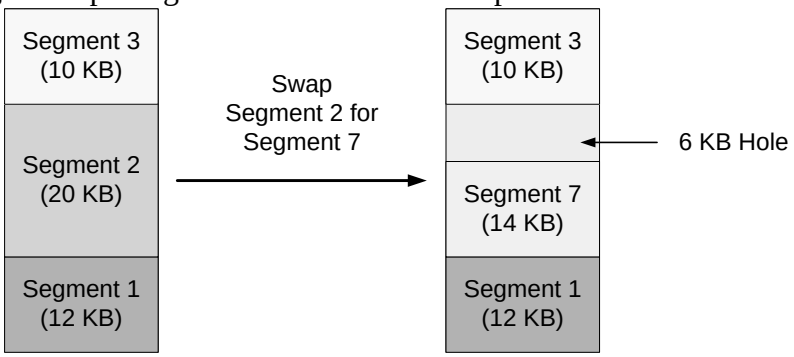

Figure 1h. A segmented memory, showing 3 segments.

1.h.
Application, critique (2 marks),

h) Given that a segmented memory, in figure 1.h., is to swap out segment 2 (20kB) for segment 7 (14kB). Draw up a diagram depicting the new segmented memory layout; before and after the swap. Comment on any effect swapping segment 2 for segment 7 has on the overall memory space. (2 marks)

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

i) Diagram depicting before and after the swap:




(1 mark)

ii) The effect of swapping segment 2 for 7 is:

Is that 6KB is freed; this leads to a 6kB hole in the memory; where a 20kB segment (2) is swapped out and a 14kB segment (7) is loaded in; this (hole) is called "External Fragmentation;" which can occur, where memory space is (potentially) wasted due to 'holes' in the physical memory. (1 mark)

**2 marks** for a totally correct part i & ii; and explicit concise explanation for second part of ii, should mention: hole, & external fragmentation.
**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 (2 marks) [6]

i) In the context of relocation of memory addresses what are *base* and *limit* registers and what do they enable a virtual memory to undertake?

(2 marks)

1.i.
Bookwork (2 marks),

i) In the context of relocation of memory addresses what are base and limit registers and what do they enable a virtual memory to undertake? (2 marks)

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

**1/ Base and limit registers are** special hardware registers.
When a process is run, the base register is loaded with the physical location where the process begins in memory.
The limit register is loaded with the length of the process' memory.
In other words, they define the logical [allowable] address space [in the physical memory].

**2/ They enable** virtual addresses to be relocated to physical address space.
This is achieved by adding the base address (in the base address register) to the address from the CPU [virtual address] to calculate its actual physical address.

**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]

j) What is a *page fault*? Describe how a page fault is handled by the memory management unit and the operating system.

(2 marks)

1.j.
Bookwork and analysis (2 marks),

j) What is a 'page fault'? Describe how a page fault is handled by the Memory Management Unit and the operating system. (2 marks)

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

In a paged virtual memory system, pages may have copies in real memory or may exist only on backing storage (usually disk or secondary memory).
If the MMU attempts an address translation but the page table indicates that there is no copy in real memory, this is a page fault. The MMU interrupts the CPU and traps to an OS page fault handler.
This is responsible for both selecting a page to reject from real memory – and writing it to disk if it has changed since it was last read – and organising the read of the required page from disk to real memory.
It then updates the page tables and transfers control back to the original program to re-execute the instruction which caused the fault.

**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]

2. a) In the context of process scheduling, what are the three main states that a process may be in, and what are the four main transitions amongst these states caused by?

(3 marks)

A process may be **ready**, **running** or **blocked**. Only one process can be running at any one time (or one per CPU in a multicore system); the process to run is selected by the scheduler. Whenever a process exits the running state, the next process to run is chosen by the scheduler from those that are ready and is launched. A process may exit the running state because it terminates (and therefore leaves the system), because it is removed due to expiry of its timeslice (in which case it reverts to the ready state), or because it blocks for I/O (in which case the process goes into the blocked state until the I/O action has completed, whereupon the process again becomes ready).

b) Three processes A, B and C all alternate between CPU bursts and I/O bursts of 5 time units each. Draw a diagram showing the states of these processes as they are run by a pre-emptive Round Robin scheduler for a total of 30 time-units, assuming that they all start ready at time-unit 0 and are queued in the order A (first), B, C (last), the time-slice adopted by the scheduler is 3 time-units, and the time for a context switch is negligible. For what fraction of the time is no process running because they are all blocked? How many context switches are performed during the 30 time-units?

(4 marks)

Diagram should look something like the following (=== is I/O burst, -- is "ready but not running"):

```
AAA------AA=====AAA------AA===
---BBB-----BB=====-BBB-----BB=
------CCC----CC=======CCC----C
               *
```

CPU is idle at * above, behaviour repeats after 16 time-units, so fraction idle CPU is $1/16^{th}$ (half a mark for $1/30^{th}$). 11 context switches.

c) Change the time-slice to 4 time-units and *either* repeat part b) for the new setup and, *or* otherwise, explain why a 3 time-unit time-slice produces a better schedule than a 4 time-unit time-slice for these processes. Are there any conditions under which the 4 time-unit time-slice might produce a schedule as good as that for the 3 time-unit time-slice (for these processes)?

(5 marks)

4 time-unit diagram should look something like the following:

```
AAAA--------A=====AAAA--------
----BBBB-----B========BBBB----
--------CCCC--C==========CCCC
                ***
```

CPU is idle at `***` above, behaviour repeats after 18 time-units, so fraction idle CPU is $1/6^{th}$ (half a mark for $1/10^{th}$). 9 context switches (should count either first or last). The 3 unit time-slice gets through one cycle of A/B/C in 16 units, while the 4 unit time-slice takes 18 units, so the former is evidently better. The description in part b) assumes that a context switch takes negligible time, so the final part invites candidates to explore what changing this assumption might lead to. The diagrams below assume a 2 unit context switch time and show that it is possible to get equally good behaviour (a 1 unit context switch still favours the 3 unit time-slice):

```
AAA------------AA==========AAA------------
-----BBB-----------BB=====------BBB-------
----------CCC----------CC============CCC--

AAAA--------------A======---AAAA-----------
------BBBB-----------B==========BBBB-----
------------CCCC--------C==============CCC
```

Both repeat after 27 time-units. Only the very best will get to this result, but only 2 of the 5 marks are reserved for it (1 for observing that a non-zero context switch time will affect matters).

d) Explain what a *semaphore* is and describe the operations that can be performed on it.

(3 marks)

Bookwork: 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 is synchronised using three semaphores, S1, S2 and S3, as shown below. Semaphores S1 and S2 are initialised to zero, while semaphore S3 is initialised to 1. All three semaphores are used only in the sections of code shown below.

| Thread A | Thread B | Thread C |
|---|---|---|
| … | … | … |
| P(S2) | P(S1) | P(S3) |
| P(S2) | P(S2) | V(S2) |
| x=3*x+4 | x=x+7 | x=x*5 |
| V(S1) | V(S1) | V(S2) |
| V(S2) | V(S2) | V(S3) |
| … | … | … |

i) If the variable x is defined as an integer shared variable, initialised to 1, and is not assigned a value in any other sections of the code apart from those shown above, what will be its value when all three threads have finished executing? What will be the values of the three semaphores S1, S2 and S3? Justify your answers.

(3 marks)

The only thread that can execute is Thread C which tries to procure S3 which is the only semaphore in a procurable state. Thread B blocks until S1 is vacated, which can only be done by Thread A. Thread A blocks until S2 is vacated.; this can only be done by Thread C, which vacates S2 first and thereby potentially releases Thread A. Thread A now tries another procure for S2 which can again only be vacated by Thread C. This happens only after thread C has executed its update of x, so this is the first action (x=1*5=5). Thread C now vacates S2, freeing Thread A to perform its action on x, followed by S3, which reverts to the procurable state. Nothing can happen in Thread B until Thread A vacates S1, and Thread C does not touch x again, so the second action on x is in Thread A (x=3*5+4=19). Now Thread A vacates S1 and finally S2 and then finishes executing its code. The vacate of S1 releases Thread B which now tries to procure S2, which will be successful once Thread A has vacated it. Then and only then Thread B can do its update of x, so this is the third action (x=19+7=26). It is also the final update to x since no more are left, so the final value of x is 26. Thread B then vacates S1 then S2, so all three semaphores will be vacated at the end (S1=S2=S3=1).

ii) Would there be any impact on the execution of the three threads if the two P operations in Thread B were interchanged (i.e. P(S2) is followed by P(S1))? Justify your answer.

(2 marks)

Thread C vacates S2 twice to help Thread A reach its update of x, so one of these vacates could now be trapped by Thread B, thereby causing deadlock (because Thread A can no longer proceed). Of course, this might not happen, so the outcome is non-deterministic.

3. a) To address the question: "What happens in an interrupt?", it is important to align the appropriate set of steps to the proper interrupt sequence. Figure 3a shows a list of out-of-order steps relating to an interrupt sequence.

Put the steps into the correct order so they enable an interrupt to sequence through the proper interrupt steps.

(3 marks)

| |
|---|
| a) return |
| b) code executed |
| c) interrupt acknowledgement |
| d) interrupt service routine |
| e) processor saves registers |
| f) interrupt line |
| g) interrupt vector |

Figure 3a. Table showing a sequence of out-of-order steps for an interrupt.

3.a.
Application (example re-ordering) (3 marks):
The following points should be covered to some degree in the answer:

The typical ordered set of STEPs is:
a/ interrupt line
b/ interrupt acknowledgement
c/ processor saves registers
d/ interrupt vector
e/ interrupt service routine
f/ code executed
g/ return

From the previously out-of-order STEPs:

g/ return
f/ code executed
b/ interrupt acknowledgement
e/ interrupt service routine
c/ processor saves registers
a/ interrupt line
d/ interrupt vector

3-2 marks for majority of above; e.g. for an answer that orders all lines correctly,
1 mark for some basic understanding (or attempt).

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

TOTAL marks (3 marks) [3]

b) Given that the now in-order steps enable an interrupt to follow the appropriate sequence, the table in Figure 3b shows a set of (out-of-order) steps for an interrupt sequence. Copy the re-ordered table from part a) into your answer book (plus a "detailed explanation of step" column) and then describe in full what happens at each step. Clearly put each step into context and describe exactly what function each performs.

(5 marks)

| **Steps [please re-order]** | **Detailed explanation of step** |
|---|---|
| a) return | |
| b) code executed | |
| c) interrupt acknowledgement | |
| d) interrupt service routine | |
| e) processor saves registers | |
| f) interrupt line | |
| g) interrupt vector | |

Figure 3b. A keyword table, requiring re-ordering and adding a comprehensive explanation of each step.

3.b.
Application (example re-coding) (5 marks):
The following points should be covered to some degree in the answer:

The typical fully commented ordered interrupt sequence is:

| **Steps** | **Detailed explanation of step** |
|---|---|
| a/ interrupt line. | The I/O device will signal that an interrupt has occurred by using the "**interrupt line**;" |
| b/ interrupt acknowledgement . | The processor then sends a special type of memory read (called an "**interrupt acknowledgement**"); |
| c/ processor saves . | The "**processor saves**" the current value of the Program Counter and the contents of its registers into memory; |
| d/ interrupt vector. | The value that the processor received from the IACK cycle identifies what device interrupted the processor; this is known as the "**interrupt vector**;" |
| e/ interrupt service routine | The interrupt vector is used to access a table that holds the starting addresses of all programmes that handle interrupts; the interrupt vector is used to find the starting address for the appropriate device handler; this program is called an "**Interrupt Service Routine**" (ISR) and contains code that handles the type of interrupt; |
| f/ code executed | This **code** is now **executed** until a special instruction (*return from interrupt*) is executed; and finally |
| g/ return | When the "**return**" from interrupt instruction is reached; the PC and registers that were stored earlier are loaded back into the processor. |

5-4 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 mark for some information & for a right-lines approach,
1 marks for some basic understanding (or attempt).
Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s).: Lecture 15: Input/Output (2).

TOTAL marks (5 marks) [8]

c) With respect to relocation and swapping, state why the ability to relocate programs is useful in the context of swapping?

(2 marks)

3.c.
Application of your knowledge - Bookwork (2 marks),

c) What is 'swapping' in the context of operating system memory management and why is it useful in the context of relocation of programs in memory? (2 marks)

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

In any multiprogrammed system, it is highly useful if any subset of a large number of processes can be resident in real memory at any one time. This means that a user can have the illusion that he/she is running a large number of processes, even if they won't fit into main memory all together. This is achieved by 'swapping'.

It is useful [in the context of relocation of programs in memory] as the processes can be moved to and from [in and out of] background storage (disk) automatically by the system thus providing the above illusion.

It is the ability of code to be relocated (i.e. position-independent code) that enables a process to be swapped out of location A and later swapped back in to location B. Hence, it is the ability to relocate that enables swapping.

**2 marks** for a totally correct content, and all issues addressed comprehensively;
**1 mark** for some basic knowledge (or attempt).

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

TOTAL marks (2 marks) [10]

d) Given an 8GB virtual address space and associated 256KB page size, calculate the *number of pages* in this virtual address space. **NOTE**: To gain full marks you must show full working.

(2 marks)

3.d.
Application (2 marks),

d) Given an 8G address space and associated 256K page sizes; calculate the number of pages in the 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 8 GB and the block size is 256 KB there are:

$$\text{Number of Pages} = \frac{\text{Address space}}{\text{Page size}}$$

$$= \frac{8\ GB}{256\ KB} = \frac{8{,}589{,}934{,}592}{262{,}144} = \frac{2^{33}}{2^{18}} = 2^{33-18} = 2^{15} = 32{,}768 = 32k\ (256K)\ pages$$

**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.

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

TOTAL marks (2 marks) [12]

e) Given a physical address size of 2GB and associated 128KB 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)

3.e.
Application (2 marks),

e) Given a physical address size of 2G and associated 128K block 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 2GB and block size 128KB.
If the virtual address space is 2 GB and the page size is 128 KB there are:

$$\text{Number of Page frames} = \frac{\text{Address space}}{\text{Block size}}$$

$$= \frac{2\ GB}{128\ KB} = \frac{2{,}47{,}483{,}648}{131{,}072} = \frac{2^{31}}{2^{17}} = 2^{31-17} = 2^{14} = 16{,}384 = 16k\ (128\ K)\ pages\ frames$$

**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.,
**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) [14]

f) One page replacement policy is the "not recently used" (NRU) algorithm; answer the following with respect to this policy:

i) State the function of the "R" and "M" bits, which appear in a page table using the NRU algorithm.

(2 marks)

3.f
Bookwork (2 marks),

i) State the function of the "R" and "M" bits, which appear in a page table using the NRU algorithm. (2 mark)

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

The R and M bits in a page table are used [in the following way]:

The R bit is set when the page is referenced; while
The M bit is set when the page is modified (or written to).

**2 marks** for an answer that depicts all the salient facts in a sensible way; and the use of keywords in context: 'referenced', and 'modified';
**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) [16]

ii) Describe in detail how the NRU algorithm works, stating explicitly how it utilises the "R" bit.

(4 marks)

Bookwork (4 marks):

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

How it works:

1. At fixed intervals, the clock interrupt triggers and clears the referenced bit (R = 0) of all the pages.
2. Referenced bit marks pages referenced in interval.
3. So during interval if page referenced R=1 then it is used.
4. If not R=0 then it is NOT used; and

At the end of interval is a candidate for replacement.

**4 marks** for an answer that depicts all the salient facts in a sensible way; and the use of keywords in context;
**2 marks** for an average answer;
**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: Virtual Memory (3).

TOTAL marks (4 marks) [20]

4. a) On a paged machine with 3 physical page frames available, a particular process makes access to the following virtual pages in the order shown:

0, 3, 7, 1, 3, 2, 1, 3, 7

Show the contents of the 3 physical page frames and the cumulative total number of page faults (PFs) after each memory access, assuming that an LRU page replacement algorithm is in use and that the 3 physical page frames are initially empty. The kind of diagram you should draw is shown in Figure 4a.

(4 marks)

| Access: | 0 | 3 | 7 | 1 | 3 | 2 | 1 | 3 | 7 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Most recent: | X | X | X | X | X | X | X | X | X | ) | Contents of |
| Second most : | X | X | X | X | X | X | X | X | X | ) | page frames |
| Third most: | X | X | X | X | X | X | X | X | X | ) | in memory |
| Total PFs: | X | X | X | X | X | X | X | X | X | | |

Figure 4a. Diagram showing 3 page frames and the cumulative total number of page faults.

4.a

Application (4 marks),

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

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Access: | 0 | 3 | 7 | 1 | 3 | 2 | 1 | 3 | 7 | | |
| Most recent: | 0 | 3 | 7 | 1 | 3 | 2 | 1 | 3 | 7 | ) | Contents of |
| Second most : | - | 0 | 3 | 7 | 1 | 3 | 2 | 1 | 3 | ) | page frames |
| Third most: | - | - | 0 | 3 | 7 | 1 | 3 | 2 | 1 | ) | in memory |
| Total PFs: | 1 | 2 | 3 | 4 | 4 | 5 | 5 | 5 | 6 | | |

Note: 3, 1, & 3 are not page faults (PF) as in physical memory.

**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) [4]

b) When a page fault occurs the CPU uses a DMA unit to move pages from/to disk (termed *disk I/O*); in this context:

i) Name and briefly describe the two registers in the DMA controller that are normally used during the disk I/O DMA process.

(2 marks)

4.b.i
Bookwork [Differentiate] (2 marks):

b.i. Name and briefly describe the two registers that are normally used during the DMA process.. (2 marks)

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

The disk I/O has two registers:

Command register where commands such as: READ, WRITE, FORMAT can be written;

Status register that indicates whether the disk is ready for the data or not.

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

**2 marks**, *1 for names and 1 for descriptions.*
TOTAL marks (2 marks) [6]

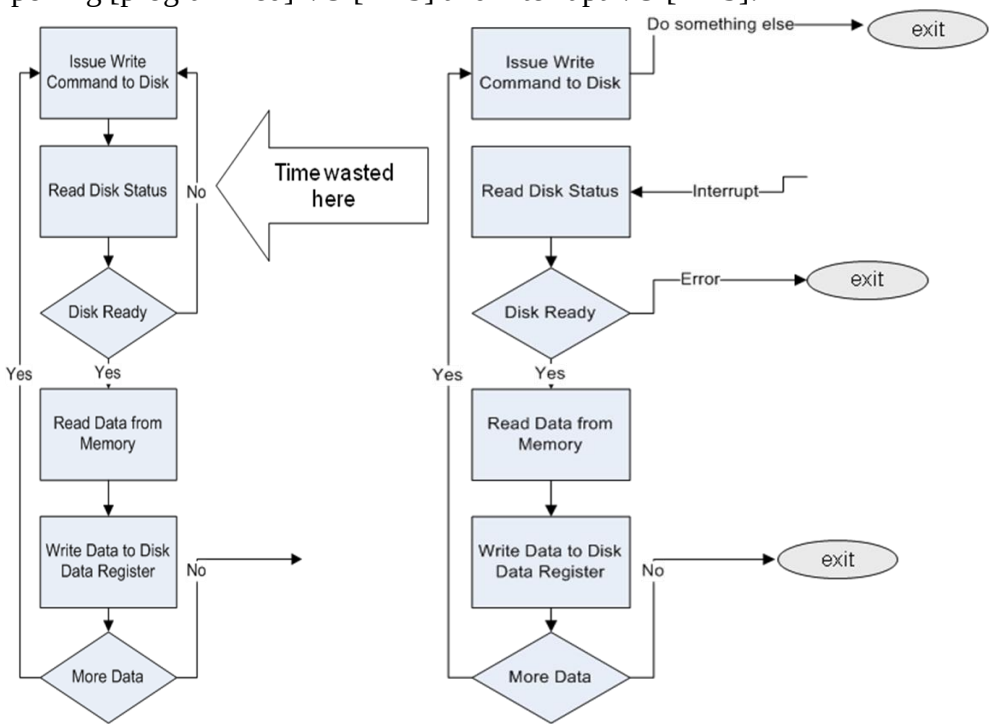
ii) Transfer of data to a disk can actually be undertaken by programmed input/output (programmed I/O) or by interrupt I/O. Draw two flow charts [diagrams] that depict the sequence of events needed to perform writing to disk using programmed I/O and using interrupt I/O.

(4 marks)

4.b.ii.
Bookwork [Differentiate] (4 marks):

(i) Transfer of data to a disk can actually be undertaken by programmed: input/output (I/O); or interrupt I/O. Draw two flow charts [diagrams] that depict the sequence of events needed to perform disk writing using polling [programmed] I/O and interrupt I/O. (4 mark)

**Example answer:-** The following points should be covered to some degree in the answer:
The two diagrams below depict:-
polling [programmed] I/O [LHS] and interrupt I/O [RHS]:




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

**4 marks** *for an answer that depicts all steps in both diagrams*.
**2 marks** *for correct answer but not detailed [enough]*,
**1 mark** *for a right-lines approach*,
**½ marks** *for some basic understanding (or attempt).*
TOTAL marks (4 marks) [10]

c) Describe clearly the algorithm used to locate a file referred to by a full (absolute) pathname in a hierarchical file system.

(4 marks)

A full (absolute) pathname consists of "components" separated by "separators". The full pathname has to be traversed, starting at the root. Each component in the pathname should be the name of a directory (apart from the final component, which can be the name of a (non-directory) file). Each directory is a file, and the algorithm must read (part of) this file in order that it can check whether the next component is indeed a directory (or 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).

d) In a file system using a File Allocation Table (FAT), the value obtained by the algorithm in part c) is the number of the first block of the file. Describe carefully how such a file system thereby provides access to the whole file.

(3 marks)

The FAT is an array of block numbers that is indexed using the block number of the "current" block, starting from the first block number that the algorithm delivers. The FAT delivers the number of the next block (if zero, there is no next block). Once the current block has been finished with, the current block number is updated to be the next block number and the next next block number will be read from the FAT.

e) In a file system using index nodes (i-nodes), the value obtained by the algorithm in part c) is the i-node number for the file. Describe carefully how such a file system thereby provides access to the whole file.

(3 marks)

Every file has a unique i-node number. The corresponding i-node (located in a special area of the disk) 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).

**END OF EXAMINATION**