NYU
New York University

Operating Systems

Revision - Final

Mohamed Zahran (aka Z)
mzahran@cs.nyu.edu
http://www.mzahran.com

About the Exam

- Take-home
- Cumulative
- Submit through Brightspace
  - Must upload a pdf file
- Exam available from Wed Dec 21st at 12pm EST and for 24 hours.
- No solutions are accepted as email attachment.
- Submissions are accepted once (unlike assignments and midterm)
- We will be very strict with plagiarism.
- Your answers must be focused. You will be penalized for:
  - Wrong answers
  - Irrelevant information

Problem 1

The OS virtualizes the CPU (gives a process the illusion that it has the whole CPU for itself), and the memory (a process thinks it has the whole memory for itself).

- Does the OS virtualize I/O too? Explain.
- Why is virtualization important?

Problem 1

The OS virtualizes the CPU (gives a process the illusion that it has the whole CPU for itself), and the memory (a process thinks it has the whole memory for itself).

- Does the OS virtualize I/O too? Explain.

Yes

- For storage: files is the way to virtualize disks.
- For other I/O devices: there is a generic I/O layer

- Why is virtualization important?

Problem 1

The OS virtualizes the CPU (gives a process the illusion that it has the whole CPU for itself), and the memory (a process thinks it has the whole memory for itself).

- Does the OS virtualize I/O too? Explain.
- Why is virtualization important?
  - Make programming easier as it relieves the programmer from low-level details.
  - Makes compilation easier as the compiler does not need to worry about other processes existing in the system.
  - Better protection as the OS is the only software that sees the big picture and can protect processes from each other.

Problem 2

Suppose we have a multicore processor with four cores. Suppose process x starts execution on core 0. When process x finished execution and exited can it be on a different core just before the exit?

Justify your answer.

Problem 2

Suppose we have a multicore processor with four cores. Suppose process x starts execution on core 0. When process x finished execution and exited can it be on a different core just before the exit? Justify your answer.

Yes, this can happen.

The process can be removed from a CPU due to I/O or time slice ending, then brought back to a different CPU based on the OS scheduling algorithm.

Problem 3

Suppose we have n processes in the system. Does that mean there are n processes in the physical memory at once? Explain.

Problem 3

Suppose we have n processes in the system. Does that mean there are n processes in the physical memory at once? Explain.

Not necessarily

We may run out of physical memory and the OS may need to put the pages belonging to some non-running processes in the disk as part of the page-replacement policy in virtual memory.

Problem 4

Suppose we have an instruction that loads 4 bytes from memory and saves them in a register. What is the maximum number of page faults that this instruction can cause? Ignore page fault that may be related to loading the instruction itself.

Problem 4

Suppose we have an instruction that loads 4 bytes from memory and saves them in a register. What is the maximum number of page faults that this instruction can cause? Ignore page fault that may be related to loading the instruction itself.

If the 4 bytes spans two pages (i.e. part at the end of a page while the second part at the beginning of the following page) then we can get up to two page-faults.

[Compilers try to reduce the chance of this scenario by having the data *aligned*]

Problem 5

When a system contains a TLB, which of the following page table structure is better?

1. single-level
2. two-level
3. three-level
4. Will not make a difference

Justify your answer.

Problem 5

When a system contains a TLB, which of the following page table structure is better?

1. single-level 2. two-level 3.three-level

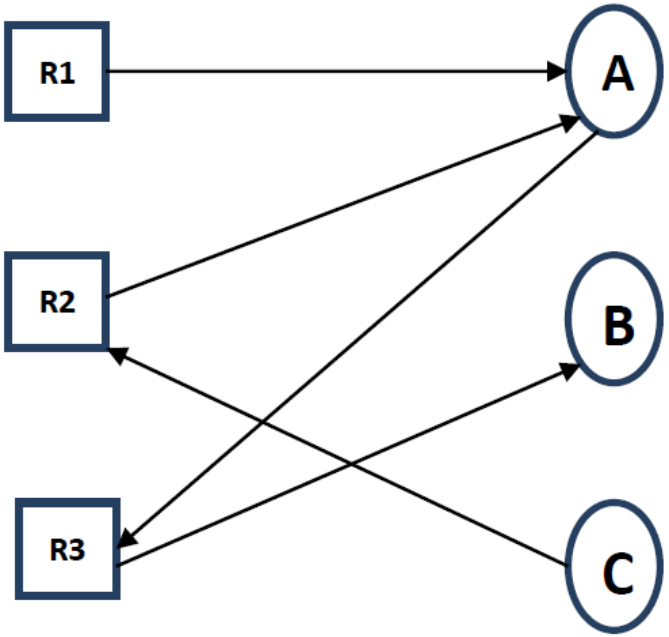
4. Will not make a difference

Justify your answer.

Multi-level page table's main goal is to reduce the amount of data related to page table that needs to be in the memory at any point of time. TLB is a hardware piece that has the goal of reducing trips to the page table. So the two goals are not related.

Problem 6

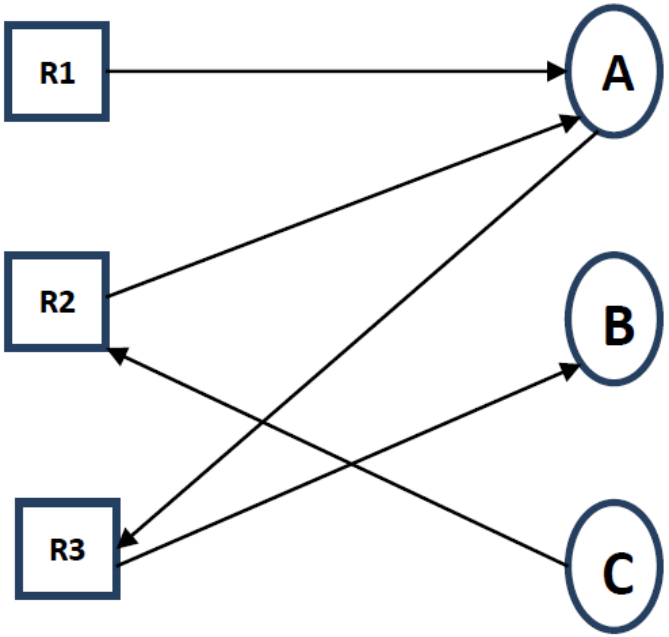
Given the following resource allocation graph where circles represent processes A, B, and C and squares represent different resources. We have one instance from each resource R1, R2, and R3. An arrow from a resource to process means the process holds that resources (e.g. process A holds resources R1 and R2). An arrow from a process to a resource means the process wants that resource (e.g. process A wants resource R3).



Do we have a deadlock in the above situation? If no, show a possible sequence of execution till all processes end. If yes, explain why.

Problem 6

Given the following resource allocation graph where circles represent processes A, B, and C and squares represent different resources. We have one instance from each resource R1, R2, and R3. An arrow from a resource to process means the process holds that resources (e.g. process A holds resources R1 and R2). An arrow from a process to a resource means the process wants that resource (e.g. process A wants resource R3).



Do we have a deadlock in the above situation? If no, show a possible sequence of execution till all processes end. If yes, explain why.

There is no deadlock.
B finishes and releases R3.
A takes R3, finishes, and releases R1,R2, and R3
C takes R2 and finishes.

Problem 7

What is the implication of a smaller page size on overall system performance? Explain

Problem 7

What is the implication of a smaller page size on overall system performance? Explain

Smaller page size → more pages → more entries in the page table → bigger page table.

This means it requires bigger parts from the physical memory and also it takes more time to reach an entry.

Problem 8

Suppose a machine has 48-bit virtual address and 32-bit physical address. If page size is 8KB, how many entries are in the page table if it is single level? Show detailed steps.

Problem 8

Suppose a machine has 48-bit virtual address and 32-bit physical address. If page size is 8KB, how many entries are in the page table if it is single level? Show detailed steps.

Page table has an entry for each virtual page number

8KB → 2^13 and virtual address = 48 bits so: virtual page number = 48 – 13 = 35 bits

The page table has: 2^35 entries

Problem 9

Suppose in a single-thread process, the thread is blocked waiting for the keyboard. If a fork() is executed by this process, will the thread of the new process also be blocked for the keyboard? State yes or no and give the reason for your choice in no more than two sentences.

Problem 9

Suppose in a single-thread process, the thread is blocked waiting for the keyboard. If a fork() is executed by this process, will the thread of the new process also be blocked for the keyboard? State yes or no and give the reason for your choice in no more than two sentences.

This situation cannot happen.

It is a single-thread process, and this thread is blocked, then how will it execute a fork()?