WHITE QUIZ
First question

- How does Nooks *recover* from an extension failure?
- What is the *main limitation* of this approach?
First question

- How does Nooks *recover* from an extension failure?
- What is the *main limitation* of this approach?

- *Nooks recovers from an extension failure by killing and restarting the failing extension.*
- *The approach does not work for all extensions.*
Second question

- Give one reason for the relatively high overhead of Nooks.
Second question

- Give one reason for the relatively high overhead of Nooks.

- Each switch of lightweight protection domains involve fetching a new page table and results in a TLB flush.
Third question

- How do *mapped files* reduce the number of context switches during file accesses?
Third question

- How do *mapped files* reduce the number of context switches during file accesses?
  - *They bring file blocks directly into the address space of the process accessing them.*
  - *They eliminate the context switches required to transfer data between the system I/O buffer and the process space.*
Fourth question

- In Mach, what are the *inheritance attributes* of
  
a) The *code segment* of a process:

b) Any of its *mapped files*:
Fourth question

- In Mach, what are the *inheritance attributes* of
  
  a) The *code segment* of a process:  *Shared*

  b) Any of its *mapped files*:  *Shared*
Fifth question

- Why TLB sizes have remained small while memory sizes have been exploding?
Fifth question

- Why TLB sizes have remained small while memory sizes have been exploding?

  - Because larger TLBs would be slower and TLBs must be very fast
Sixth question

- Consider a virtual memory system with 4 KB pages, 24 GB of RAM and a TLB with 512 entries.
- What would be the *coverage* of this TLB?
Sixth question

- Consider a virtual memory system with 4 KB pages, 24 GB of RAM and a TLB with 512 entries.
- What would be the coverage of this TLB?

$$512 \times 4\text{KB} = 2\text{MB}$$
Seventh question

- How do Navarro et al. propose to handle dirty superpages?
- Why?
Seventh question

- How do Navarro et al. propose to handle *dirty superpages*?
- Why?

- They propose to disband superpages the first time one of their base pages gets modified.
- Otherwise we would have to save the whole superpage when it gets expelled from main memory.
Eighth question

- What can cause *false sharing* in a multicore system
- How can we solve the problem?
Eighth question

- What can cause *false sharing* in a multicore system
- How can we solve the problem?

- *False sharing occurs when two distinct data items appear in the same cache line, they are accessed by two different threads and one of them is frequently updated.*
- *We should move one of the two items to a different address.*
YELLOW QUIZ
First question

- Why TLB sizes have remained small while memory sizes have been exploding?
First question

- Why TLB sizes have remained small while memory sizes have been exploding?

  - *Because larger TLBs would be slower and TLBs must be very fast*
Second question

- Consider a virtual memory system with 4 KB pages, 16 GB of RAM and a TLB with 256 entries.
- What would be the coverage of this TLB?
Second question

- Consider a virtual memory system with 4 KB pages, 16 GB of RAM and a TLB with 256 entries.
- What would be the **coverage** of this TLB?

\[ 256 \times 4 \text{KB} = 1 \text{MB} \]
Third question

- How do Navarro et al. propose to handle *dirty superpages*?
- Why?
Third question

- How do Navarro et al. propose to handle *dirty superpages*?
- Why?

- They propose to disband superpages the first time one of their base pages gets modified.
- Otherwise we would have to save the whole superpage when it gets expelled from main memory.
Fourth question

- How do *mapped files* reduce the number of context switches during file accesses?
Fourth question

- How do **mapped files** reduce the number of context switches during file accesses?
  - *They bring file blocks directly into the address space of the process accessing them.*
  - *They eliminate the context switches required to transfer data between the system I/O buffer and the process space.*
Fifth question

In Mach what are the two possible *inheritance attributes* for the *data segment* of a process?
Fifth question

- In Mach what are the two possible inheritance attributes for the data segment of a process?

  a) Copy  
  b) Share
Sixth question

- What can cause *false sharing* in a multicore system
- How can we solve the problem?
Sixth question

- What can cause *false sharing* in a multicore system?
- How can we solve the problem?

- *False sharing occurs when two distinct data items appear in the same cache line, they are accessed by two different threads and one of them is frequently updated.*
- *We should move one of the two items to a different address.*
Seventh question

- How does Nooks *recover* from an extension failure?
- What is the *main limitation* of this approach?
Seventh question

- How does Nooks *recover* from an extension failure?
- What is the *main limitation* of this approach?

- *Nooks recovers from an extension failure by killing and restarting the failing extension.*
- *The approach does not work for all extensions.*
Eighth question

Why does Nooks XPC mechanism use *calls by value and result*?
Eighth question

Why does Nooks XPC mechanism use *calls by value and result*?

- *The call by value and result delays updates until the procedure has completed and allows the Nooks XPC mechanism to check their validity.*