CS250P Computer Systems Architecture

Homework 3, Due: December 12th, 222

Submit via gradescope! Link will be available soon.

Reminder: While the questions in this homework ask the same kind of questions I would ask in the finals, the questions here require more research, reading, and more thinking in general, since the homework has a more lenient time limit!

Question 1: Privileged Instructions

Privileged instructions are necessary to support a modern operating system like Linux or Windows. Let's assume a user-privilege process is running on such a machine, and discovers the host processor hardware does not support an instruction it requires to run. Let's assume this instruction performs machine-privilege operations, so simply changing the user software is not a solution.

As discussed in class, one solution to such a problem is to catch the unsupported instruction exception and emulate the unsupported instruction in the software exception handler. If the host operating system currently does not support emulation of this particular instruction, How can the user-privileged process achieve emulated instruction execution?

① Read the "mtvec" register to locate the Interrupt Descriptor table, and update the handler for unsupported instructions.

② Update the "mstatus" register to elevate itself to a higher privilege, to catch the unsupported instruction exception.

③ The emulated instruction solution is impossible in this case, unless the OS is modified to accommodate.

④ The emulated instruction solution is fundamentally impossible in this case.

Question 2: Virtual Memory and Page Table

(a) Consider a hierarchical page table implementation where the size a page is 4 KiB (4096 Bytes), and the size of each page table is limited also to a 4 KiB page. Assuming 4 Byte PTEs, how many layers do we need to be able to address a 32-bit address space?

(b) In the hierarchical page table introduced above, how much memory space would the page table require, if a process is only using 32 MB of contiguous memory?

(c) Copy-on-write is a method that drastically reduces the memory copy overhead during process fork. What flags should the PTE have, to support copy-on-write? (Select all that applies)

① Valid

② Dirty

③ Read only

④ Memory resident

⑤ Executable

Question 3: Inverted Page Table

First, do some reading on Inverted Page Tables

1. Link: https://en.wikipedia.org/wiki/Page_table#Inverted_page_tables
2. Link: https://www.8bitavenue.com/page-table-vs-inverted-page-table/

Primarily, inverted page tables aim to reduce the memory overhead of page tables.

(a) Can inverted page tables be implemented on a generic x86 or RISC-V machine, whose hardware implement a non-inverted, hierarchical page table? (Yes/No)

(b) Why, or why not?

(c) The performance problem of inverted page tables can be significantly solved using a hash-based approach. There must be a reason x86-64 and RISC-V use hierarchical page tables instead of inverted page tables. What do you think that is? (Please be concise!)

Question 4: Memory Consistency Model

The following figure is from the Wikipedia article for Dekker's algorithm, which implements a mutex between two processes.

```
p0:
   wants_to_enter[0] ← true
   while wants_to_enter[1] {
      if turn ≠ 0 {
         wants_to_enter[0] ← false
         while turn ≠ 0 {
           // busy wait
         }
         wants_to_enter[0] ← true
      }
   }

   // critical section
   ...
   turn ← 1
   wants_to_enter[0] ← false
   // remainder section
```

```
p1:
   wants_to_enter[1] ← true
   while wants_to_enter[0] {
      if turn ≠ 1 {
         wants_to_enter[1] ← false
         while turn ≠ 1 {
           // busy wait
         }
         wants_to_enter[1] ← true
      }
   }

   // critical section
   ...
   turn ← 0
   wants_to_enter[1] ← false
   // remainder section
```

(a) Would this algorithm be safe with Total Store Order (TSO) memory consistency? Remember, TSO adds a queue-semantic write buffer which is not propagated immediately to other cores in the system.

(b) Why or why not?

(c) Is it possible to make Dekker's algorithm safe again, under TSO, without adding new instructions to the RISC-V base ISA?

(d) Why or why not?