MITx 6.004.3x
Computation Structures 3: Computer Organization

Help

Course Progress Dates Discussion

Course / 17. Virtualizing the Processor / Lecture Videos (38:19)

Previous ✓ ✓ ✓ ✓ ✓ Next

LE17.2 SVC Handlers

Bookmark this page

Calculator

LE17.2.1: Wait SVC Handler

0.0/1.0 point (ungraded)

The WAIT() supervisor call expects the address of a user's memory location in R0. The WAIT handler examines the value of that memory location and if it's positive, decrements the value in the memory location by 1 and resumes execution of the user's program at the instruction following the WAIT() SVC. If the memory location is not positive, the handler should arrange to re-execute the WAIT() SVC next time the user's program runs and then yield the remainder of the current time slice.

Your goal is to fill in the required code for Wait_h(), the kernel-mode handler for the WAIT SVC. You can use the kernel procedure GetUserLocn(addr) to get the values of a location in the user's memory and SetUserLocation(addr,v) to set the value of a memory location. The kernel subroutine Scheduler() is also available for your use. Select the correct choice of instructions to include in the **if** and **else** clauses of the Wait_h() stub shown below. Make sure to include all the required instructions. Assume that the selected instructions are executed in the order they appear.

```
Wait_h() {
  int locn = UserMState.Regs[0]; // user location to test
  int value = GetUserLocation(locn); // current value
  if (value > 0) {
    // if clause code

  } else {
    // else clause code

  }
}
```

if clause code (select all that apply):

- ☐ UserMState.Regs[XP] = UserMState.Regs[XP] - 4;
- ☐ SetUserLocation(locn,value)
- ☐ SetUserLocation(locn,value-1) ✓
- ☐ GetUserLocation(locn)
- ☐ Scheduler();

else clause code (select all that apply):

- ☐ UserMState.Regs[XP] = UserMState.Regs[XP] - 4; ✓
- ☐ SetUserLocation(locn,value)
- ☐ SetUserLocation(locn,value-1)
- ☐ GetUserLocation(locn)
- ☐ Scheduler(); ✓

Calculator

Explanation

If the value at the memory location is positive, then the value should be decremented by one and then exec

If the value at the memory location is positive, then the value should be decremented by one and then execution of the user's program should resume at the instruction following the WAIT() SVC. This is achieved by making a call to SetUserLocation with a value of value - 1 inside the if clause.
If the memory location is not positive, then the handler should re-execute the WAIT() SVC the next time the user's program runs. This is achieved by decrementing the value of the user's XP register by 4 so that it re-executes the supervisor call. This is followed by a call to Scheduler() in order to schedule the next process to run right away rather than waiting for the time slice of the current process to complete before moving on to the next process.

Submit

Answers are displayed within the problem

LE17.2.2: YieldN SVC Handler

0.0/1.0 point (ungraded)
The Yield() SVC can be used in user-mode programs on a time-sharing system to give up the remainder of their current time slice. The kernel implementation of the Yield simply calls the kernel Scheduler() routine to choose another process to execute. When the yielding process is next scheduled, user-mode execution resumes with the instruction following the Yield() SVC.

Complete the code for the handler for a new SVC, YieldN(), which expects a numeric value, N, in the user’s R0 and behaves as if the user program had contained N consecutive Yield() SVCs. When execution resumes following the completion of YieldN(), R0 should contain 0. Select the correct choice of instructions to include in the **if** and **else** clauses of the YieldN_h() stub shown below. Make sure to include all the required instructions. Assume that the selected instructions are executed in the order they appear.

```
YieldN_h() {
  if (UserMState.Regs[0] > 0) {
    // if clause code

  } else {
    // else clause code

  }
}
```

if clause code (select all that apply):

- ☐ UserMState.Regs[0] = UserMState.Regs[0] - 1; ✓
- ☐ UserMState.Regs[0] = UserMState.Regs[0] - N;
- ☐ UserMState.Regs[0] = 0;
- ☐ UserMState.Regs[XP] = UserMState.Regs[XP] - 4; ✓
- ☐ UserMState.Regs[LP] = UserMState.Regs[LP] - 4;
- ☐ YieldN();
- ☐ Scheduler(); ✓

else clause code (select all that apply):

- ☐ UserMState.Regs[0] = UserMState.Regs[0] - 1;

Calculator

☐ UserMState.Regs[0] = UserMState.Regs[0] - N;

☐ UserMState.Regs[0] = 0; ✓

☐ UserMState.Regs[XP] = UserMState.Regs[XP] - 4;

☐ UserMState.Regs[LP] = UserMState.Regs[LP] - 4;

☐ YieldN();

☐ Scheduler();

Explanation
The handler for the YieldN() SVC can simulate N consecutive Yield() SVCs by repeating the YieldN() instruction N times. To do this, each time it executes the instruction, it needs to decrement the remaining count by 1. This is done by decrementing register R0 by 1. It then needs to re-execute the call to YieldN(), this is achieved by decrementing the XP register by 4 to point to that instruction rather than the one immediately after it. Each time through, it needs to call Scheduler() in order to yield to other processes. This happens as long as the value of R0 is > 0. When this is not true, that means that we are done and R0 should be set to 0 in case it wasn't 0 at this point.

Submit

Answers are displayed within the problem

Discussion

Hide Discussion

Topic: 17. Virtualizing the Processor / LE17.2 SVC Handlers

Add a Post

Show all posts by recent activity

? How does wait SVC work?
What's the purpose of >...The WAIT handler examines the value of that memory location and if it's positive, decrements the value in... 3

< Previous Next >

Calculator

News

Legal

Terms of Service & Honor Code
Privacy Policy
Accessibility Policy
Trademark Policy
Sitemap
Cookie Policy
Your Privacy Choices

Connect

Idea Hub
Contact Us
Help Center
Security
Media Kit

GET IT ON Google play
Download on the App Store

Calculator