**Slide 1**

## Administrivia

- Recurring meetings for M/W office hours created. Information shared by e-mail and reachable from my home page (`http://www.cs.trinity.edu/~bmassing`).

  First of these is today. It will be my first try at using the one-at-a-time feature. If it doesn't work, fallback is Google Chat. `chat.google.com` seems to work better than clicking "Chat" from TMail. I'm kind of hoping at least a few of you will try it, so I get a better sense of how it will work!)
- I just relayed to all of you a message from Dr. Fogarty about Discord, as a possible way to connect a bit more with your classmates. Looks promising?

**Slide 2**

## Exam 1, Revisited

- I prefer that you do this during one of the Zoom meeting times:
  - Thursday March 26, 5pm to 7pm.
  - Friday March 27, 4pm to 6pm.

  I've set up the meetings like office hours, with one-at-a-time access.
- I plan to put the exam in our shared folder during those times. If you need to take the exam at another time, let me know and I'll make it available then.
- If you have an SAS accommodation allowing you extra time, let me know. My idea is that you'd start the exam during one of the scheduled periods and work for your allowed time.
- As mentioned Monday — PDF from me to you (shared Google Drive folder), PDF from you to me (e-mail or your "graded work" folder).
- Questions?

**Slide 3**

## Sidebar(?): Parallel Execution and Synchronization

- A lot of commodity hardware these days features multiple processing units ("cores") sharing access to memory. One reason for this is that in theory we can make individual applications faster by splitting computation up among processing elements.
- Having processing elements share memory makes parallel programming easier in some ways but has risks ("race conditions"). Avoiding the risks requires some way to control access to shared variables (e.g., to implement notion of "lock").

**Slide 4**

## Parallel Execution and Synchronization, Continued

- Most texts on operating systems discuss synchronization issues and present several solutions ("synchronization mechanisms"), some rather high-level and others not.

  (Why is this in O/S textbooks? because O/Ss typically have to manage "processes" executing concurrently, either truly at the same time or interleaved.)
- The most primitive can (with some simplifying assumptions) be implemented with no hardware support. But hardware support is very useful.

**Slide 5**

## Sidebar: Why is Implementing a Lock Hard?

- It might seem like it would be straightforward to implement a lock — just have an integer variable, with value 0 meaning "unlocked" and anything else meaning "locked". And then you "lock" by looping until the value is 0, then setting to nonzero:

```
while (lock != 0) {}
lock = 1;
```

  and "unlock" by setting back to 0.

- But this doesn't work! (Why not?)

**Slide 6**

## Instructions for Synchronization

- Key goal in designing hardware support for synchronization is to provide "atomic" (indivisible) load-and-store. This allows writing a low-level implementation of "lock" idea.
- Many architectures do this with a single instruction (e.g., "test and set" or "compare and swap"). Requires two accesses to memory so may be difficult to implement efficiently.
- MIPS approach: Same idea, but using a pair of instructions, `ll` ("load linked") and `sc` ("store conditional").

Slide 7

## MIPS Instructions for Synchronization

- `ll` loads a value from memory and somehow remembers the location and value. Syntax:

  ```
  ll reg1, displacement(reg2)
  ```

  Operands used as for `lw`.

- `sc` stores a value into memory — *IF* the location has not changed since a previous `ll` from that address.

  ```
  sc reg1, displacement(reg2)
  ```

  Operands used almost as for `lw`, except that `reg1` is set to indicate whether the store "succeeded" (i.e., value had not changed since `ll`). So one can regard a (`ll`, `sc`) pair as forming a single atomic load/store.

- (How to make this work? Hardware designers' problem! glib answer but maybe all we can do in this course.)

Slide 8

## MIPS Instructions for Synchronization — Example

- Example of use (from textbook p. 122):

```
again:
    addi    $t0, $zero, 1       # $t0 <- value to place in lock
    ll  $t1, 0($s1)             # $t1 <- old lock value
    sc  $t0, 0($s1)             # try to set new lock value
    beq $t0, $zero, again       # $t0 == 0 means store failed
                                #   (so try again)
    add $s4, $zero, $t1         # $s4 <- old lock value
```

**Slide 9**

## Minute Essay

- Are you able to get to all the resources I want you to have access to for the exam? sample solutions to homeworks and quizzes, your graded work, etc.?
  And when I say "I'll put the exam in the shared Google Drive folder for the course — okay?
- We've been doing online teaching for a few days now. How's it going for you?