Some usual stuff

- Today's office hours in 006 at 4:30
- Grading
  - Homework 2 back today (average: 45/50)
  - I have old HW1/project 1/midterms, pick up at the end
- Project 3 out next Monday
  - VM emulation and page replacement: nice and easy
- Today:
  - Midterm problem 2
  - Project 2 – clear up a few issues; questions
  - Some more VM practice problems

Midterm question 2

```
Class ReaderWriterLock {
        Semaphore mutex = 1,
        OkToRead = 0,
        OkToWriter = 0;
        int AR=0, // # of readers that have acquired a read lock
        WR=0, // # of readers waiting to acquire a read lock
        AW=0, // # of writers that have acquired a write lock
        WW=0; // # of writers that are waiting for a write lock
…
```


Readers

```
void AcquireReadLock() {
        P(mutex); // P == decrement
        if (AW == 0) {
                V(OkToRead); // V == increment
                AR++;
        } else WR++;
        V(mutex);
        P(OkToRead);
}
void ReleaseReadLock() {
        P(mutex);
        AR--;
        if ((AR == 0) && (WW > 0)) {
                V(OkToWrite);
                AW++; WW--;
        }
        V(mutex);
}
```


Writers

```
void AcquireWriteLock() {
        P(mutex);
        if (AW + AR == 0) {
                V(OkToWrite);
                AW++;
        } else WW++;
        V(mutex);
        P(OkToWrite);
}
```

```
void ReleaseWriteLock() {
        P(mutex);
        AW--;
        if (WW > 0) {
                V(OkToWrite);
                AW++; WW--;
        } else {
                while (WR > 0) {
                        V(OkToRead);
                        AR++; WR--;
                }
        }
        V(mutex);
}
```


Issues

a) Why is this deadlock-free?
b) scheduling policy…
c) fix writer starvation
   i. writers run exclusively
   ii. readers may run concurrently with other readers
   iii. when any reader is granted a readlock, then all readers waiting for a readlock **at that time** are also granted readlocks
   iv. no additional readers are granted readlocks if any writer has requested a writelock
d) fix reader starvation
   …

P2 part 4 – interrupts vs locks

- In general, what are the problems with just disabling interrupts everywhere?
- It's probably ok to disable interrupts in most of your library code
  - It's short
  - We don't care about performance
- Most important thing: get it working

Webserver w/user threads

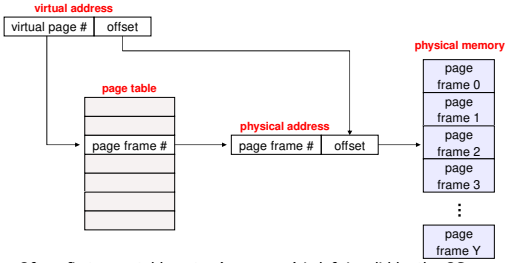
- Might not work!!!
  - Synchronous I/O
  - E.g. accept() problem
    - yield() in main thread after handing off the socket id makes things better
- Use pthreads for part 5 and part 6
- We won't test sioux with user threads

Part 6

- Keep in mind clients and server probably have the same bandwidth if run on two CSE hosts!
- Best to run webclient with –l 1 (i.e. one loop per client) for 5 and 25 clients
  - Easier to explain what you see
  - If you want to average, run webclient multiple times

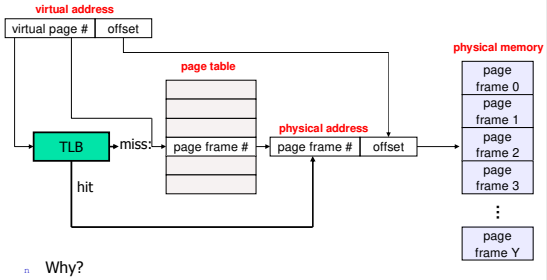
Project 2 – last questions?

VM exercise 1



- Often, first page table entry (page zero) is left invalid by the OS
  - why?
- How can we use paging to set up sharing of memory between two processes?

TLBs



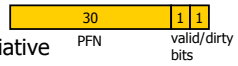
- Why?
  - No TLB: Average number of memory accesses per virtual addr ref: **2**
  - With a TLB (99% hit rate): 0.99*1 + 0.01*2 = **1.01**

VM exercise 2

- Consider a program consisting of 25% load/store instructions.
  - What is the base # of memory accesses per executed instruction with no virtual memory?
  - Assuming a VM setup with three-level page tables and no TLB, how many extra memory accesses per instruction executed does this program need?
  - What if we have a TLB with a 95% hit rate? With 100% hit rate?

What does a TLB look like?

- Consider the VM setup from midterm question 3:
  - 4K pages, 64-bit arch, 3-level 4K page tables.
  - Let's assume a PTE is



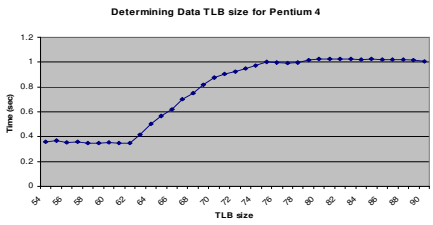
- Assume TLB is fully-associative

TLB size

- How would you experimentally determine TLB size?

TLB size

- Step through a huge array in such a way that each access goes to a different page
- Step an increasing number of times
- look for timing increases



Page table/TLB examples

- Intel x86
  - 4K pages (common) or 4M pages (jumbo pages)
  - Two-level page tables
  - Pentium 4: 64-entry TLB
- AMD-64
  - still 4K or 2M pages
  - Four (!) PT levels for 4K pages; three for 2M pages
  - Two-level TLB (40 entries/512 entries)
    - Why? What does this buy you?
- Alpha
  - 8K page size
  - Three-level page table, each one page
  - Alpha 21264: 128-entry TLB

Example Page Sizes

Computer	***Page Size***
Atlas 512	48-bit words
Honeywell-Multics	1024 36-bit words
IBM 370/XA and 370/ESA	4 Kbytes
VAX family	512 bytes
IBM AS/400	512 bytes
DEC Alpha	8 Kbytes
MIPS	4 kbytes to 16 Mbytes
UltraSPARC	8 Kbytes to 4 Mbytes
Pentium	4 Kbytes or 4 Mbytes
PowerPc	4 Kbytes
IA-64	4 Kbytes to 4 Gbytes