CS 31: Intro to Systems
Course Recap

Kevin Webb
Swarthmore College
April 30, 2015
Reading Quiz
Reading Quiz

Just kidding. Did I scare you?
Course Survey

• >95 % have responded

• Thank you!

• Extra quiz/participation grade to be dropped
Final Exam

- Thursday, May 14, 2:00 PM. SCI 199
- Similar format to the midterm
- You get ~100% more time
- Exam is ~20% longer
- ~2/3 post-midterm material
Review Session

A. Monday

B. Tuesday

C. Wednesday

D. Thursday (morning / early afternoon)

E. Don’t care and/or don’t plan to attend
A. early morning
B. late morning
C. early afternoon
D. late afternoon
E. evening
day Review Session

A. early morning
B. late morning
C. early afternoon
D. late afternoon
E. evening
Course Recap

• This course was a vertical slice of computer
  – From lowest level: simple logic
  – To high level: large, complex programs run on OS

• Big goal: make complex machine easier to use
  – Hide details with the right abstractions
  – Improve performance when possible
Lowest Level

• Storing and representing data
  – 2’s complement integers, floating point, etc.
  – Arithmetic using bits

• Logic gates: simple hardware
Hardware Abstraction: Circuits

- Combining gates to build specific circuits
  - arithmetic (adders, ALUs)
  - storage (latches, registers)
  - control (fetch, decode, multiplex)
CPU

• Combine circuits to create a CPU
  – Periodic clock: fetch, decode, execute instructions
Instruction Set Architecture

• ISA defines CPU / software interaction
  – Machine properties (# registers, address modes)
  – Method for controlling hardware (assembly lang)

\[
x = y >> 3 | x \times 8
\]

```assembly
movl -8(%ebp), %eax       # R[%eax] ← x
imull $8, %eax            # R[%eax] ← x*8
movl -12(%ebp), %edx      # R[%edx] ← y
rshl $3, %edx             # R[%edx] ← y >> 3
orl %eax, %edx            # R[%edx] ← y>>3 | x*8
movl %edx, -8(%ebp)        # M[R[%ebp-8]] ← result
```
Conventions

• Agreed upon system for using ISA
  – e.g., manipulating the stack, register meaning

  Callee’s frame.
  Callee’s local variables.
  Caller’s Frame Pointer
  Return Address
  First Argument to Callee
  …
  Final Argument to Callee
  Caller’s local variables.

  Caller’s frame.
  …
  Older stack frames.
  …
Storage and Memory

• Allocating memory (stack vs. heap)

• Pointers
  – malloc() / free()
  – address of (&)
  – dereferencing
  – arrays, 2D arrays
The Memory Hierarchy

- **Local secondary storage (disk)**: Larger, Slower, Cheaper per byte; ~100 M cycles to access.
- **Remote secondary storage (tapes, Web servers / Internet)**: Slower than local disk to access.
- **Cache(s) (SRAM)**: ~10’s of cycles to access.
- **Main memory (DRAM)**: ~100 cycles to access.
- **CPU instructions can directly access**
- **On Chip Storage**: 1 cycle to access.
Caching

• Improve performance by keeping a small memory for frequently-used data
  – Many parameters inform address division (tag, idx)
    • direct map vs. associative
    • block size

• Exploit major idea: **Locality**
  – temporal / spatial
Operating System

• Software supports: making programs easy/fast

• Three major abstractions:
  1. Process
  2. Thread
  3. Virtual memory

• Mechanism vs. policy
Processes

• Program in execution
  – fork() / exit() to create / terminate

• Represents all of the resources of a task
  – virtual address space (process memory)
  – open files
  – process ID, other accounting info

• One or more threads of execution
Threads

- Execution context within a process
- Independently scheduled
Virtual Memory

• Allow processes to behave as if they have the entire memory of the machine
• Translate from virtual (fantasy) address to physical
Virtual Memory

• Use disk to store data that hasn’t been used lately
  – (Another instance of exploiting locality)
Mechanism & Policy

• Mechanism: the ability to do something

• Policy: rules for governing the mechanism(s)

<table>
<thead>
<tr>
<th>Mechanism</th>
<th>Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context switching</td>
<td>CPU scheduling</td>
</tr>
<tr>
<td>Cache eviction</td>
<td>Cache replacement policy</td>
</tr>
<tr>
<td>VM paging to disk</td>
<td>Page replacement policy</td>
</tr>
</tbody>
</table>

• “Best” policy usually varies by workload!
Concurrency & Parallelism

• Single CPU core performance has plateaued
  – Hardware giving us more CPU cores instead

• Programmer’s responsibility to use them!

• Big opportunity for performance benefits!
Multi-threading in Practice (pthreads)

• Not always intuitive to reason about...

• Potential problems
  – race conditions
  – deadlock
  – priority inversion, etc.

• Requires careful synchronization
Next Systems Courses

• Fall 2015
  – Networks
• Spring 2016
  – Compilers
  – Parallel & Distributed
• Fall 2016
  – TBD
• Spring 2017
  – Databases
Questions?

• Thank you for a great semester!