Reading Quiz
Reading Quiz

Just kidding. Did I scare you?
Final Exam

• Thursday, May 12, 2:00 PM. SCI 199

• Similar format to the midterm

• You get ~100% more time
• Exam is ~15% longer

• ~2/3 post-midterm material
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

![Logic gate diagrams](image-url)
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)

\[
x = y \gg 3 \mid x \ast 8
\]

\[
\begin{align*}
\text{movl} & \ -8(\%ebp), \ %eax & \# R[\%eax] & \leftarrow x \\
\text{imull} & \ $8, \ %eax & \# R[\%eax] & \leftarrow x \ast 8 \\
\text{movl} & \ -12(\%ebp), \ %edx & \# R[\%edx] & \leftarrow y \\
\text{rshl} & \ $3, \ %edx & \# R[\%edx] & \leftarrow y \gg 3 \\
\text{orl} & \ %eax, \ %edx & \# R[\%edx] & \leftarrow y \gg 3 \mid x \ast 8 \\
\text{movl} & \ %edx, \ -8(\%ebp) & \# M[R[\%ebp-8]] & \leftarrow \text{result}
\end{align*}
\]
Conventions

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

![Diagram of stack frame conventions]

- Callee’s local variables.
- Caller’s Frame Pointer
  - Return Address
  - First Argument to Callee
  - Final Argument to Callee
- Caller’s local variables.
- Older stack frames.
Storage and Memory

• Allocating memory (stack vs. heap)

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

- **Smaller, Faster, Costlier per byte**
  - CPU intrs can directly access
  - On Chip Storage
  - 1 cycle to access

- **Larger, Slower, Cheaper per byte**
  - Main memory (DRAM)
  - ~100 cycles to access

  - Flash SSD / Local network

  - Local secondary storage (disk)
  - ~10’s of cycles to access

  - Remote secondary storage (tapes, Web servers / Internet)
  - ~100 M cycles to access
  - Slower than local disk 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
Questions?

• Thank you for a great semester!