CSE 351: The Hardware/Software Interface

Section 10
Final review
Non-inclusive topic list

* Caches
* Exceptional control flow
* Processes
* Virtual memory
* Dynamic memory allocation
* Garbage collection
* Memory perils and pitfalls
* C versus Java
Caches

- What purpose do they serve?
- How do direct-mapped, set-associative, and fully-associative caches work?
- What are temporal and spatial locality and how do they affect evictions and miss rates?
- When do cold misses, conflict misses, and capacity misses occur?
Exceptional control flow

- Asynchronous exceptions
  - Interrupt signals such as SIGINT (caused by Ctrl+C)

- Synchronous exceptions
  - Traps (e.g. system calls such as open and read)
  - Faults (e.g. division by zero)
  - Aborts (e.g. memory error in hardware causes a crash)
Processes

A process is a running instance of a program
Each process has the illusion of exclusive use of the CPU and memory
How does the OS provide this illusion?

- How are `fork()`, `exec()`, and `wait()` used to spawn and manage processes?
- Bonus points (not really): What do children become if we don’t reap them?
Virtual memory

- What problem does virtual memory solve?
- How does virtual address to physical address translation work, and what are the components involved in the process?
- How does protection and sharing of pages between processes work?
Dynamic memory allocation

- How does dynamic memory allocation using an explicit free list work?
- How do `malloc` and `free` (`mm_malloc` and `mm_free` in lab 5 terms) interact with the heap?
- When does memory fragmentation occur?
Garbage collection

Why is garbage collection in C not an easy proposition?

At a high level, how does garbage collection in Java work?
Memory perils and pitfalls

Be able to identify:

- Bad (invalid) pointers
- Reads of uninitialized memory
- Double frees
- Memory leaks
Questions?

- Ask any exam- or lab-related questions
- Fill out course evaluations in last ten minutes