CS 367 (001)

Computer Systems and Programming

Instructor: Huzefa Rangwala

CS 367 @ GMU

Topics

- **what do I need to know?**
- **where is everything?**
- **how can I succeed?**
- **32,000 ft view**

where are things?

- slides and documents – www.piazza.com
- piazza – www.piazza.com
  - all files, schedule, announcements, discussions
- blackboard – mymason.gmu.edu
  - only for submitting projects, viewing grades
- textbook
  - Computer Systems: A Programmer's Perspective
  - 3rd edition; all examples moved to 64-bit!
  - Available on reserve at the Gateway Library (JC):
    QA76 . 5 . B795 2016.
- VSE account (Log in to Zeus)
  http://labs.vse.gmu.edu/index.php/FAQ/INeedAnAccountOrAPassword

Syllabus CheckList

- https://cs.gmu.edu/~hrangwal/cs367

- Questions ?

what do I need to know?

- This course strips away abstractions, and looks "under the hood"
- goals:
  - learn what actually happens when you load/run a program
  - learn about bits, memory, assembly, processes
  - find/eliminate bugs efficiently
  - prepare for systems courses (OS, compilers, architecture, networks)

how to get an A

- do all readings on time (before class), complete book exercises
- attend all lectures, participate
- try assignments early enough to ask questions. always turn in 100% working code
- go to office hours / piazza regularly with questions!
- study early and well for tests

how to get a B

- do pretty much all readings, work through book exercises
- only miss a couple meetings, catch up on what you missed
- start assignments earlier than a few days before deadline; get programs all working
- occasionally go to office hours / piazza as needed
- study quite hard just before tests

how to get a C

- only miss a couple meetings, but get distracted by the internet, shiny things, etc.
- do some readings, but not before class
- try hard on assignments, but a bit late or in one 'power session'. no time for clarifications/help.
- "I don't have time for office hours" (or, "I'll start at office hours and try to do all my learning there")
  - *office hours are to help you get unstuck, not to help you through the entire assignment. If you just bounce between all office hours to try to start/finish a project, you're punishing yourself and everyone else!*

how to get a D

- decide lecture isn't worth much time, try to just do work at home
- skip readings, unless you get really stuck on work
- attempt to start/finish assignments in one "power session" before deadlines, never get to ask questions.
- fail to keep up with the cumulative nature of the course, stumble on the same ideas each project
- realize about 2/3 into the semester that you're failing, and honestly try your best to salvage a C for the semester. (Hint: it's already too late)

how to Fail

- decide the first few lectures were easy, and stop coming to class
- skip the reading because it's not worth much of your grade/time
- start assignments too late to ever get clarifications/questions answered
- cram for tests the night before, if at all

Processor Design

- Write-Pair-Share Activity

- What are the fundamental blocks of a processor ?

What affects a program performance ?

You've (perhaps) been misled…

- **int ≠ integer**
  - 32-bit int: has a maximum value!
  - overflow: what happens when we store 10 pounds of int in a 5 pound sack?
    - 50000*50000 → **-1794967296** ????

- **float≠real**
  - float: like scientific notation, with a limited number of significant digits *and* a limited magnitude exponent.
  - we lose precision quite often, with big vs small operands.
  - (1e20 + -1e20) + 3.14 → **3.14**
  - 1e20 + (-1e20 + 3.14) → **0**

assembly code matters

- performance often requires tweaking low-level details, understanding the realities of memory
- security issues are usually assembly-level affairs
  - both fighting and creating malware
- system software
  - compiler target
  - OS manages process state

memory matters

- memory is finite
  - must manage competing needs between data, control, other programs…
- memory varies in speed/cost/access behavior
  - what is cache memory? If it's so good, why don't we have more of it?
  - ability to violate our assumptions leads to so many bugs!
  - why does reordering two loops cause a 21x slowdown??

Memory Performance Example

```
void copyij(int src[2048][2048],
int dst[2048][2048])
{
int i,j;
for (i= 0; i< 2048; i++)
  for (j = 0; j< 2048; j++)
       dst[i][j] = src[i][j];
}
```

```
void copyij(int src[2048][2048],
int dst[2048][2048])
{
int i,j;
for (j= 0; j< 2048; j++)
  for (i = 0; i< 2048;i++)
       dst[i][j] = src[i][j];
}
```

21 times slower
on Pentium 4

- Hierarchical memory organization
- Performance depends on access patterns
  - Including how we step through multi-dimensional array

Great Reality #4

- *There's more to performance than asymptotic complexity*

- Constant factors matter too!
- And even exact op count does not predict performance
  - Easily see 10:1 performance range depending on how code written
  - Must optimize at multiple levels: algorithm, data representations, procedures, and loops
- Must understand system to optimize performance
  - How programs compiled and executed
  - How to measure program performance and identify bottlenecks
  - How to improve performance without destroying code modularity and generality

course identity

- we focus on the programmer, not the tool creator.
  - CS440: build a small compiler (how do I use it?)
  - CS471: build a small OS (how do I use it?)
  - CS367: learn what these tools are and how to interact with them

prerequisites

- CS 262 (or CS222) C programming
- ECE 301 (or ECE331) digital electronics

- we write C code here; not much lecture time is spent re-covering the language.
- we write and run programs on zeus.vse.gmu.edu.
  - Don't expect to do all your work in a fancy GUI IDE
  - we work at the systems level with systems tools
  - grading is performed on zeus

some topics we'll cover

- data representation
  - binary representations, binary math
  - data structures at the byte level
- program representation
  - assembly
  - function calling conventions
- object files, linking
- memory
  - virtual memory, address translation
  - caches, TLBs
- processes and threads