# CSCI 310 — Computer Organization

Jeremy Iverson

College of Saint Benedict & Saint John's University

# logistics

instructor

Jeremy Iverson (`jiverson002@csbsju.edu`)
MAIN 258, (320) 363-5542
office hours: MTThF 12–1pm

textbook

*Computer Systems*, 5th Edition, Warford

website

`https://csbsju.instructure.com/courses/16318`



- I prefer to be called Jeremy
- Encourage questions right away
- Emphasize the importance of the Canvas site for finding information about the class
- office hours

  Mention outlook calendar & my home page

  For those unfamiliar with Outlook meetings, then they should schedule another way and we will go over this in meeting
- go through Canvas page organization quickly

# objectives

identify and describe the key components of a computer system and how the components interact

explain the layered architecture of computer systems and how each layer relates to the others

explain how data and programs are represented inside computers

explain the role of operating systems in managing storage, processes and programs

# evaluation

seven (7) assignments

- mix of worked problems and programming
- do your own work
- use each other as resources only
- due dates are strict (no partial credit for late assignments)

three (3) topical exams and one (1) cumulative

- closed book / open note

point distribution

assignments: 32% total
exams: 14% each
final: 26%



Should be student's own work, may work together but must individual solutions

Want to encourage using each other as resources, but do not want some students to rely on other students for answers

need more info...

see `course syllabus`!

# a word to the wise

computers $\not\equiv$ magic

programming $\not\equiv$ magic

# a word to the wise

computers ≠ magic

programming ≠ magic

banned phrases

“it worked when I tried it earlier”

“I don’t know, it just works”

“but it gives the right answer”



this class is not about your amazing ability to program, it is about your ability to reason about how the computer understands what you are saying

# a dose of honesty

We are not learning new problem-solving abstractions.

Not directly anyway.

## a dose of honesty

We are not learning new problem-solving abstractions.

We are unpacking the abstractions that we depended on in previous courses — seeing how the expressiveness of our favorite programming languages is represented and carried out by a computer.

We are building a foundation for understanding how future abstractions can be understood by a computer.



Not directly anyway.

## a dose of honesty

We are not learning new problem-solving abstractions.

We are unpacking the abstractions that we depended on in previous courses — seeing how the expressiveness of our favorite programming languages is represented and carried out by a computer.

We are building a foundation for understanding how future abstractions can be understood by a computer.

This process is often tedious and mechanical.



Not directly anyway.

A computer is not a *creative* device. In fact, it is only superior to humans in its ability to carry out tedious operations at an incredible speed.

# a good question

why bother?

- better debugger
- better programmer
- better problem-solver



most of you will neither write nor maintain compilers nor will you program in assembly on a regular basis, so then why bother...

better debugger — improve tracing skill and have a better idea of how things you write are understood by the machine

better programmer — debugging is a skill that many “programmers” lack, but you will have improved this skill therefore, you will be a better programmer

better problem-solver — because of your new understanding of the organization of a computer system you will better understand its capabilities and limitations — you will also gain experience moving between levels of abstraction, thus improving your abstract thinking

# a look ahead

become proficient in C

internalize the von Neumann architecture

develop a mental model for program execution

get very comfortable with binary

we will spend approx. one (1) week on C in class; that is not enough time to master it the level necessary for this class

# a bit of parting advice

remember that this is a 300-level course

I expect you to do some learning on your own and come to class prepared to enhance / clarify / correct the understanding the you created. I want to help you explore your understanidng of the material, not necessarily help you gain a basic understanding of the material

# a bit of parting advice

remember that this is a 300-level course

if something is confusing, tell me

I expect you to do some learning on your own and come to class prepared to enhance / clarify / correct the understanding the you created. I want to help you explore your understanidng of the material, not necessarily help you gain a basic understanding of the material

otherwise, I am relying on my own experience to decide which material to emphasize and/or clarify

# teaching philosophy

Your job is to empower those you teach; when you do for them what they should be doing for themselves, you create dependency rather than empowerment.

It is easy to give in to the frustration that results from seeing amazing possibilities for the people you are teaching, and you want it more for them than they want it for themselves.

Don't give in to that frustration!

— Based on passage from "Resisting Happiness" by Matthew Kelly

I sincerely believe that all of you can master this material and I want more than anything else this semester, to help you do that!However, don't expect me to give you answers. And do get frusterated with me when I challenge you to discover the answers for yourself.

questions?

# activity

download this slide deck and follow the instructions on the next slide

if there is still time, talk about all of the material for this course being hosted on GitHub, and how they can access it