Welcome to Comp 411!

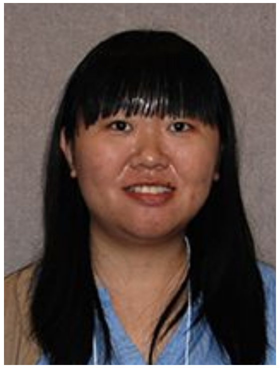
1. Course Mechanics
    a. What do I have to do to get an A in this course?
    b. Where are the course materials posted online, because I'm pretty sure that I am gonna sleep through a lot of these lectures?
    c. Fridays, is he serious?
2. Course Objectives
    a. How do computers work?
    b. Show me the binary?
    c. Some assembly required.
3. Course Changes

Whos

Lectures:

Leonard McMillan (SN 316)
Office Hours: M 2-4pm

TAs:

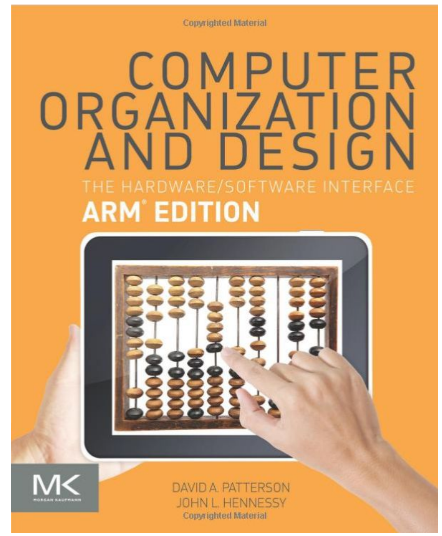


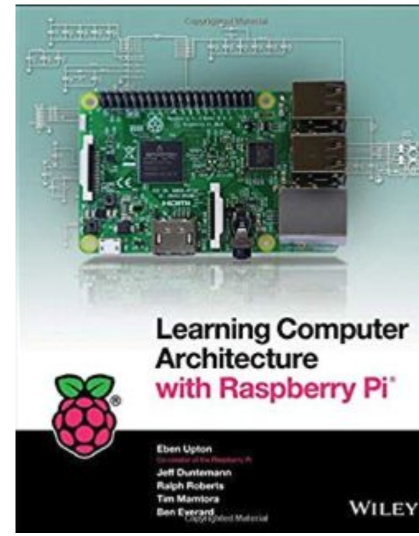
Rui Zhang &
Jacob Vosburgh
Office Hours: TBA

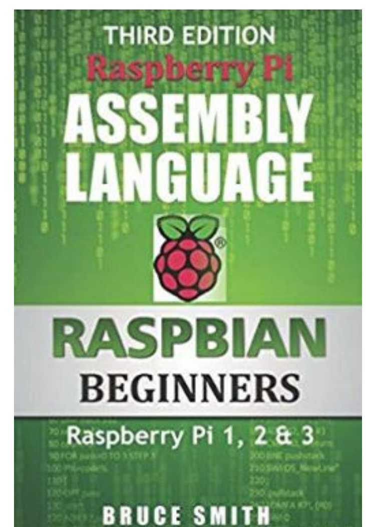
Whats

Book: None Required,
Supplemental Texts

- Will he follow any of these books?
  - Definitely not
- Are the problem set answers in the book?
  - Perhaps
- Why do I need them then?
  - In case you find yourself lost, need additional examples, or need a doorstop



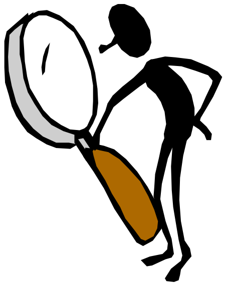




Course Mechanics

Grading:

Best 5 of 6 problem sets	25%
Best 9 of 10 laboratories	18%
2 in-class exams	32%
Final exam	25%



You will have at least two weeks to complete each problem set. Problem sets will be online. Late problem sets will not be accepted, but the lowest problem-set score will be dropped.

Friday Labs, starting next week (8/31), are mandatory, and will meet on most Fridays. Grade is based on completing a "lab checklist".

I will attempt to make Lecture Notes, Problem Sets, and other course materials available on the web before class on the day they are given.

Course Website

Logged in as: *mcmillan* Log out

Home | Research | Courses | Publications | Setup

Announcements

- **August 22, 2018:** The first class meeting in SN014 (☹summer is over).

Course Description

Comp 411, *Computer Organization*, explores the topic of how computers work, in terms of both software and hardware. It covers a wide range of topics including what a *bit* is, and why bits are the atoms in the universe of computation. We also discuss how information is represented and processed in hardware, and arrive to the conclusion that, to a computer, everything is data, including the instructions that underly software.

Comp 411 also covers the wide range of languages, and layers of translation, used for computation--spanning from machine language to assembly language to high-level compiled and interpreted languages. We will also touch on the conventions that will enable us to constuct large programs, modular software systems, and even programs that manage the loading, execution, and creation of other programs.

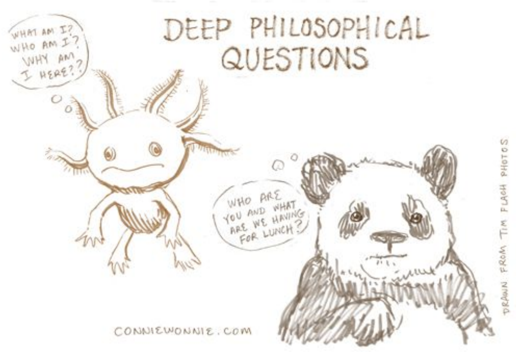
We will then delve deeper into computer hardware to discover what means to be *digital*. We will explore how simple combinational logic can be made to perform math and manipulate bits and how logic with state can be made to perform a series of operations. This will culminate in the virtual construction of a simple, yet fully functional computer.

http://csbio.unc.edu/mcmillan/index.py?run=Courses.Comp411F18

Goals of Comp411

To answer fundamental questions:

- What does a computer do with my program?
- How is data represented in a computer?
  - Numbers
  - Strings
  - Arrays
  - Photographs
  - Music



- How is a ***program*** represented in a computer?
- Are there limits to what a computer can do?

Goal 1: To Demystify Computers

Strangely, most people seem to be afraid of computers.

People only fear things they do not understand!

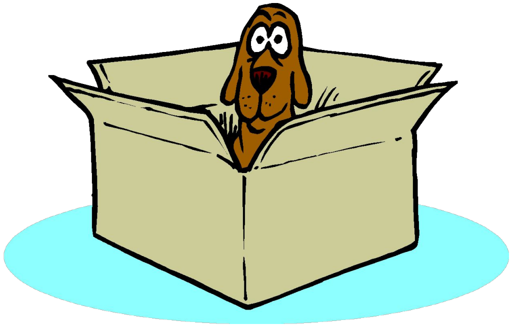
"I do not fear computers, I fear the lack of them."
- Isaac Asimov (1920 - 1992)

"Fear is the main source of superstition, and one of the main sources of cruelty. To conquer fear is the beginning of wisdom."
- Bertrand Russell (1872 - 1970)

"Nobody knows exactly what's going on because of computers!"
- Donald Trump

Goal 2: The Power of Abstraction

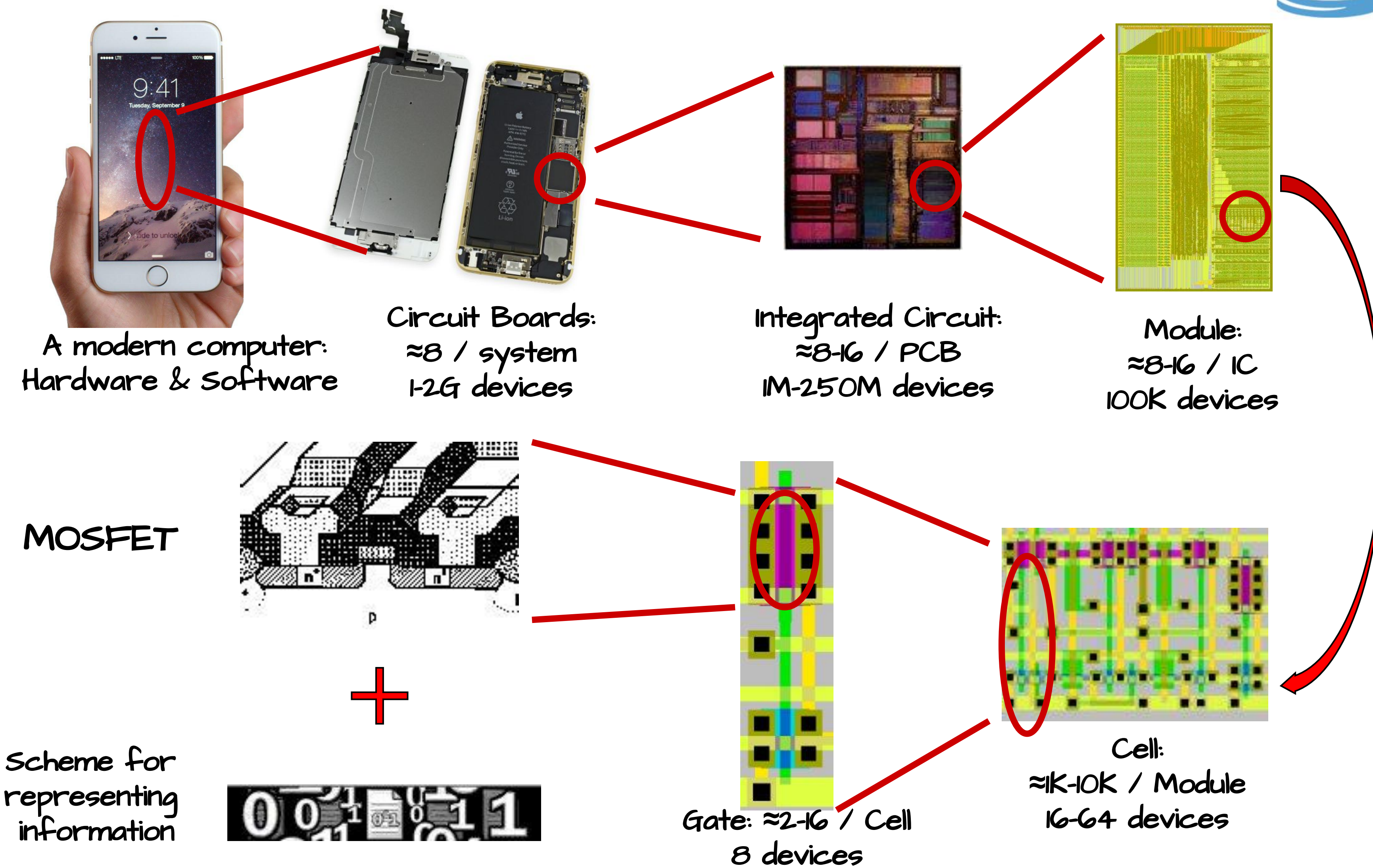
Define a function, develop a roust implementation, and then put a box around it.



Abstraction enables us to create unfathomable systems, including computer hardware and software.

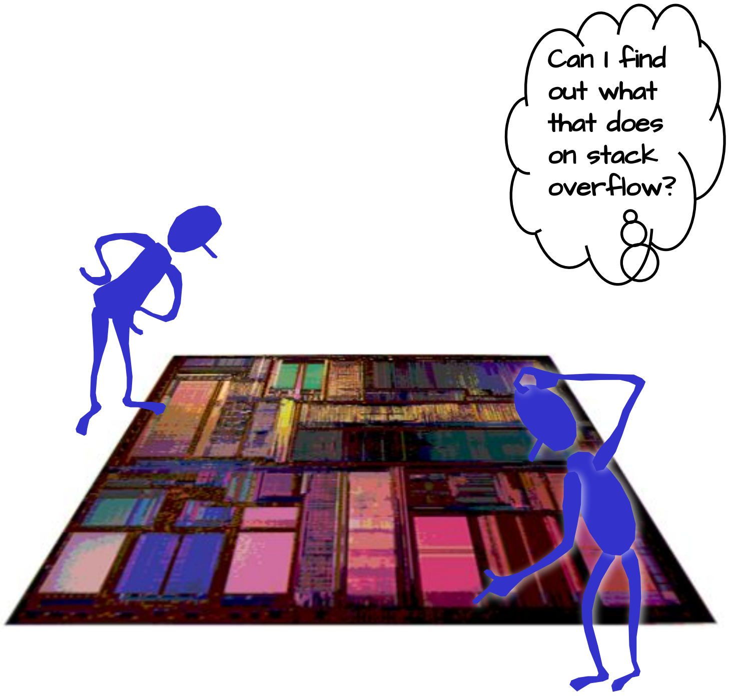
Why do we need ABSTRACTION? Imagine a billion...

Orchestrating systems with >1G components



What's in a computer?

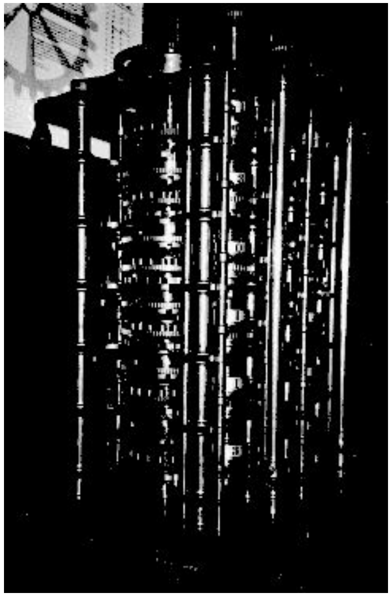
- Structure
    - Hierarchical design
    - Limited complexity at each level
    - Reusable building blocks
- Interfaces
    - Key element of system engineering typically outlives its implementation
    - Isolate design from technology, allows evolution
    - Major abstraction mechanism
- What makes a good system?
    - "Bang for the buck." Minimal mechanism, maximal function
    - Reliable, resilient, reusable
    - Accommodating future improvements

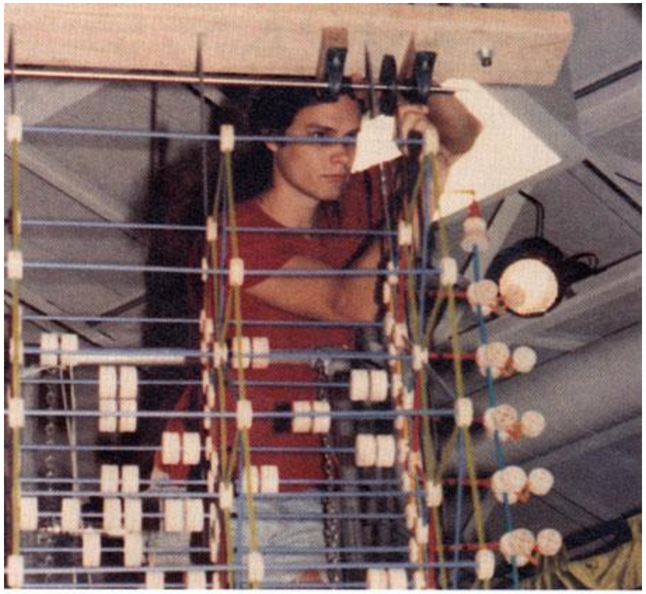


Computational Structures

What are the fundamental elements of computation?

Can we define computation independent of implementation or the technology that it is implemented with?





Edward Hardebeck helps to assemble the Tinkertoy computer

What do programs really do?

By now you should be able to look at a program specification and figure out what it does.

What does this do?

How would you figure it out?

Try f(36), f(64), f(100)

```
int f(int x) {
    int r;
    int odd = 1;
    for (r = 0; x >= odd; r++) {
        x -= odd;
        odd += 2;
    }
    return r;
}
```

How does a computer do it?

What does a computer do with this program specification?

```
int f(int x) {
    int r;
    int odd = 1;
    for (r = 0; x >= odd; r++) {
        x -= odd;
        odd += 2;
    }
    return r;
}
```

```
f:      mov     r1,r0
        mov     r2,#1
        mov     r0,#0
        b       test
loop:   sub     r1,r1,r2
        add     r2,r2,#2
        add     r0,r0,#1
test:   cmp     r1,r2
        bge     loop
        bx      lr
```

It translates it to a series of simple instructions…

Are there limits to computation?

- Will some new instruction be invented that fundamentally change how fast computers solve problems?
- Can computers solve ***any*** well specified problem?
- Can we predict how long it will take for a computer to solve a given problem?
- Does there exist a new model of computation?

A Program emulating a Computer

```
int memory[16384];    // for instructions and data
int register[32];     // for variables
int pc;               // next instruction to execute
int flags;            // persistent state

void main(void) {
    pc = 0;
    while (1) {
        instruction = memory[pc];
        pc = pc + 1;
        flags = execute(instruction);
    }
}
```

A computer is just an interpreter that executes simple program loop

Where are we going?

- How is data represented, stored, and manipulated in a computer?
- What basic operations does a computer use?
- What does mean to "compute"?
- Are there limits to what can be computed?
- Why are computers so fast?
- What am I asking a computer to do when I give it a program to execute?
- How are programs translated into computer instructions?
- Why are some programs faster than others that perform the same function?

Summary

- 411 answers the following questions:
  - How is information represented, stored, and manipulated by a computer?
  - What does a computer really do with my program?
  - How do you design, build, and manage large systems?
- 411 logistics
  - M, W in general are lectures and discussions
  - F ~2 hr labs starting 9/7 (We'll have lectures on 8/24 & 8/31)