1. Introduction
    a. About me
        i. First time being an instructor of record for a class
        ii. If you have any comments or constructive feedback, let me know!
    b. About this class (why computer architecture?)
        i. This is a required class for both CS and CSE
        ii. “Core material” for most CS graduate programs
        iii. My goal is to get you to remember some of this knowledge to use in industry or your next computer architecture class
        iv. My *hope* is that I get some of you interested in computer architecture

2. About the class itself
    a. Discussions on second half of Wednesdays by our TA, Yuan
    b. Three quizzes (20%), one midterm (20%), and one final (30%)
        i. All of these will be on Thursdays, which are in a different room!
        ii. First quiz on Thursday, will get back to you before the drop deadline
    c. Four to five labs (30%, still debating on number)
        i. Mostly in Logisim, one might be in C++
        ii. First assignment will get you used to Logisim
        iii. Can turn in up to 48 hours late for non-linear penalty
    d. Regrades
        i. One week from return of assignment
        ii. Office hours only
    e. Websites (there’s a lot of them)
        i. Github is where all the course materials get posted
        ii. Canvas is for submitting assignments
        iii. Gradescope is for returning your tests
        iv. Piazza is for discussion
    f. Textbook
        i. Reading schedule on Github
        ii. *Digital McLogic Design* provided on Canvas for some of the digital design stuff
        iii. Otherwise, *Computer Organization and Design* for everything else
            1. 9^{th} or 10^{th} edition is fine

3. Computer architecture
    a. Abstraction layers
        i. Software
            1. Applications, OS
            2. Could divide the OS further into file system, drivers, kernels…
        ii. Hardware
            1. Hardware devices, gates, wires, transistors, electrons
            2. Again, could subdivide further
        iii. Computer architecture
            1. The “hardware-software interface” according to Hennessey and Patterson of Stanford and UCB

- iv. Computer architecture is a broad topic
  1. You're not going to know everything after this class
  2. Get a general overview
  3. Some of the material could be useful in industry, say, a SWE position
     a. Example: designing an application's working set to fit in L2 cache
- v. What we're going to talk about
  1. Digital design – basic building blocks
  2. Design parts of a CPU and memory
  3. Learn about busses and about memory
  4. Do this in a different order than the book
     a. Memory is important
     b. Bussing is also important, but doesn't lend itself to good homework

b. Definitions
- i. Computer architecture
  1. Attributes of a system visible to the programmer
  2. Those which impact the logical execution of a program
- ii. Computer organization
  1. Operations, units
  2. Their interconnections that realize the architectural specifications

c. Tasks of a computer
- i. Transfer data between external devices
  1. Keyboard to monitor
  2. Microphone to speaker, so on
- ii. Storage device
  1. Network to memory
- iii. Data processing
  1. Internal or external source and destination
- iv. Control external devices

d. Parts of a computer
- i. I/O
  1. Mouse, keyboard main examples we think of
  2. Other peripherals count too (like speakers)
- ii. Main memory: RAM, caches
- iii. System bus
- iv. CPU
  1. Registers, store values
  2. ALU, perform operations
  3. Internal bus, transfer data
  4. Control unit
     a. Sequencing logic, where to go next
     b. Control unit registers, decoders
     c. Control memory

4. History of computing

a. Mechanical
- i. 1801 – Joseph-Marie Jacquard's loom
- ii. 1842 – Charles Babbage
  1. Difference engine to compute polynomials
  2. Analytical engine, like modern machines

- b. Electromechanical
  - i. 1936 – Alan Turing "universal computing machine"
  - ii. Data and program on single tape
- c. Electronic
  - i. First generation – vacuum tubes
    1. 1945 – John Von Neumann working on the Manhattan Project
       - a. With team, designs architecture that is used by nearly every machine today
       - b. Von Neumann architecture – get to this later
    2. 1946 – Electronic Numerical Integrator Analyzer and Computer, or ENIAC
       - a. Used to calculate trajectories for bombing runs
       - b. Not very good – 20% made it within 1000'
       - c. Huge, 80' long and 8.5' high
       - d. Difficult to program, using patch boards
  - ii. Second generation – transistors
    1. 1959 – memory via small ferro-magnetic donuts with wires running through them
    2. Source of the term "core memory"
    3. 1K memory was the size of a shoe box
  - iii. Third generation – planar transistors
    1. 1964 – transistors on pieces of silicon
    2. Printed circuit boards (PCBs)
  - iv. Fourth generation – CPU on one chip
    1. 1971 – use very-large-scale-integration (VLSI) to accomplish
  - v. Fifth generation – now(?)
    1. Lines are fuzzy
    2. Standard chips, programmable logic devices (PLAs)
    3. Field programmable gate array (FPGAs)
    4. Application specific integrated circuits (ASICs)
       - a. Today, commonly heard with crypto mining

5. Designing for performance
   - a. Increase CPU performance
     - i. Increase clock frequency
     - ii. Increase size and speed of caches on CPU
     - iii. More parallelism
       1. Pipelining and other instruction-level parallelism techniques
       2. Branch prediction
       3. Speculative execution
     - iv. Improve interface between RAM and CPU
   - b. Power wall
     - i. Want to increase clock frequency
     - ii. $P = cfv^2$ (p = power, c = capacitance, f = frequency, v = voltage)
     - iii. Lower voltage to increase frequency, effective because of the squared term
     - iv. Need a certain voltage to differentiate 0 and 1, we're at that point
     - v. Can't increase power further without burning chips
     - vi. Hit this power wall around 2006
   - c. Gordon Moore's law, 1965
     - i. Every 18 months, number of transistors on a chip will double
     - ii. Still holding (roughly), but dark silicon problem
     - iii. Can't power all the transistors at once without burning the chip

d. (Robert) Dennard scaling, 1974
   i. As transistors get smaller, power density stays constant
   ii. Make a transistor smaller, uses less power
   iii. Broke down around 2006