

# CSCI 210: Computer Architecture

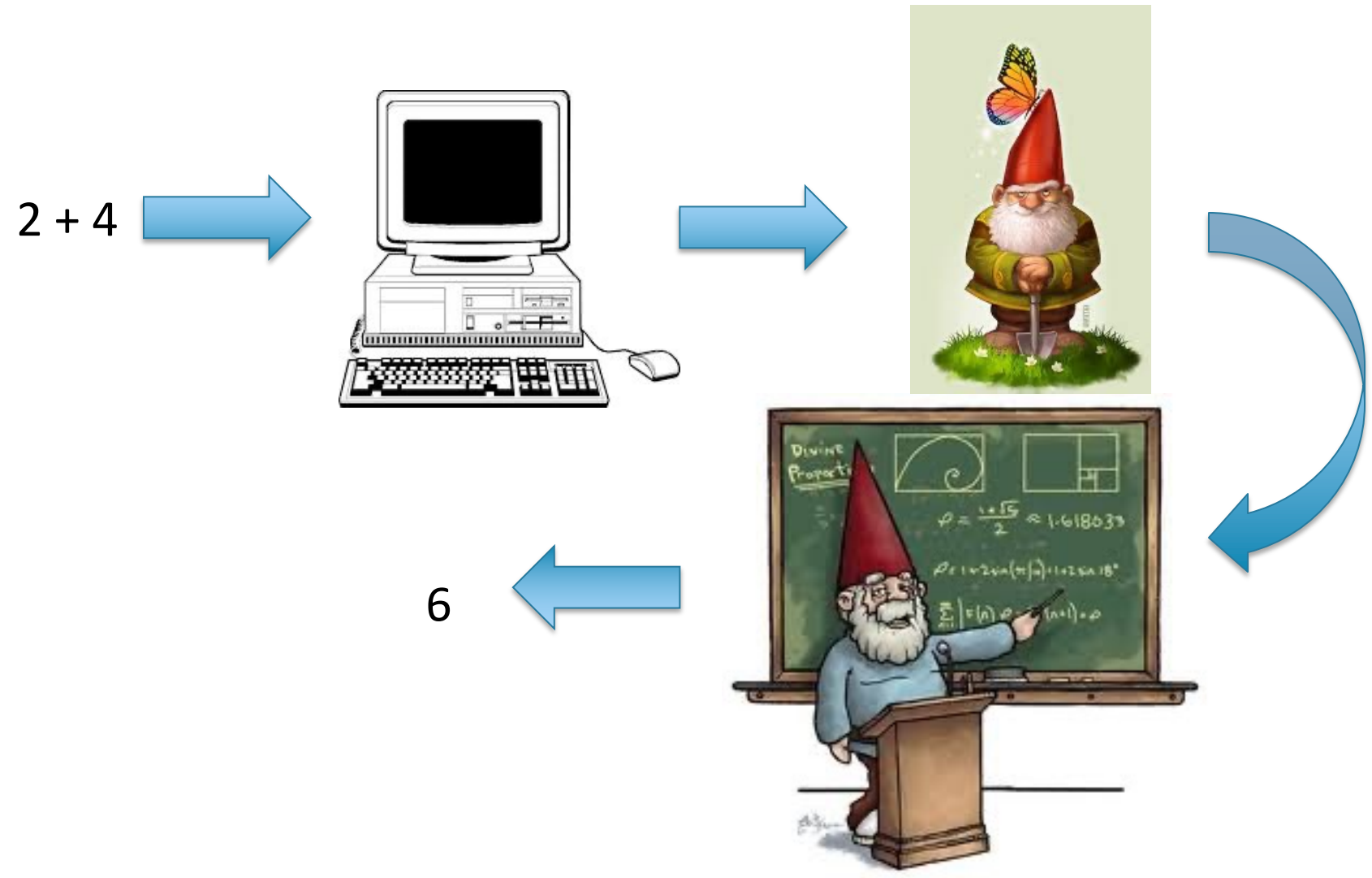
## Lecture 1: Introduction

Stephen Checkoway

Oberlin College

Slides from Cynthia Taylor

# Previous Conceptions of How Computers Work



# What is a computer?



# What is a computer?



# What is a computer?



By IFCAR - Own work, Public Domain,  
<https://commons.wikimedia.org/w/index.php?curid=17238574>

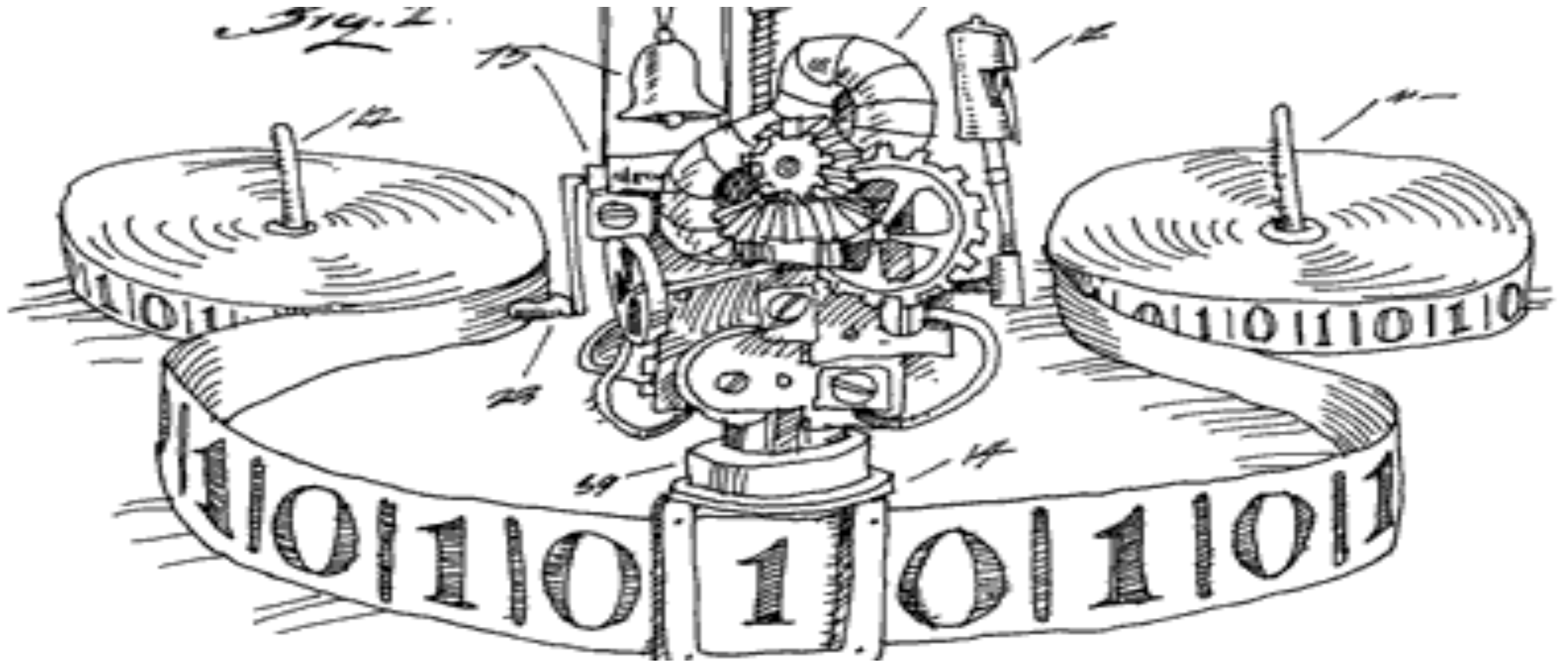


By Kskhh - Own work, CC BY-SA 4.0,  
<https://commons.wikimedia.org/w/index.php?curid=84103399>

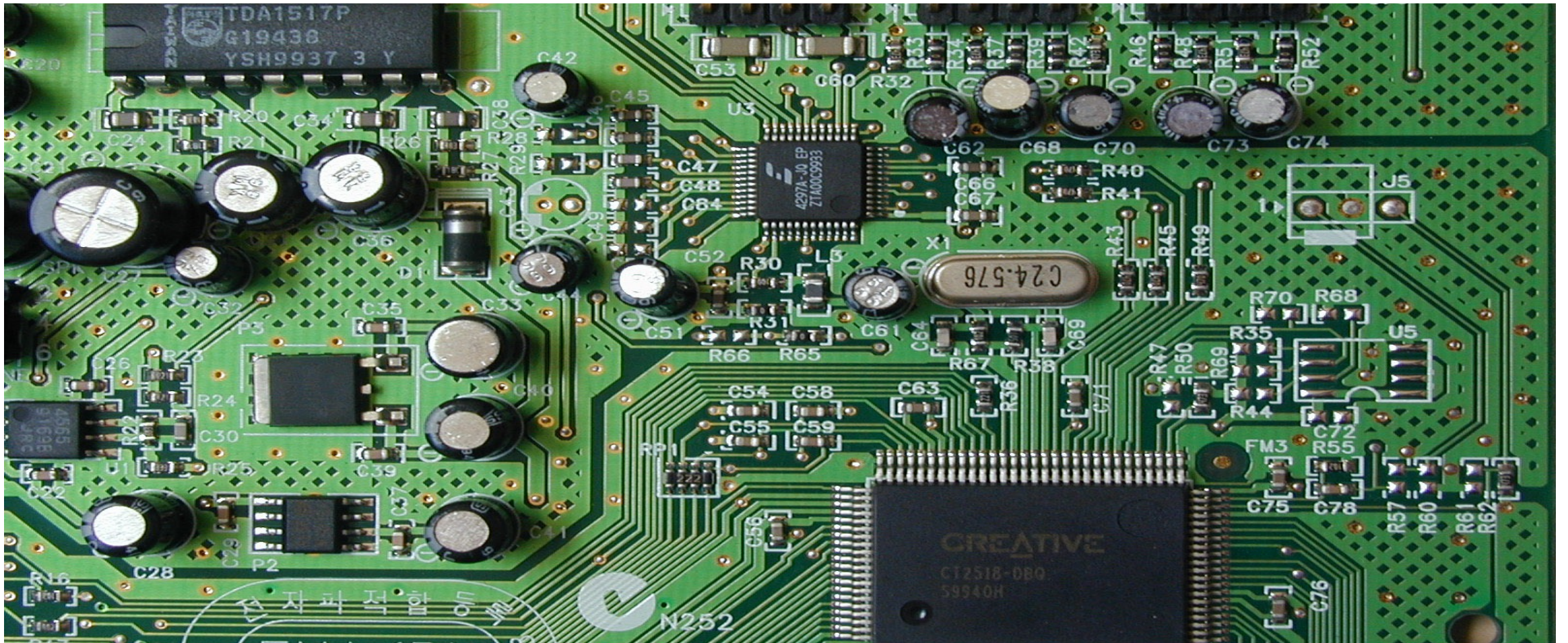


By Tmthetom - Own work, CC BY-SA 4.0,  
<https://commons.wikimedia.org/w/index.php?curid=40636987>

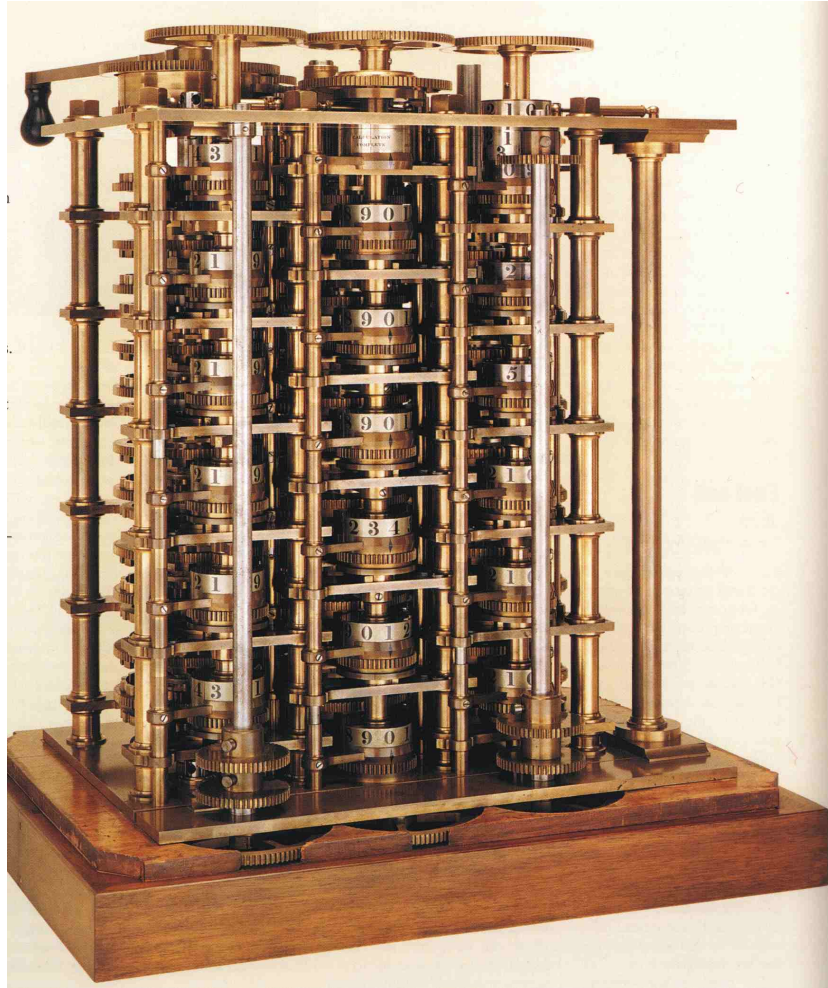
# What is a computer?



# What is a computer?



# Babbage's Difference Engine





# Computer Science History: Ada Lovelace

## YOUNG ADA LOVELACE



- Daughter of Lord Byron
- Wrote programs for the theoretical Analytical Engine
- Invented the idea of loops

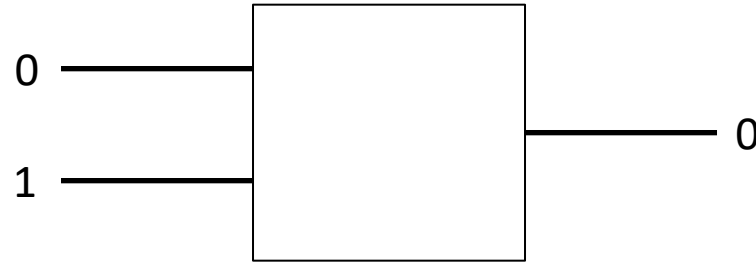
Comic by Kate Beaton, <http://www.harkavagrant.com/index.php?id=298>

# What is a computer?



- A device that reliably combines a given set of inputs to create the same output

# But how does it Python?



```
def main():  
    n = eval(input( "How many numbers should I sum?: "))  
  
    sum = 0  
    for i in range(1,n+1):  
        .. sum = sum + i  
    print("The sum of the first", i, "positive integers is", sum)  
  
main()
```

# Answer: Abstractions!

- We build high-level complex things by abstracting away the low-level complicated details
- We design computer systems in terms of these abstractions
- Abstractions are similar to metaphors (e.g., “display this text file in a *window*”)
- This works by building progressively more low-level abstractions
  - It takes a lot of work to display text in a window; one step along the way is “draw a line on the screen” which itself needs “color a pixel on the screen”

# Discuss with your neighbors

- Introduce yourselves
- What are some different metaphors we use in computers?
  - Desktop, bugs, ??

# Metaphors that Computers Use

# Levels of Abstraction

- User Interfaces
- High Level Languages
- Assembly Language
- Instruction Set Architectures
- Physical chip

# In This Class

- What are the fundamentals we build these abstractions on top of?
- How do we create these abstractions?



# Who am I? Professor Stephen Checkoway

- Research: Computer security
- Fun Facts:
  - I'm face blind
  - I have two cats



# Class will be graded based on:

- Labs — Programming assignments
- Problem Sets — Written assignments
- Reading Exercises — Interactive online textbook
- Class participation — Clicker questions!
- Final project – simulating memory and running experiments

# Labs

- Programming assignments designed to explore the architecture concepts we learn in class
  - Java, MIPS, logic gates

# Problem Sets

- Written assignments where you solve problems related to computer architecture
- Examples:
  - Converting decimal numbers to binary or hex
  - Simple assembly language programs
  - Drawing circuit diagrams
  - Answering questions about the CPU

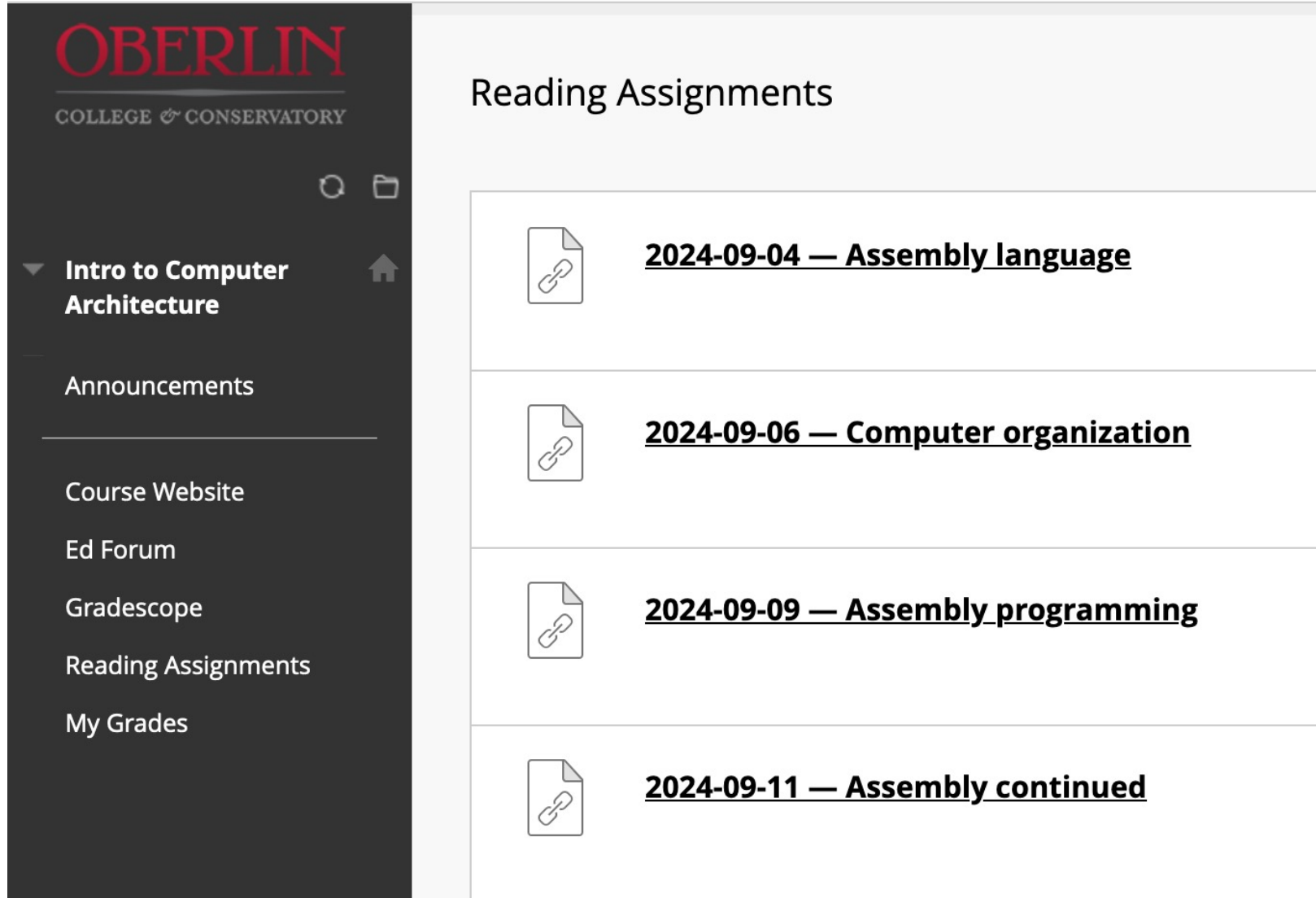
# Problem Sets

- Can be resubmitted within 1 week of receiving your grade
- Final problem set grade is 25% your original submission grade, 75% your new grade.
- Problem Set 0 due Friday, September 6 at 23:59

# Reading

- We will be using a zyBook
- Can buy directly from zyBook, or buy an access code from the bookstore
  - Click any zyBooks assignment link in Blackboard (Do not go to the zyBooks website and create a new account)
  - Subscribe
- Financial aid:
  - Email Michele Kosboth <mkosboth@oberlin.edu> in Financial Aid
  - Let me know if that doesn't work
  - zyBooks will let you use the book for free for the first month

# Reading on BlackBoard



The screenshot displays the BlackBoard interface for the course 'Intro to Computer Architecture'. On the left is a dark navigation sidebar with the Oberlin College & Conservatory logo at the top. Below the logo are icons for a refresh button and a folder icon. The sidebar menu includes: 'Intro to Computer Architecture' (with a dropdown arrow and a home icon), 'Announcements', 'Course Website', 'Ed Forum', 'Gradescope', 'Reading Assignments', and 'My Grades'. The main content area is titled 'Reading Assignments' and contains a list of four items, each with a document icon and a link symbol:

- [2024-09-04 — Assembly language](#)
- [2024-09-06 — Computer organization](#)
- [2024-09-09 — Assembly programming](#)
- [2024-09-11 — Assembly continued](#)

# Reading

- Due **BEFORE CLASS** on the day it is listed on the class schedule
- All readings linked from BlackBoard
- **FIRST READING DUE WEDNESDAY**



# Clickers!



- Lets you vote on multiple choice questions in real time.
- Like pub trivia, except the subject is always computer architecture.
- You need one by Wednesday

# iClicker Accounts

- You must create an iClicker account to ensure your grades are counted:
- Visit [iClicker.com](https://iclicker.com) > Create an Account > Student.
- Or, download the iClicker Student iOS/Android app. Select Sign Up! to create your account.
- If you already have an iClicker account, just sign in! Do not create and use more than one iClicker account as you will only receive credit from a single account.
- If you have a physical iclicker, you do not need to pay anything for the account – just enter the iclicker id

# Masking Policy

- I'm going to be wearing a mask
- You may wear a mask at any point for any reason
- If you're feeling unwell or recovering from illness, please mask
- I will have masks available if you would like one

# Questions?

# How This (and every) Class Works

- I try to create an optimal situation for you to learn the material
- You and your classmates work together to construct new knowledge
- Our goal as a class is to support each other in learning

# Group Discussion Norms

- Make sure everyone gets to talk.
- Have everyone state their answer before discussing which answer is correct.
- Take turns reporting out.
- If you think someone is wrong, ask them to explain their thinking rather than just dismissing it.

# Class Norms

- Contribute as you feel comfortable
  - If you're not comfortable answering, you can pass.
  - If you're not usually inclined to speak much in class, push yourself to ask questions more often.
- Be aware of the space you take up in class
  - Make space for others, use some space for yourself
- The main goal of every person in the class should be to engage proactively with the ideas we understand the least. If someone asks a question/makes a comment that seems obvious to you, show them respect.

# Collaboration Policy

- Discuss the labs/problem sets with anyone
- Post questions on Ed
  
- Don't show anyone your code
- If you work through how to solve a problem, please change relevant numbers from the assigned problems
- Must write down answers separately



# Questions?

# The Challenge of Computer Architecture

- The industry changes faster than any other.
- The ground rules change every year.
  - new problems
  - new opportunities
  - different tradeoffs
- It's all about making programs run faster or use less energy or provide more features than the other company's machine.

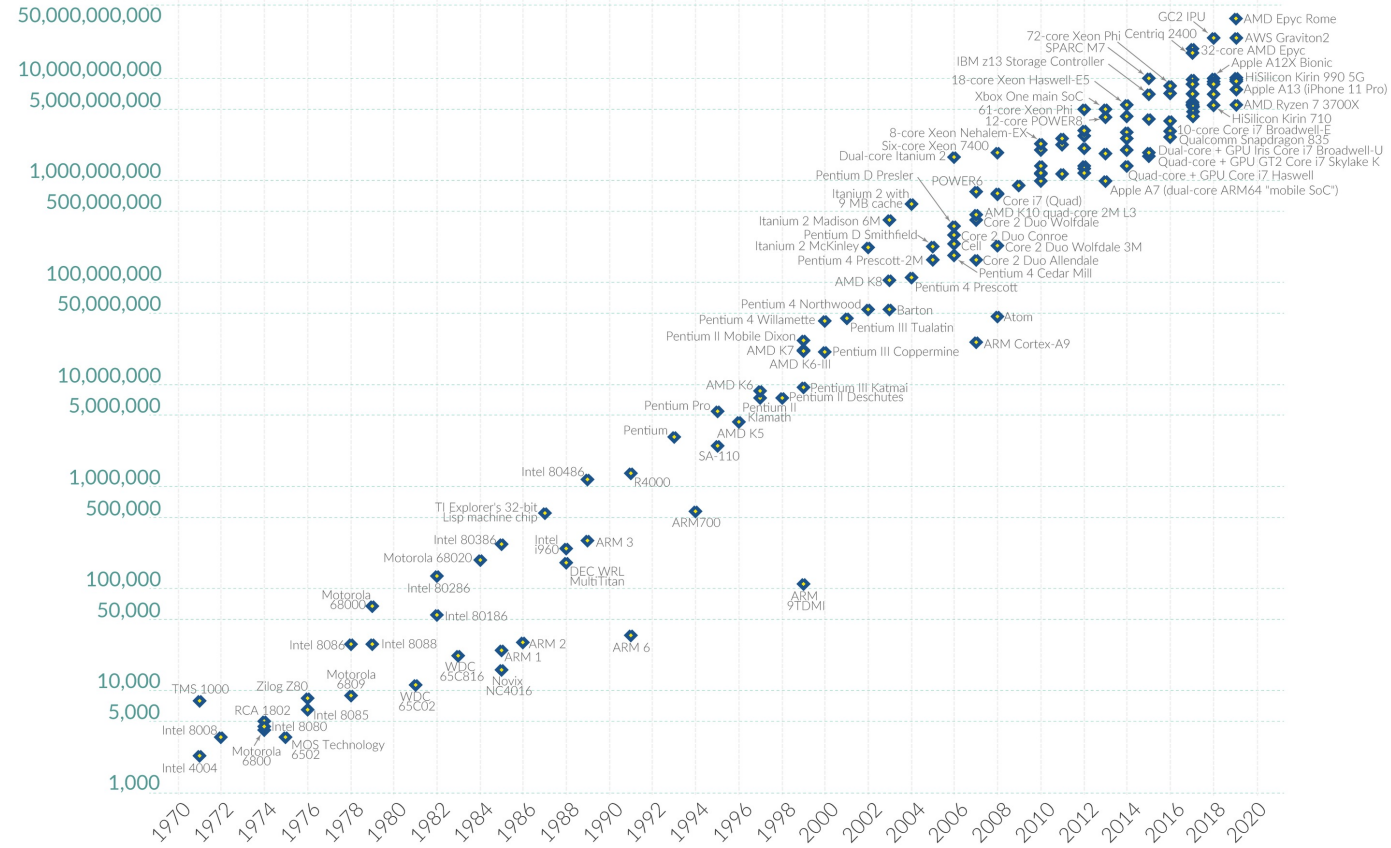
# Moore's Law

Moore's Law: The number of transistors on microchips doubles every two years

Our World in Data

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.

## Transistor count

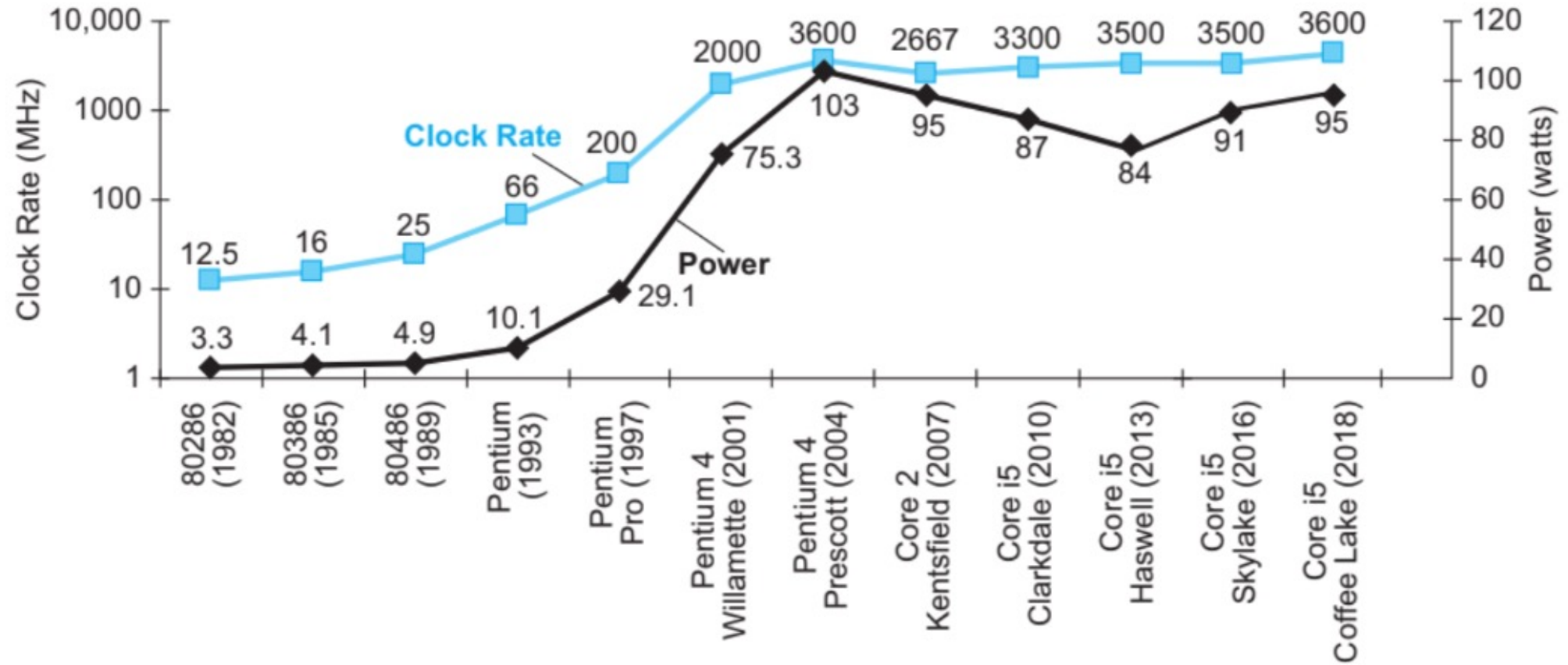


Data source: Wikipedia ([wikipedia.org/wiki/Transistor\\_count](https://wikipedia.org/wiki/Transistor_count))

OurWorldinData.org – Research and data to make progress against the world's largest problems.

Licensed under CC-BY by the authors Hannah Ritchie and Max Roser.

# Clock rate and Power with Time

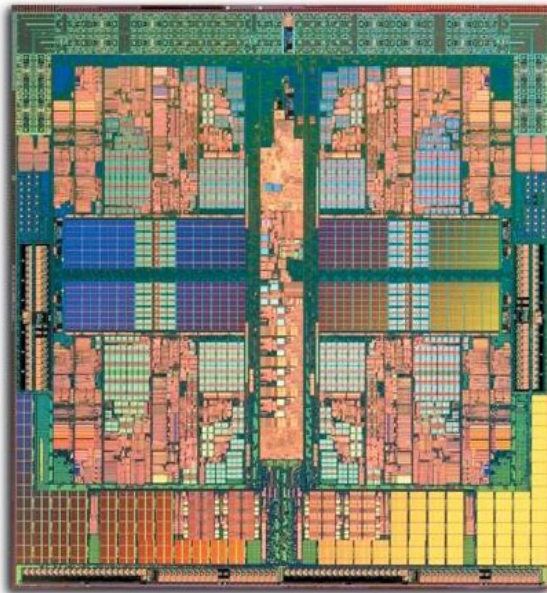


# Cooling

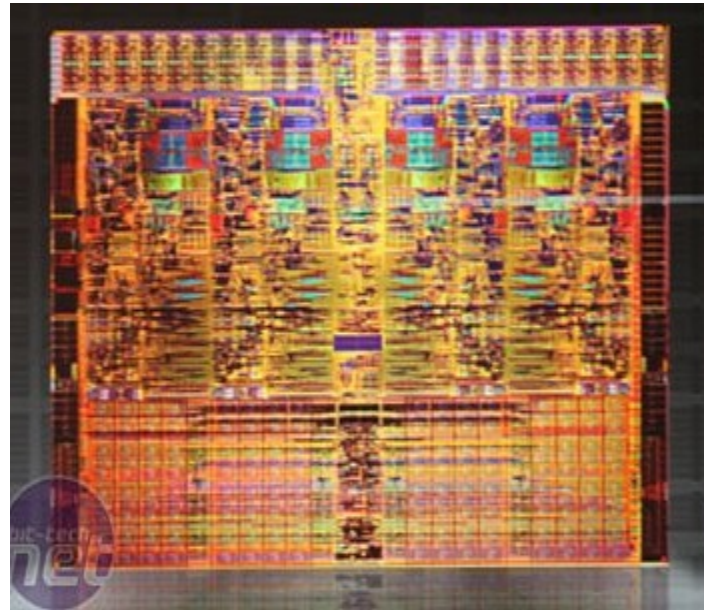
- CPUs get HOT
  - Switching those little transistors on and off takes power!
  - Power turns into heat.
- If they get too hot, they will burn out
- Can no longer efficiently cool number of transistors on a chip

# All is not lost

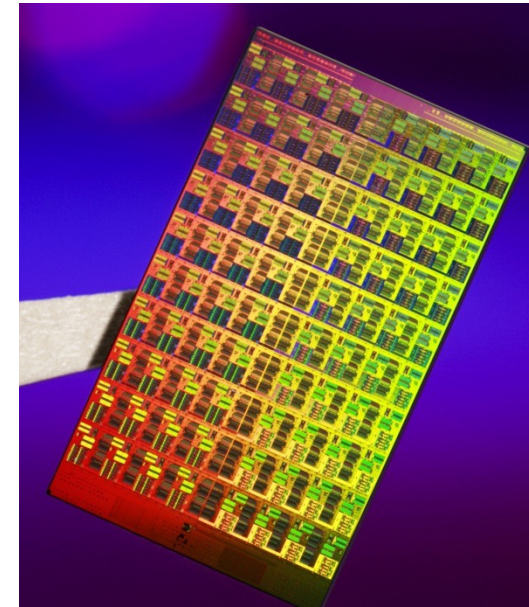
- If we can't run a single instruction faster, what if we run a bunch of instructions at once?



Intel Quad Core



Intel Nehalem



Intel: 80-core prototype

# Problems we (YOU) have to deal with when writing parallel code

- Only works for tasks that aren't overly sequential.
- Have to be able to balance what tasks are running on all cores.
- Have to deal with overhead of scheduling and communication.

If one process can run a program at a rate of  $X$  per second, how quickly can two processes run a program?

- A. Slower than one process ( $<X$ )
- B. The same speed ( $X$ )
- C. Faster than one process, but not double ( $X-2X$ )
- D. Twice as fast ( $2X$ )
- E. More than twice as fast ( $>2X$ )



# Code PERFORMANCE is dictated by COMPUTER ARCHITECTURE

- The goal of this class is NOT to make you into electrical engineers.
- The goal of this class is to expose you to the world of computer design:
  - Using a historical perspective will lead us from simpler to more complex designs
  - Will help you understand what in a design leads to better performance.

# Understanding Computer Architecture Will Let You

- Write better code
- Write faster code
- Understand what is and isn't possible

# You need computer architecture if you're a:

- Hardware designer
- Embedded systems programmer
- Computer Security professional
- Video Game Speedrunner

# Reading

- Next lecture: Assembly Language
  - Remember, click the assignment link from BlackBoard to subscribe to the book to get access
  - Read zyBook Section 1.3 & 1.4
- Problem Set 0 due Friday