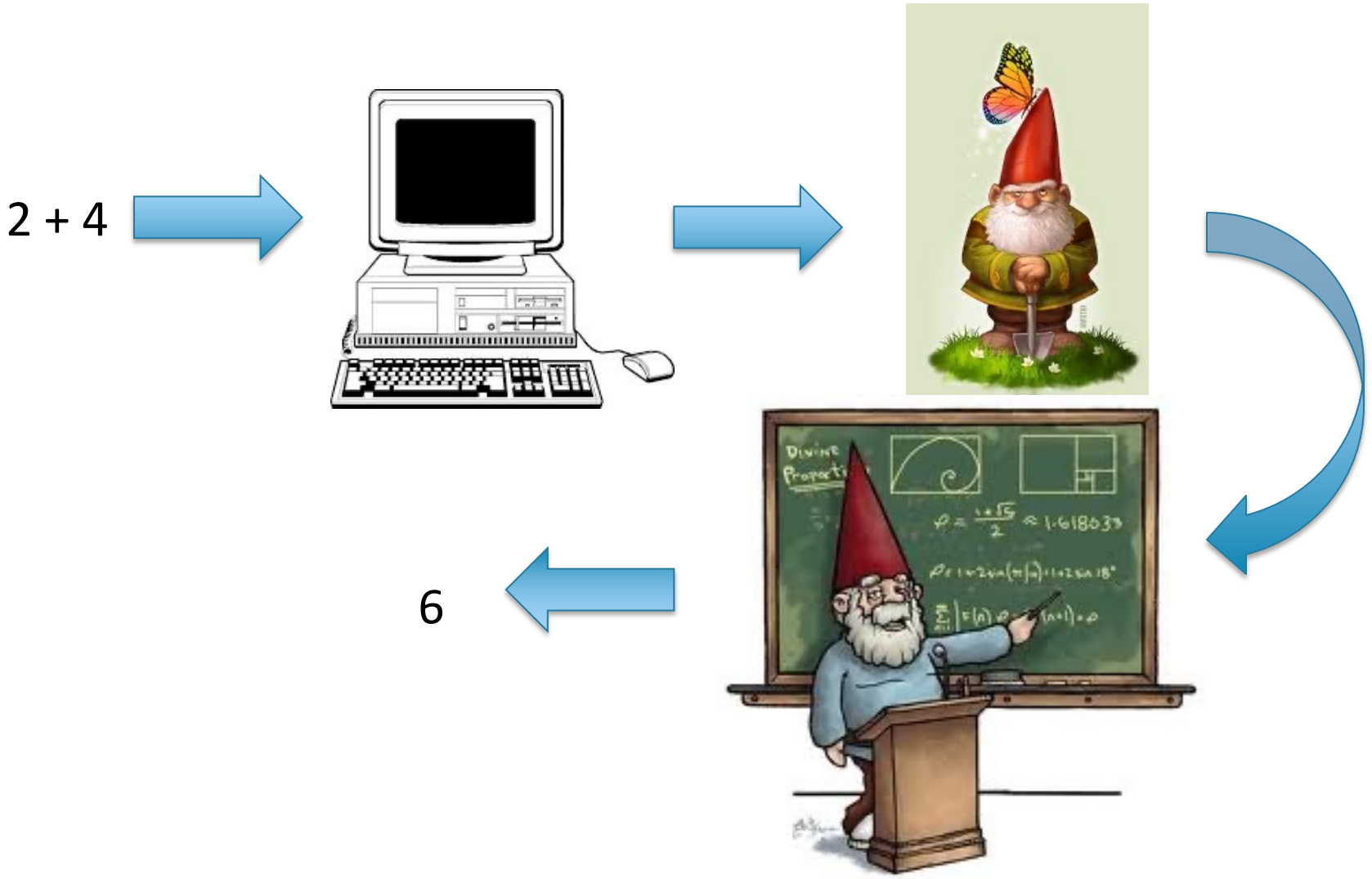
CSCI 210: Computer Architecture
Lecture 1: Introduction

Stephen Checkoway
Oberlin College
Slides from Cynthia Taylor

Previous Conceptions of How Computers Work



What is a computer?

Is this a computer?

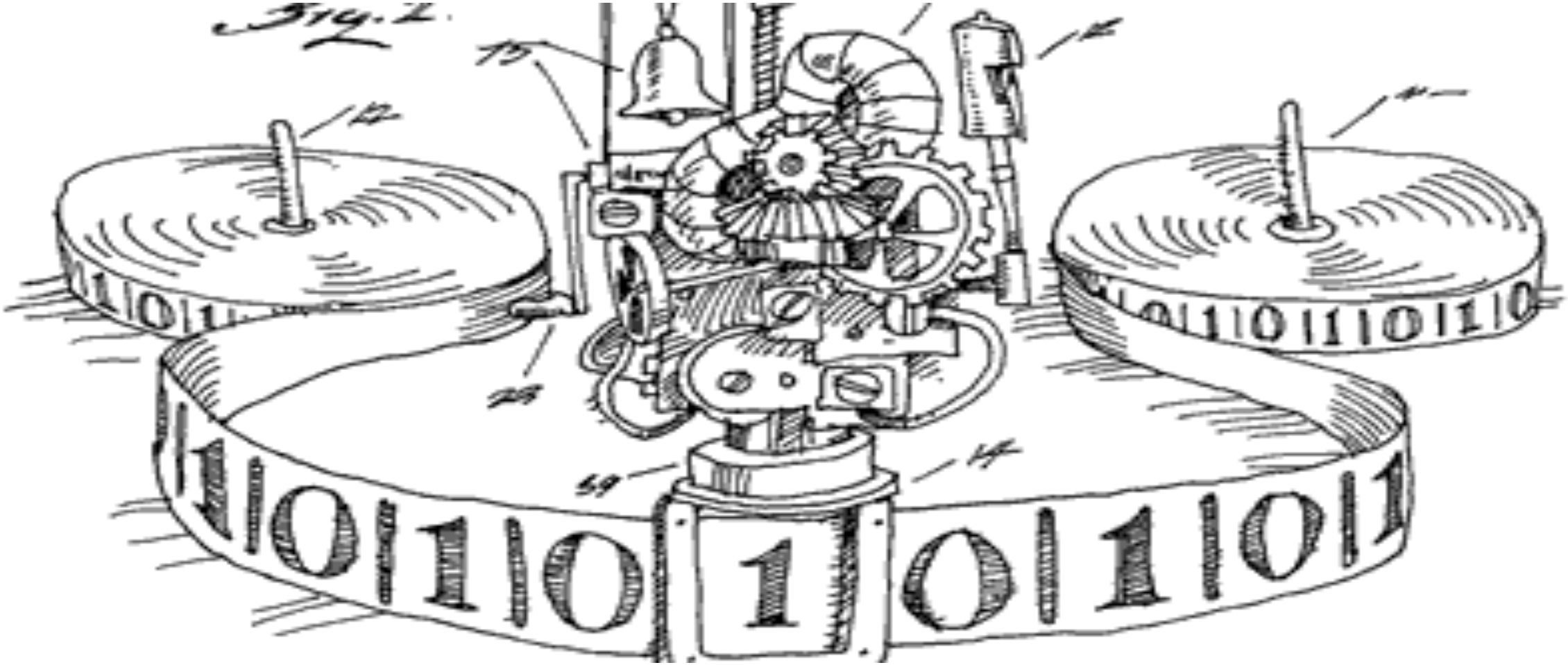
Are these computers?

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

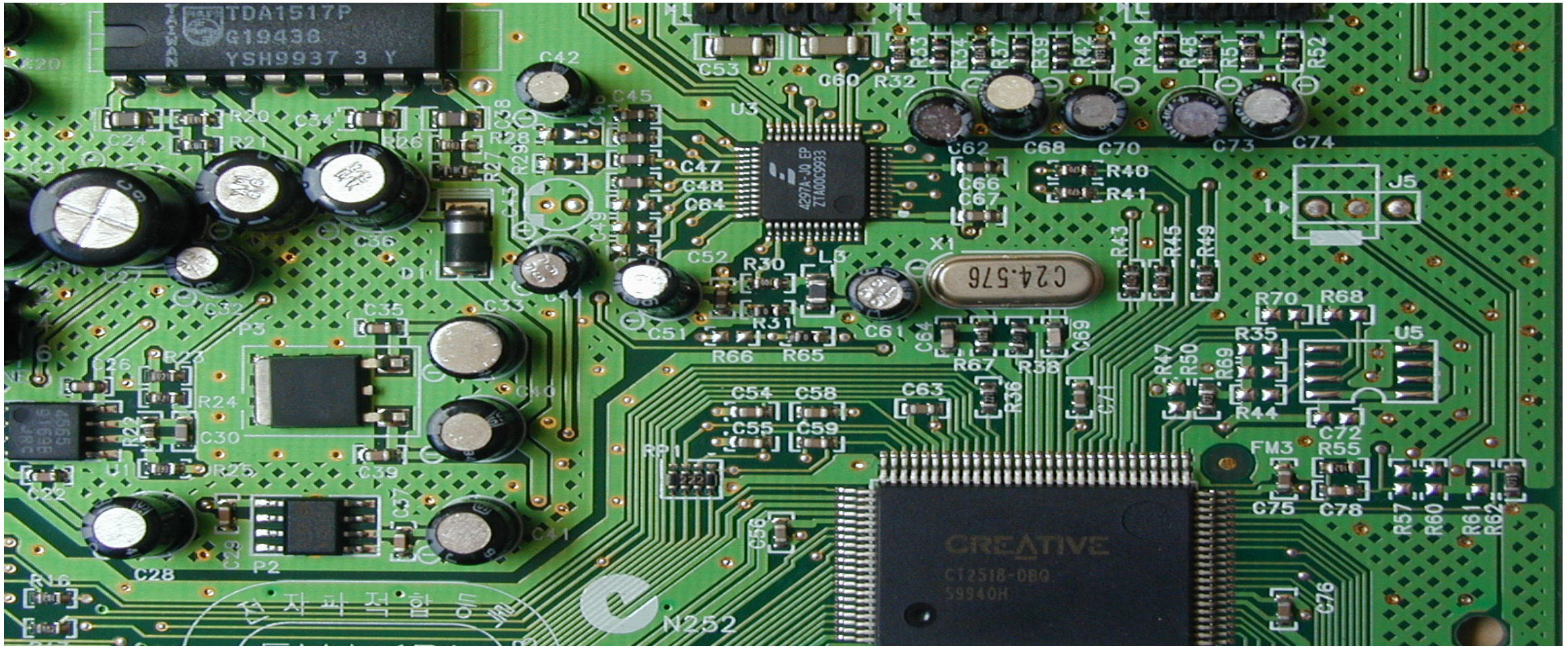
By Kskhh - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=84103399

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

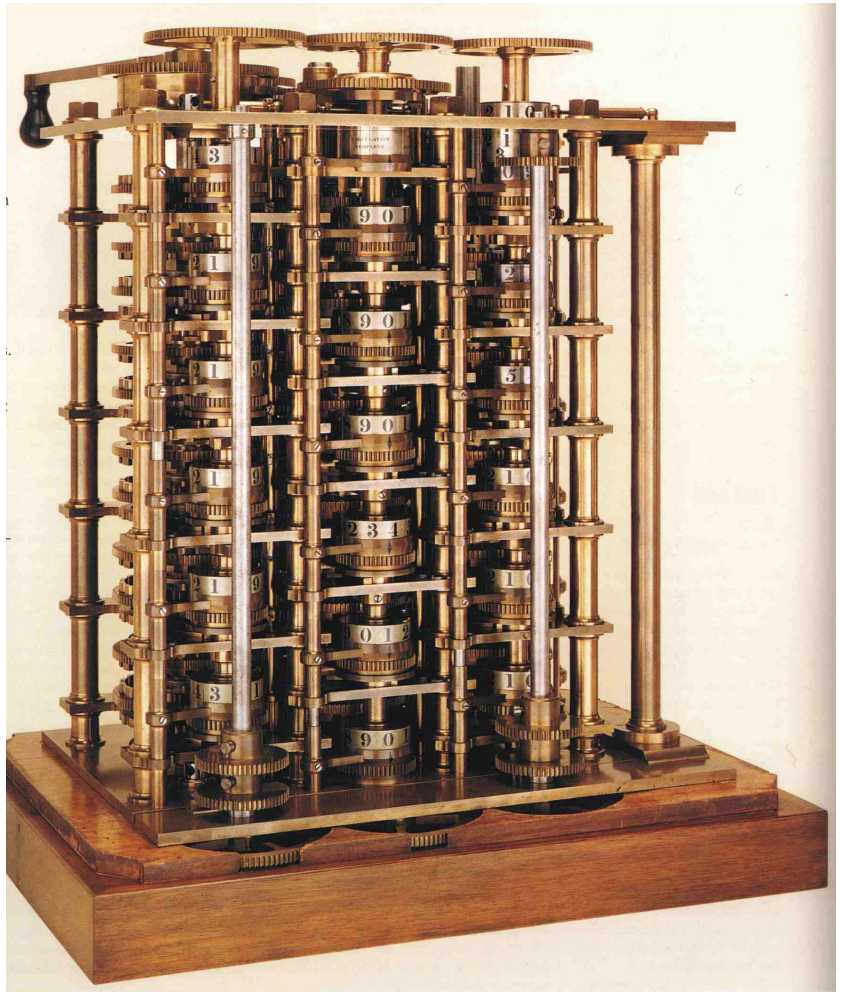
What is a computer?



What is a computer?



Babbage's Difference Engine

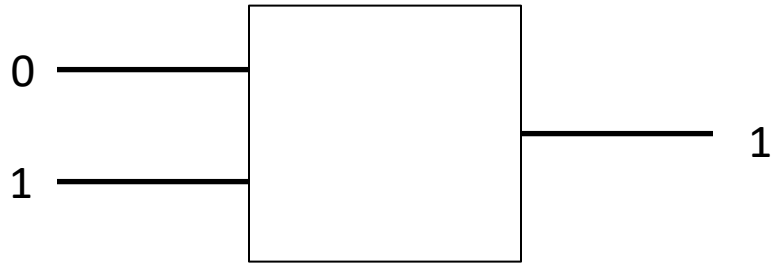


Computer Science History: Ada Lovelace

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

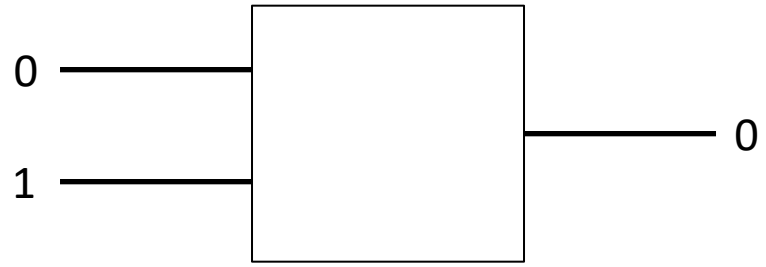
Comic by Kate Beaton, http://www.harkavagrant.com/index.php?id=298

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 from 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, ??

Some Levels of Abstraction

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

In This Class

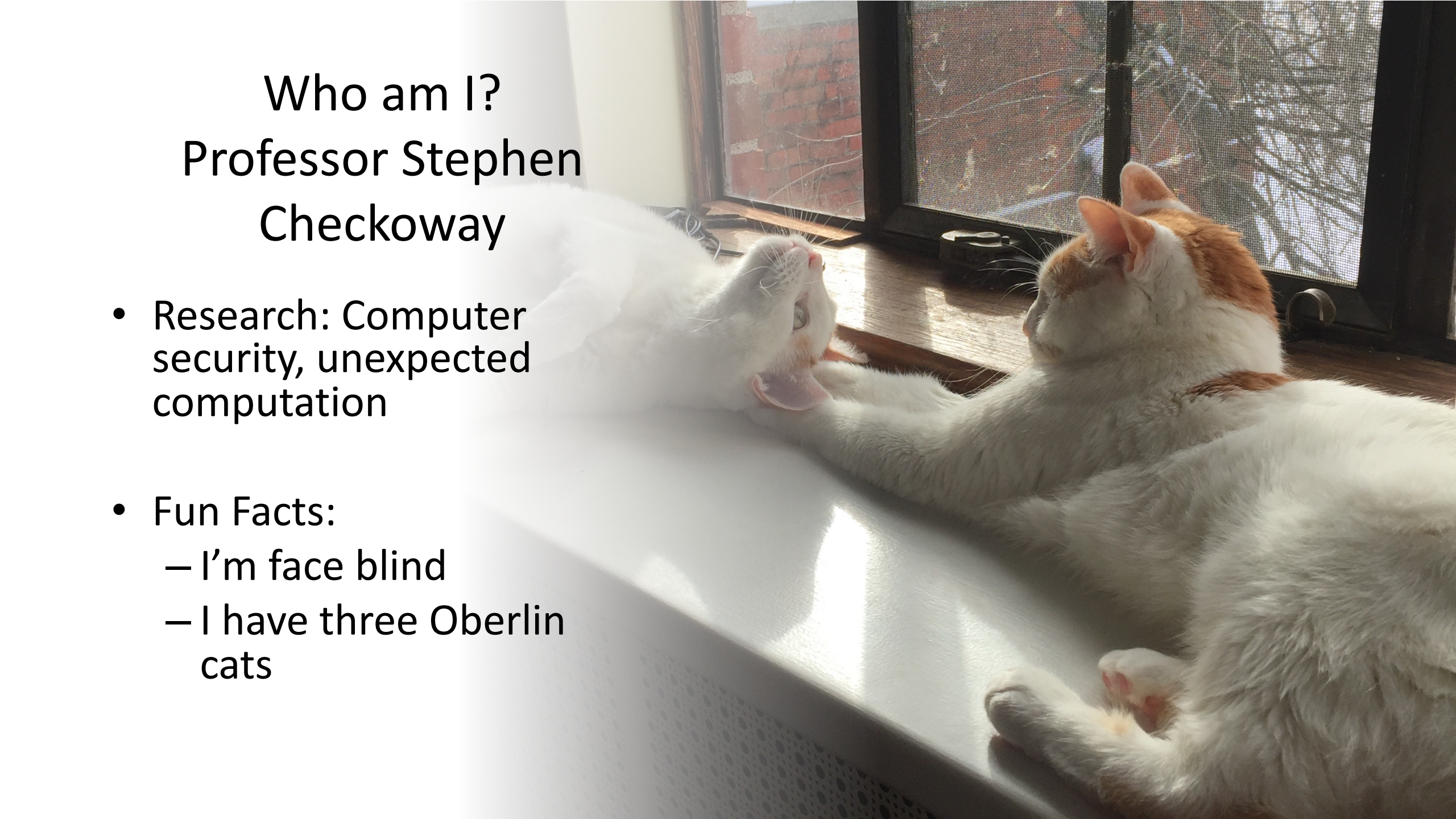
- What are the fundamentals we build these abstractions on top of?

- How do we create these abstractions?

Who am I? Professor Stephen Checkoway

- Research: Computer security, unexpected computation

- Fun Facts:
  - I’m face blind
  - I have three Oberlin cats



Office Hours

- Please come to office hours (currently on Thursday from 13–15 in King 231; but subject to change)
- I'm also available at other times by appointment
- I really am face blind and it *really* helps if you introduce yourself when you come to office hours

Class will be graded based on:

- Labs — Programming assignments
- Problem Sets — Written assignments
- Reading Exercises — Short daily quizzes; **due before class starting Friday!** On GradeScope, linked from BlackBoard
- Class participation — Clicker questions!
- Final project – simulating memory and running experiments

Labs

- Programming assignments designed to explore the architecture concepts we learn in class
  - MIPS assembly, Rust, 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

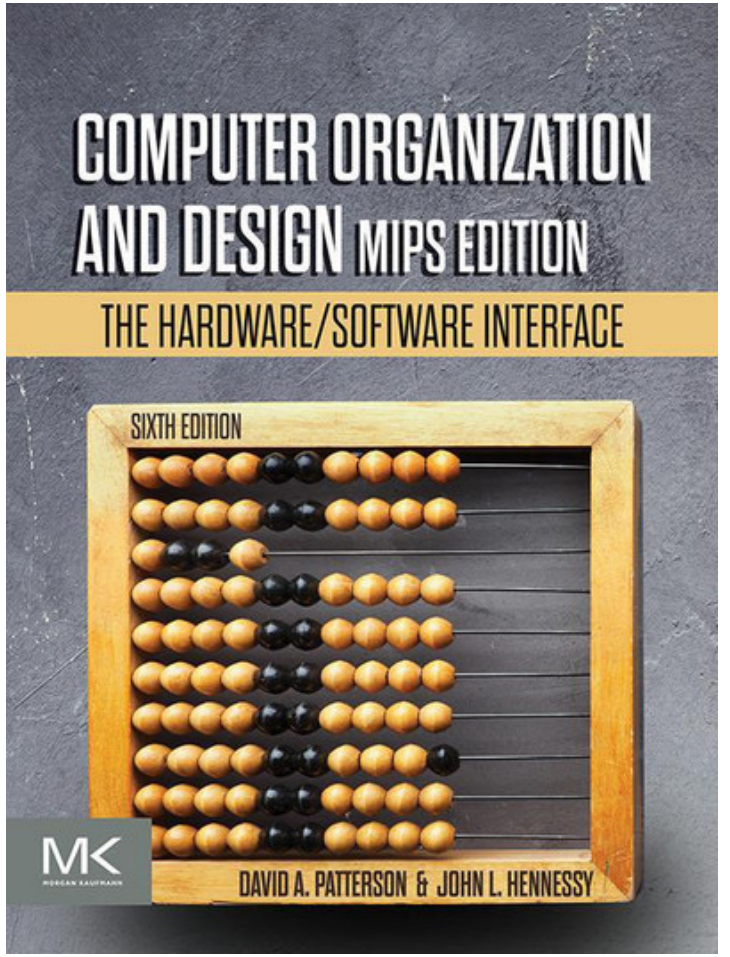
- Can be resubmitted within 1 week of receiving your grade (except for problem set 0 and any that aren't graded before reading period, usually 11 and 12)

- Final problem set grade is 25% your original submission grade, 75% your new grade.

- Problem Set 0 **due this Friday**, February 6 at 23:59

Reading

- We will be using Patterson and Hennessy's *Computer Organization and Design*
- This book is available to you for free from O'Reilly
- You'll need to create an account with your Oberlin email address if you don't already have one; instructions on the course web page (linked from BlackBoard)



Reading

- Due BEFORE CLASS on the day it is listed on the class schedule
- All readings exercises are on GradeScope
- FIRST READING DUE Friday

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 next Friday

iClicker Accounts

- You must create an iClicker account to ensure your grades are counted:
- Visit 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

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; there are no competitions here

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
- 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

AI Policy

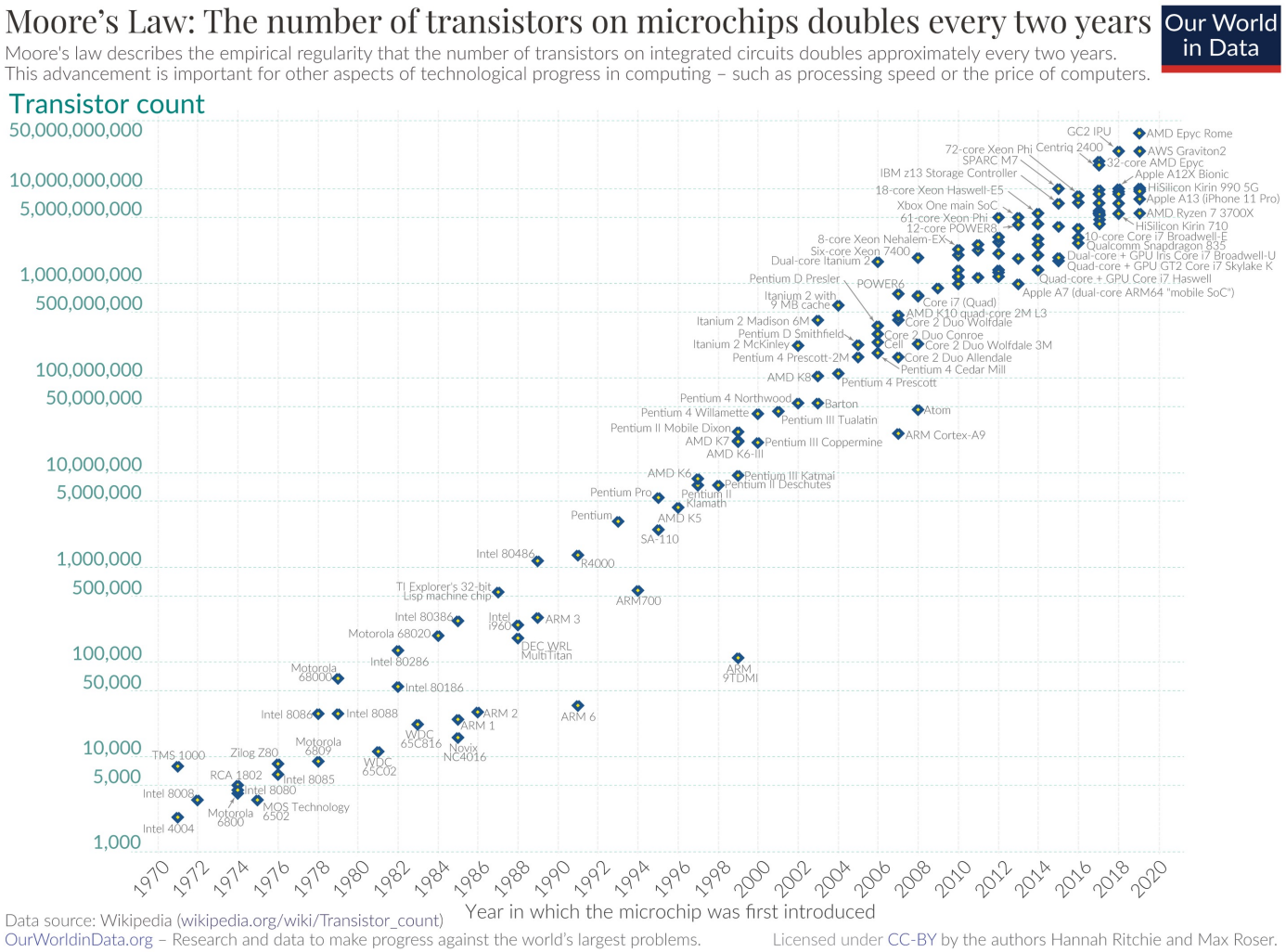
- Don’t use generative AI to solve your homework

Questions?

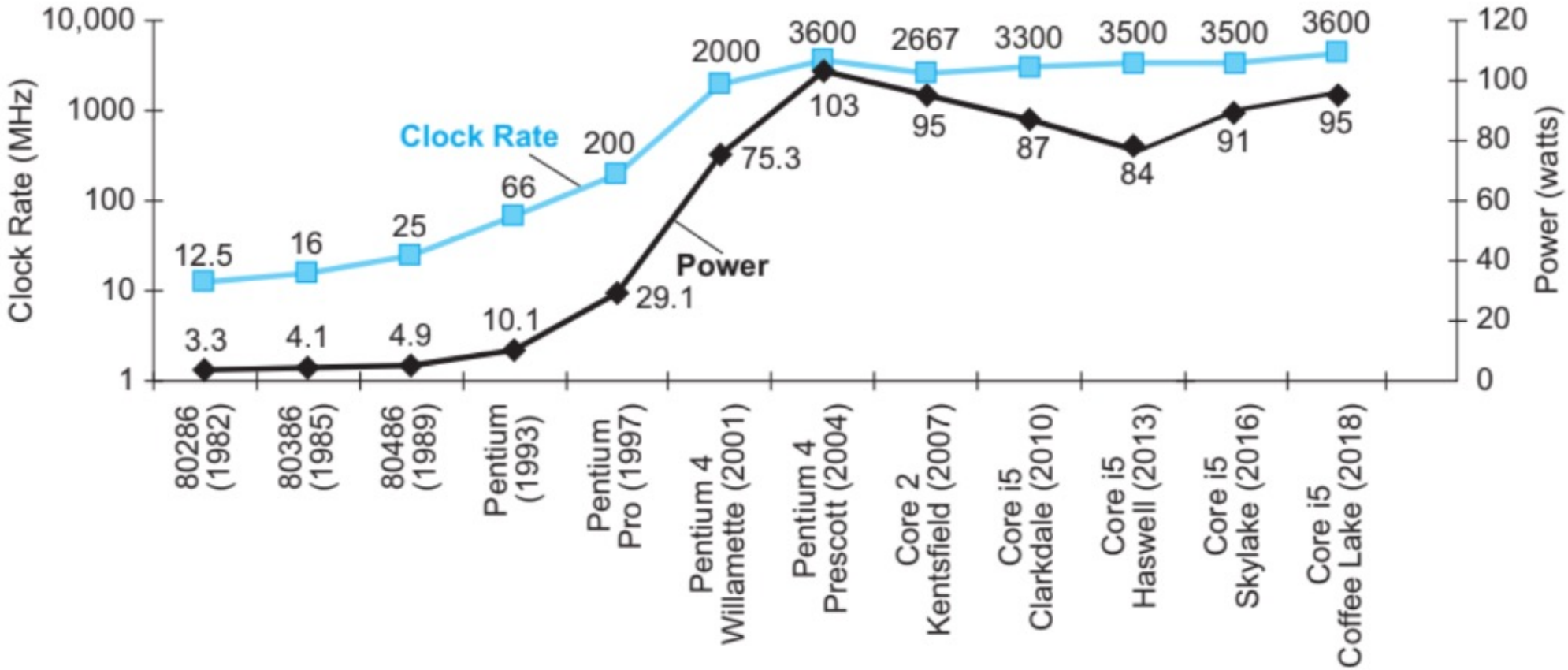
The Challenge of Computer Architecture

- 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
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.
Our World in Data
Transistor count
50,000,000,000
10,000,000,000
5,000,000,000
1,000,000,000
500,000,000
100,000,000
50,000,000
10,000,000
5,000,000
1,000,000
500,000
100,000
50,000
10,000
5,000
1,000
GC2 IPU
AMD Epyc Rome
72-core Xeon Phi
Centriq 2400
AWS Graviton2
SPARC M7
32-core AMD Epyc
IBM z13 Storage Controller
Apple A12X Bionic
18-core Xeon Haswell-E5
HiSilicon Kirin 990 5G
Xbox One main SoC
Apple A13 (iPhone 11 Pro)
61-core Xeon Phi
AMD Ryzen 7 3700X
12-core POWER8
HiSilicon Kirin 710
8-core Xeon Nehalem-EX
10-core Core i7 Broadwell-E
Six-core Xeon 7400
Qualcomm Snapdragon 835
Dual-core Itanium 2
Dual-core + GPU Iris Core i7 Broadwell-U
Quad-core + GPU GT2 Core i7 Skylake K
Pentium D Presler
POWER6
Quad-core + GPU Core i7 Haswell
Itanium 2 with 9 MB cache
Apple A7 (dual-core ARM64 "mobile SoC")
Core i7 (Quad)
AMD K10 quad-core 2M L3
Itanium 2 Madison 6M
Core 2 Duo Wolfdale
Pentium D Smithfield
Core 2 Duo Conroe
Cell
Itanium 2 McKinley
Core 2 Duo Wolfdale 3M
Pentium 4 Prescott-2M
Core 2 Duo Allendale
Pentium 4 Cedar Mill
AMD K8
Pentium 4 Prescott
Pentium 4 Northwood
Barton
Pentium 4 Willamette
Atom
Pentium III Tualatin
Pentium II Mobile Dixon
ARM Cortex-A9
AMD K7
Pentium III Coppermine
AMD K6-III
AMD K6
Pentium III Katmai
Pentium II Deschutes
Pentium Pro
Pentium II
Klamath
Pentium
AMD K5
SA-110
Intel 80486
R4000
TI Explorer's 32-bit Lisp machine chip
ARM700
Intel 80386
Intel i960
ARM 3
Motorola 68020
DEC WRL MultiTitan
Intel 80286
ARM 9TDMI
Motorola 68000
Intel 80186
Intel 8086
Intel 8088
ARM 2
ARM 6
ARM 1
WDC 65C816
Motorola 6809
Novix NC4016
TMS 1000
Zilog Z80
WDC 65C02
RCA 1802
Intel 8085
Intel 8008
Intel 8080
Motorola 6800
MOS Technology 6502
Intel 4004
1970
1972
1974
1976
1978
1980
1982
1984
1986
1988
1990
1992
1994
1996
1998
2000
2002
2004
2006
2008
2010
2012
2014
2016
2018
2020
Year in which the microchip was first introduced
Data source: Wikipedia (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

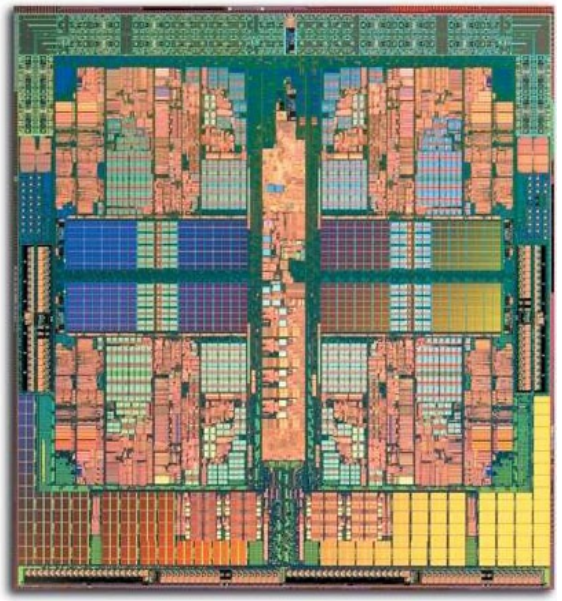


Cooling

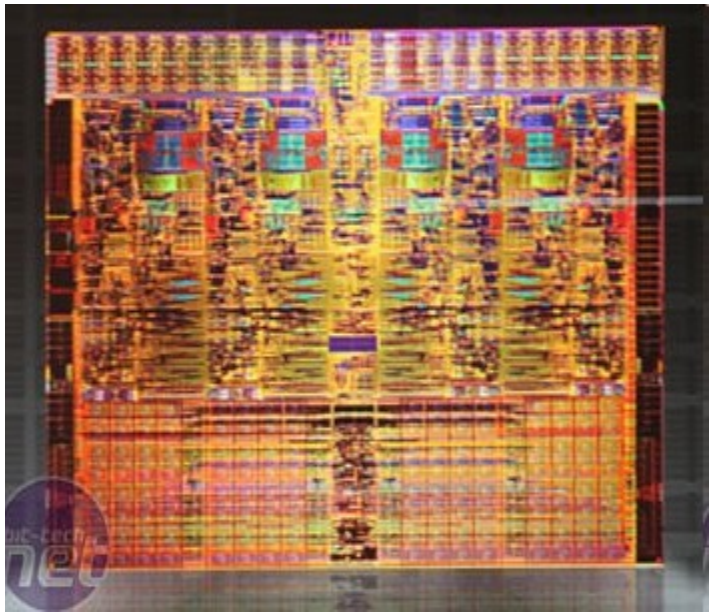
- CPUs get HOT
  - Switching those little transistors on and off takes power!
  - Power turns into heat.
- If they get too hot, they will burn out
- Can no longer efficiently cool all transistors on a chip at speeds we would like

All is not lost

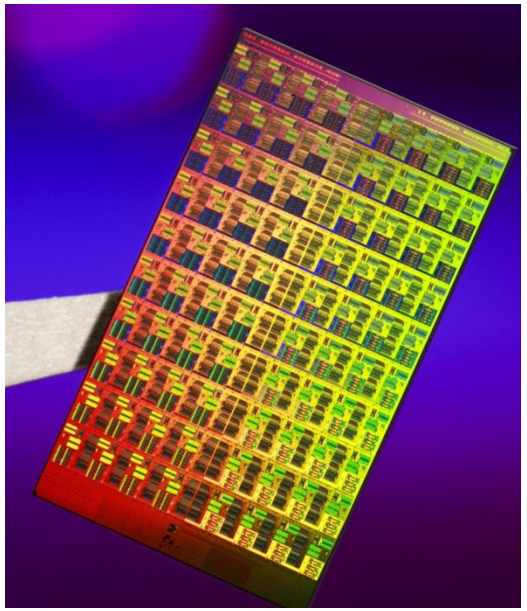
- If we can’t run a single instruction faster, what if we run a bunch of instructions at once?



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 influenced 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
  - Read book Section 1.3 and answer questions on GradeScope

- Problem Set 0 due Friday