ECE/CS 250
Computer Architecture

Introduction

Duke University

Slides are derived from work by
Andrew Hilton (Duke), Alvy Lebeck, Lee (Duke),
Benjamin Lee (Duke), and Amir Roth (Penn)

Instructor

- Professor: Dan Sorin
    - Office: I wish!
    - Email: sorin@ee.duke.edu

      Office Hours: TBD

- Who am I?
    - Duke BSE ‘96 (yes, I saw Bobby Hurley & Grant Hill at Cameron)
    - Wisconsin PhD ‘02 (yes, I saw awful basketball but good hockey)
    - Computer architect
        - I took equivalent of this course and made this my career
        - Might happen to you, too! ☺
    - Co-founder of Realtime Robotics

Graduate TAs and Teaching Associate

- Graduate TAs
  - Shiyu Li - ECE PhD student
  - Entropy Xu – CS PhD student
  - Eric Yeats - ECE PhD student

- Teaching Associate
  - Yesenia Velasco (CS)

Undergrad Teaching Assistants

- Undergraduate TAs (UTAs)
  - ~33 awesome undergrads who aced this class and like to help
  - Head UTA: Anshu Dwibhashi
- Will help with
  - Answering questions on piazza
  - Holding office hours to help with tools and software
  - Recitation work
- Will NOT bail you out at 4pm when deadline is at 5pm

- Please treat all TAs with respect and kindness

Getting Answers to Questions

- There are too many students for you all to email me
  - So now what do you do if you have a question?
  - You go through this list in this order …

1. Course website: static info
   http://www.ee.duke.edu/~sorin/ece250/
   - syllabus, schedule, rules/policies, prof/TA info, office hour info
   - links to useful resources
2. Sakai: dynamic added info
   - From me: lecture slides, announcements, assignments, grades
   - From you: uploaded homeworks

Continued on next slide …

Getting Answers to Questions

3. Piazza: questions/answers

- Post all questions here – then everyone can see the answer(s) posted there by me, a TA, or your fellow classmate
- Piazza is linked from the Sakai site (look on left-hand side)
- Questions must be "public" unless very good reason otherwise
- You must search Piazza first to see if question already answered
- Professor and TAs will NOT answer direct emails about homeworks or anything that pertains to more than 1 student

4. Contact TA directly if: grading issue

More on this issue in a few slides

continued on next slide …

Getting Answers to Questions

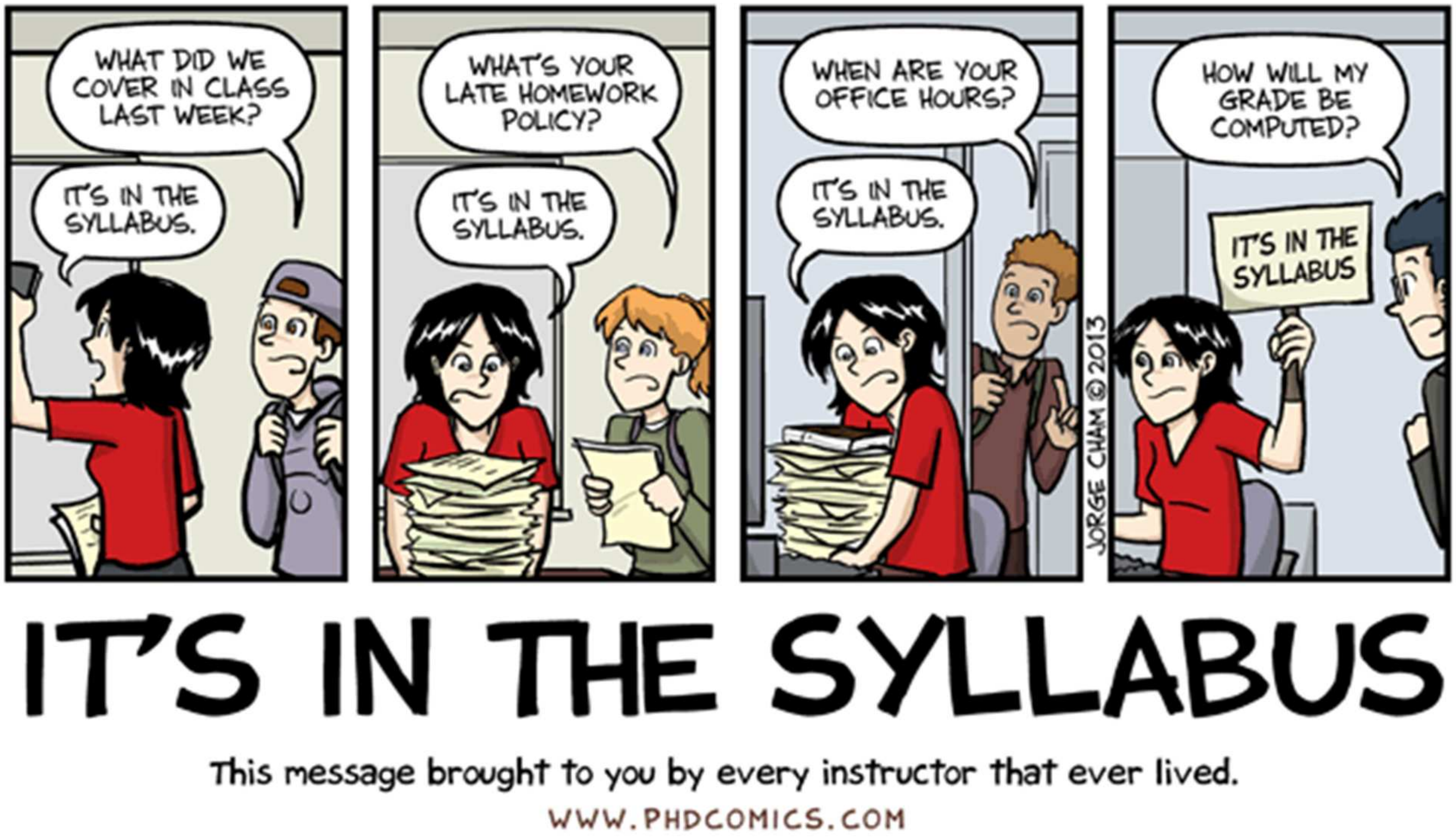
5. Contact teaching associate (Ms. Velasco) directly if issue that is specific to you and that can’t be posted on Piazza (e.g., missing exam)

6. Contact professor if none of the above have satisfied your problem.

Note 1: I really dislike this system, but there is just no way for me to handle issues from 400+ students.

Note 2: I will find ways to meet (many of) you

Getting Answers to Questions


WHAT DID WE COVER IN CLASS LAST WEEK?
IT'S IN THE SYLLABUS.
WHAT'S YOUR LATE HOMEWORK POLICY?
IT'S IN THE SYLLABUS.
WHEN ARE YOUR OFFICE HOURS?
IT'S IN THE SYLLABUS.
HOW WILL MY GRADE BE COMPUTED?
IT'S IN THE SYLLABUS
JORGE CHAM © 2013
IT'S IN THE SYLLABUS
This message brought to you by every instructor that ever lived.
WWW.PHDCOMICS.COM

Textbook

- Text: *Computer Organization & Design* (Patterson & Hennessy)
  - 5th edition of the textbook
  - Not the "ARM edition"
  - Nor the "Revised Printing"
  - Not even the "Super-special Extra-minty Revised Revision"

- Very highly recommended, particularly for those of you who learn best visually and who would benefit from more examples and (somewhat) different explanations
  - Could some of you survive without it?   Yes.

- We will not cover material in the textbook in a strictly linear fashion

Notes About Lectures and Lecture Slides

- Lecture slides available on Sakai before class
  - Value (just reading slides) << Value (attending/watching class)
  - Missing class = missing important course material
- Lectures will be recorded in cloud
  - I did this even before it was cool (i.e., before pandemic)
  - If possible, I encourage you to attend class lectures
    - You can ask questions
    - It adds structure to our weird situations
    - It prevents you from falling behind

Other Resources

- There are many online resources, including:
  - Unix tutorials
  - C programming tutorials
  - Coursera course on programming by Prof. Hilton (Duke)
  - Coursera course on computer architecture
  - Panopto lecture recordings from prior semesters
  - Etc.
- Some useful links on course website
- Feel free to use these materials, but none are required

Workload

- Readings from textbook
- Homework assignments – **done individually**
  - Pencil and paper problems
  - Programming problems in C and assembly
  - Digital logic design problems (like designing a computer)
- Recitations – **done with partners (or individually)**
  - During recitations, work with partners/groups (or individually) on ungraded "assignments" to help you learn skills you will need for homeworks and tests
  - I repeat: you will struggle and/or fail on homeworks if you don't learn the material in recitations
  - Goal: learning through hands-on, low-stress practice
  - UTAs will help students in real-time during synch. recitations

Lectures & Recitations are Required

- We rely on you attending lectures and recitations
  - It's ok to have questions and struggle, but we need you to do the best you can to follow lectures and recitations
- We use Piazza and office hours to help students with questions and students who are struggling
  - And we are very happy to do so!
- We CANNOT use Piazza and OH as substitutes for lecture and recitations
  - This places impossible demand on our TAs
  - Do not skip lectures and recitations and then ask for individual instruction

Grading

- Grade breakdown

Homework	45%
Midterm #1	15%
Midterm #2	15%
Final Exam	25%

Exams will be asynchronous.
More on that later.

- Late homework policy – **no exceptions, no extensions**
  - 0-24 hours late: Take earned score and multiply by 0.9
  - 24-48 hours late: Take earned score and multiply by 0.8
  - >48 hours late: no credit
  - Applies to whole assignment (not per question)

Policy will be applied uniformly and consistently so as to be fair to all.

A Bit More on Grading

- I want you all to succeed!
  - My goal is for you to learn and thrive
  - My goal is NOT to find ways to give out low grades

- There is no curve → how others perform has no impact on your grade → don't worry about others
- If you all earn an A, I will give you all an A
  - There is no predetermined grade distribution

- Corollary: I will strive to be as fair as possible, so as to give all of you the exact same opportunity
  - I won't play favorites
  - All rules will be applied consistently

How You Will Be Graded

You will be graded on what you submit and when you submit it.

- Excuses that will NOT be considered
  - I meant to turn in a different file
  - I did all the work but somehow forgot to submit it
  - I got confused and submitted the wrong version of the file
  - I was only one minute late
  - I was really busy these past few days
  - I thought it uploaded to Gradescope → double-check uploads
  - I just forgot to submit one of the files → double-check uploads
  - My computer crashed last night → use git, back up your work
- I will accept dean's excuses and (most) STINFs

Homework #0 will (re-)cover these issues

Homework Grading Procedure

- UTAs grade homework assignments
- UTAs follow grading rubrics that we share with you
- Should be little ambiguity about where grades come from

If You Believe Your Grade is Incorrect

- If you believe your grade is incorrect, you have 5 (FIVE) days to raise your complaint.
  - No complaints after 5 days will be considered
- For homeworks, follow this procedure:
  - Contact the UTAs who graded it
    - Your discussion section UTAs grade your homework
  - If still unsatisfied, contact grad TA
  - If still unsatisfied, contact teaching associate
- For exams, follow this procedure:
  - Contact a grad TA
  - If still unsatisfied, contact teaching associate

Academic Misconduct

- Academic Misconduct
  - Refer to Duke Community Standard
  - Homework is individual – you do your own work
  - Common examples of cheating:
    - Running out of time and using someone else's output
    - Borrowing code from someone who took course before
    - Using solutions found on Web
    - Having a friend help you to debug your program
- I will not tolerate any academic misconduct!
  - Software for detecting cheating is very, very good … and I use it
  - All 250 profs catch cheating and refer to Office of Student Conduct
- "But I didn't know that was cheating" ≠ valid excuse

Academic Misconduct

- The last time I taught this class, I sent 15 students to OSC
  - 2 were suspended for a year
  - It was a royal pain in the cache [pun may be funny in 7 weeks]
- I get no joy from catching students, but I will not tolerate cheating
  - It's unfair to honorable students doing their own work
- You are far better off turning in your own work, even if it's not great
- Reminder: you are not in CS 201 any more

- I will not tolerate any academic misconduct!
  - Software for detecting cheating is very, very good … and I use it
  - All 250 profs catch cheating and refer to Office of Student Conduct

- "But I didn't know that was cheating" $\neq$ valid excuse

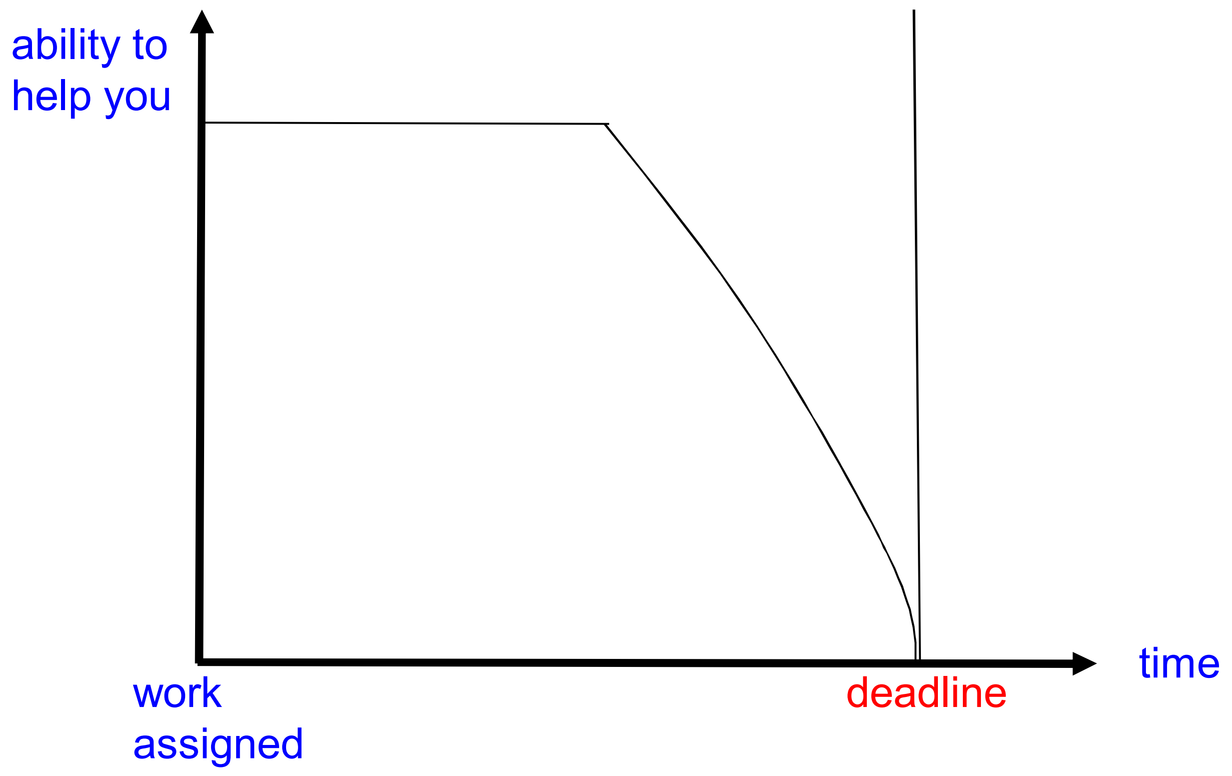
A Small Bit of My Teaching Philosophy

Summary: I will treat you like responsible adults

- I ~~won't~~ can't take attendance
    - Nor will I give pop quizzes to force attendance. I simply expect you to attend/watch lectures.
- I'll expect you to be responsible
    - **Don't forget to turn something in or accidentally turn in the wrong document or turn something in 2 minutes late. You'll be graded on what you turn in and when you do it.**
- I'll give you much more time to complete assignments than is necessary, and I'll let you manage your own time
    - Even if you get sick or get swamped in another class ~~or go out of town for an interview~~, etc., then you will still have enough time to complete assignments … but only if you start them early.

Start Assignments Early

- The TAs and I can help you if you start early. We are **very** happy to help you succeed in this class!!
- We'll try our best, but we can't do much to help you when you show up day before assignment is due.



Goals of This Course

- By end of semester:
  - You will know how computers work
    - What's inside a computer?
    - How do computers run programs written in C, C++, Java, Matlab, etc.?
  - You will design hardware that computers use
  - You will understand the engineering tradeoffs to be made in the design of different types of computers
  - You will learn some C programming
  - You may, like me, decide to become an architect. ☺
- If, at any point, it's not clear why I'm talking about some topic, please ask!

Outline of Introduction

- Administrivia
- What is a computer?
- What is computer architecture?
- Why are there different types of computers?
- What does the rest of this course look like?

Reading Assignment

- Patterson & Hennessy
    - Chapter 1
    - This is a short and relatively easy-to-read chapter

What is a Computer?

- A computer is just a machine
  - A bunch of switches and logic that we'll talk about later
- Yes, but what does this machine do?
  - Whatever you tell it to do! No more, no less
- A computer just does what software tells it to do
  - Software is a series of **instructions**
- ICQ (In-Class Question): What instructions does a computer need?

Computers Execute Instructions

- What kinds of instructions are there?
  - Arithmetic: add, subtract, multiply, divide, etc.
  - Access memory/storage: read, write
  - Conditional: if condition, then jump to other part of program
  - What other kinds of instructions might be useful?
- So how do computers run programs in Java or C/C++ or Matlab or whatever whippersnappers use these days?
  - None of us write programs in binary (zeros and ones) …
  - We'll get to this in a few minutes

Instruction Sets

- A computer can only execute instructions that are in its specific machine language
- Every family of computers has a different **instruction set** that it understands
  - Intel and AMD's IA-32 (x86): Core i7, AMD Ryzen, etc.
  - ARM: In many embedded processors (e.g., smartphones)
    - Used by many companies (e.g., Qualcomm, Apple)
  - Intel's IA-64: Itanium, Itanium 2
  - IBM's POWER: In IBM PCs, Cell Processor (in PS3!), old Macs
  - SPARC: In old computers from Oracle
  - MIPS: this is the example used in the textbook
- Note: no computer executes Java or C++
  - Not even Matlab (sorry, Dr. G)

Outline of Introduction

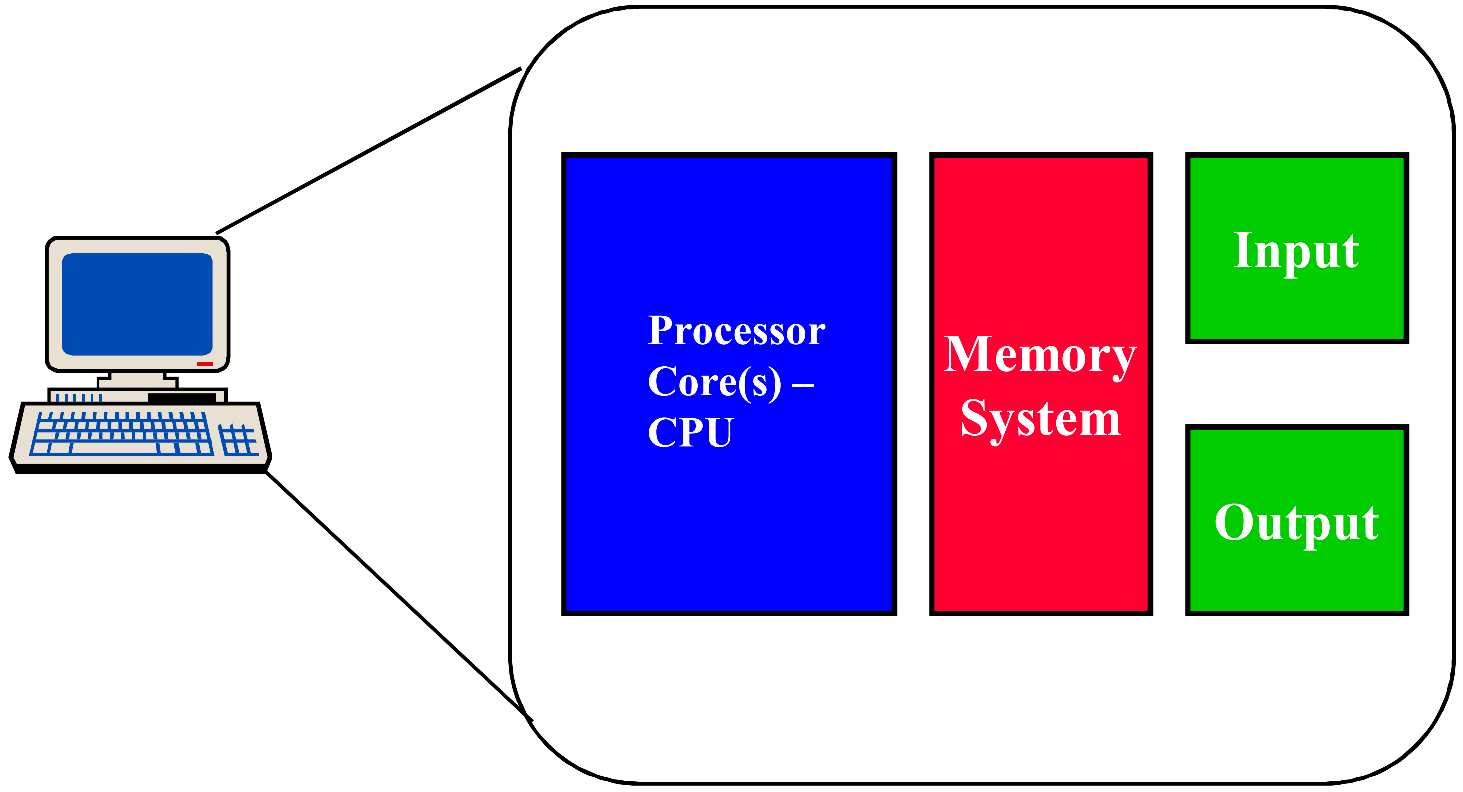
- Administrivia
- What is a computer?
- What is computer architecture?
- Why are there different types of computers?
- What does the rest of this course look like?

Hint: It Doesn't Involve Skyscrapers …

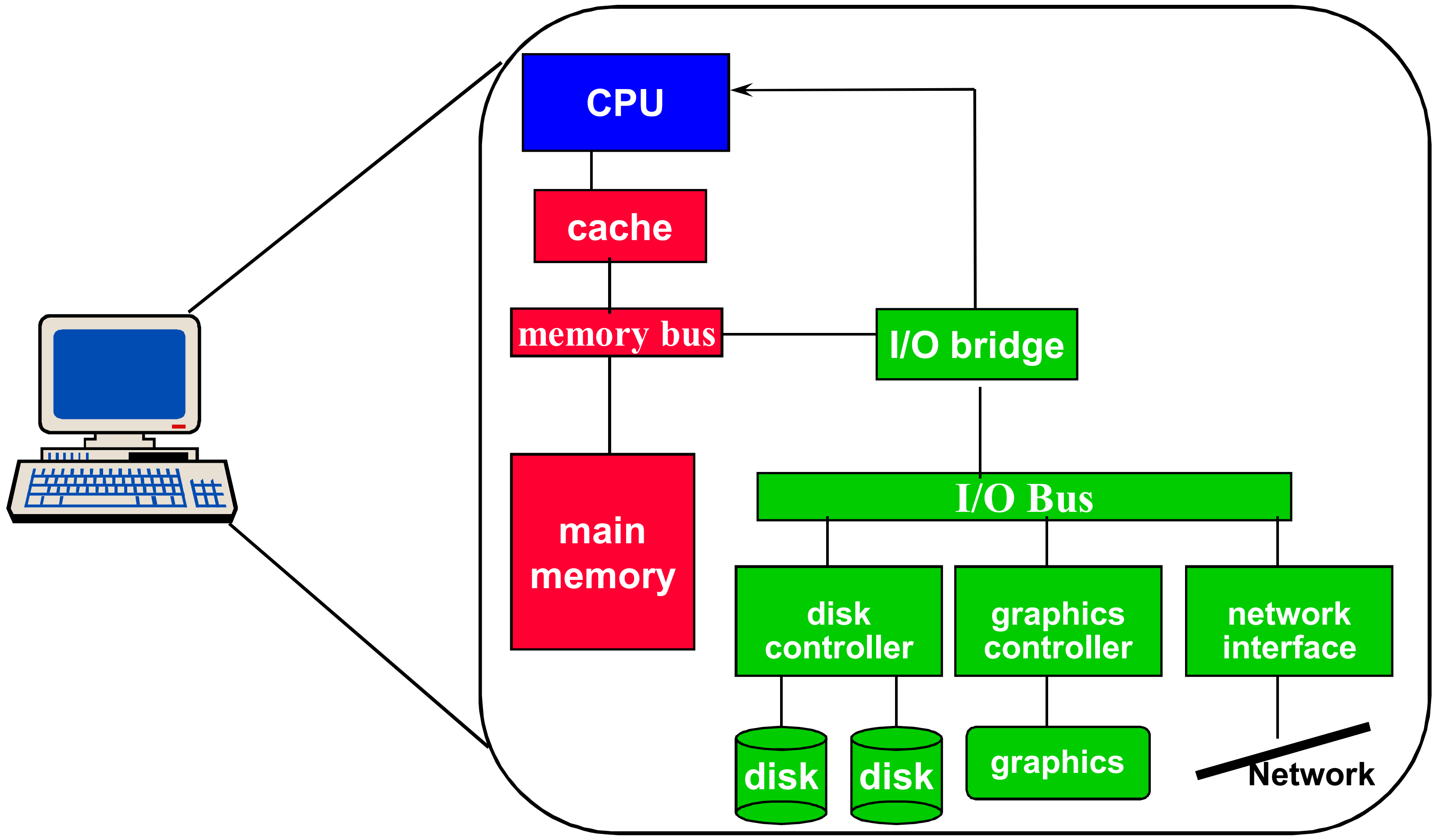
- Strictly speaking, **computer architecture** specifies what hardware looks like (its interface), so that we can write software to run on it
  - Exactly what instructions does it have
  - Number of memory/storage locations it has
  - And more that we'll learn about later in semester
- **Important point:** there are many, many different ways to build machines that provide same interface to software
  - There are many **microarchitectures** that conform to same architecture
  - Some are better than others! If you don't believe me, I'll trade you my original Intel Pentium for your Intel Core i7
- ICQ: So what's inside one of these machines?

The Inside of a Computer

- The Five Classic Components of a Computer

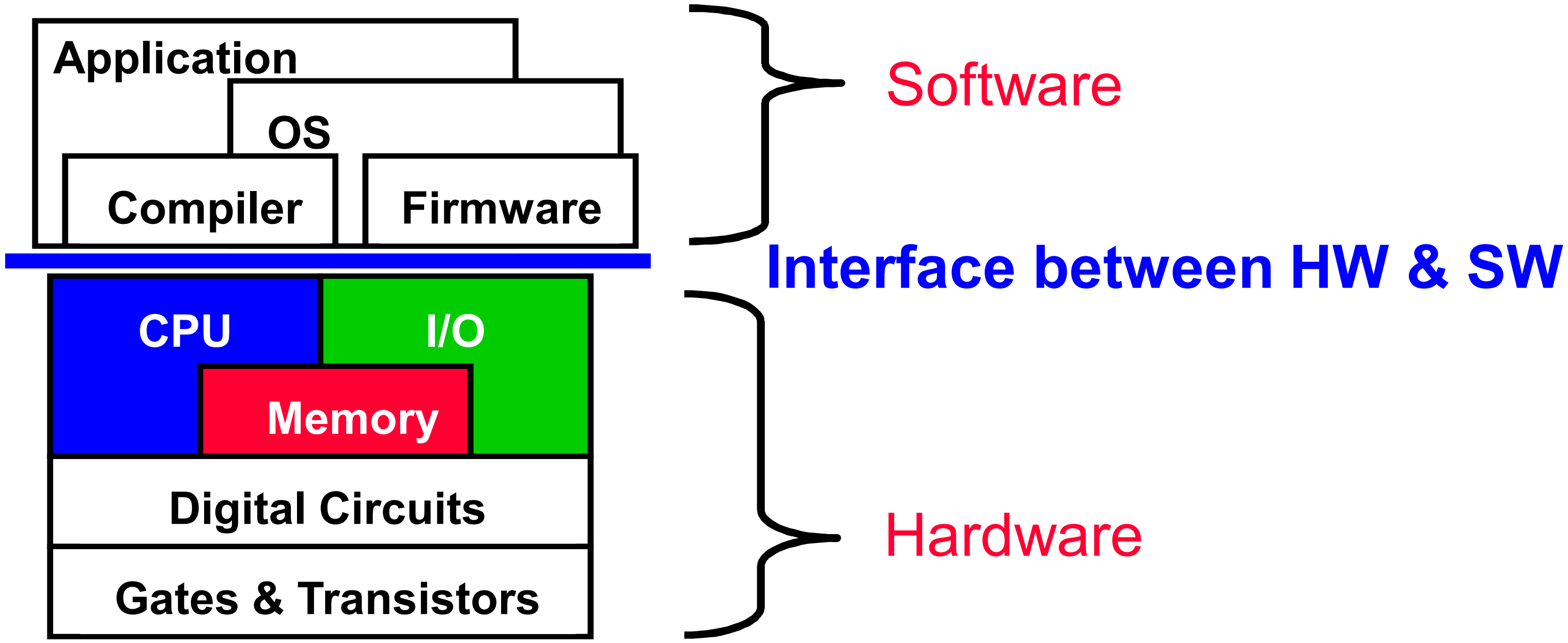


System Organization



What Is ECE/CS 250 All About?

- Architecture = interface between hardware and software



- ECE/CS 250 = design of CPU, memory, and I/O
- ECE/CS 350 = building it in hardware

Outline of Introduction

- Administrivia
- What is a computer?
- What is computer architecture?
- Why are there different types of computers?
- What does the rest of this course look like?

Differences Between Computers

- We have different computers for different purposes

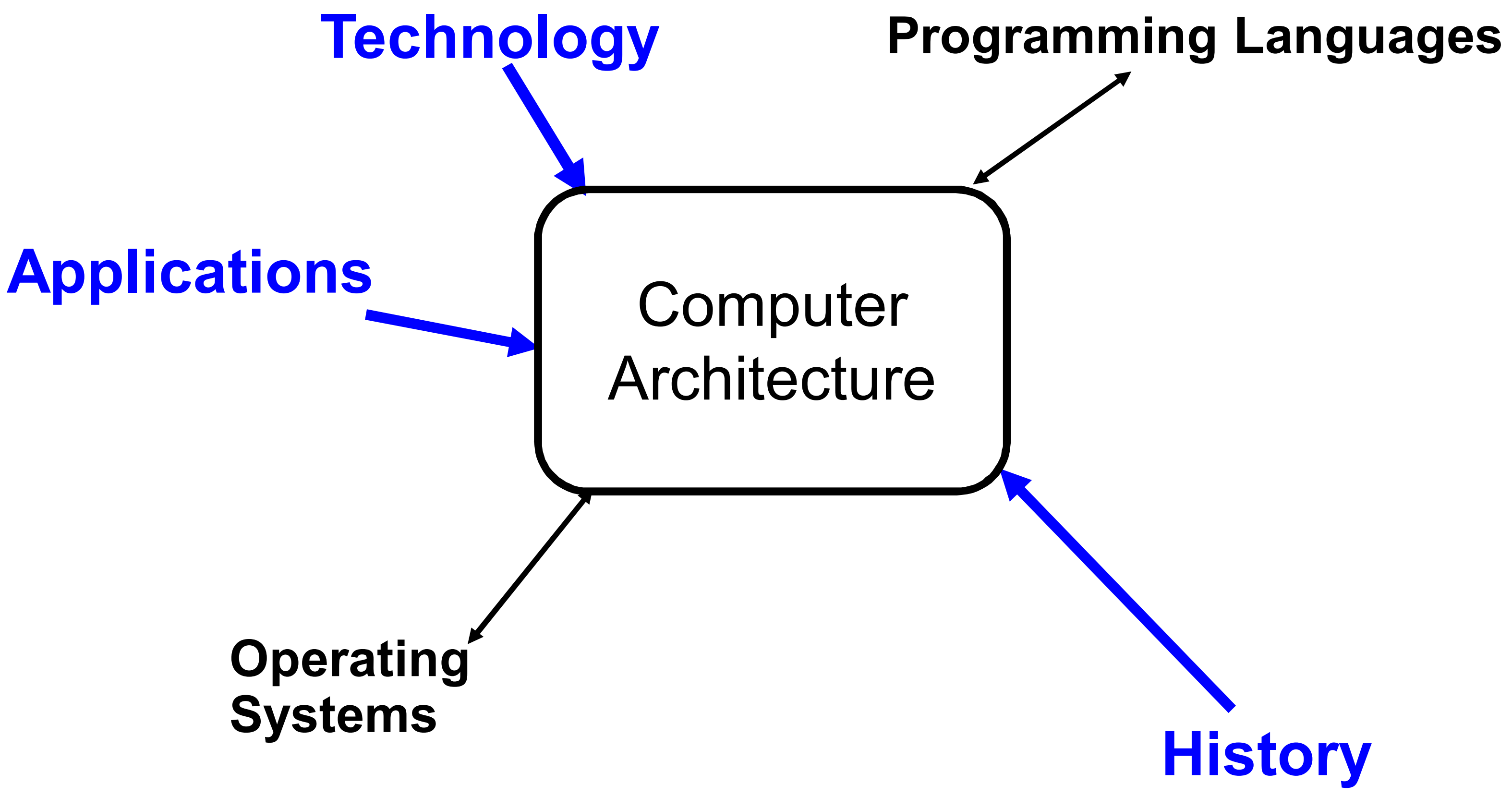
- Some can achieve performance needed for high-performance gaming
  - E.g., AMD APU in Xbox One
- Others can achieve decent enough performance for smartphone without using too much power
  - E.g., Qualcomm Snapdragon, Intel Pentium M (M=mobile), etc.
- And yet others can function reliably enough to be trusted with the control of your car’s brakes

ICQ: What computers do you use?
ICQ: Which of those computers do you own?

Kinds of Computers

- "Traditional" personal computers
  - Laptop, desktop, tablet
- Less-traditional personal computers
  - Smartphone, smart watch, gaming system, ~~Google glass~~, etc.
- Hidden computers (many are now in "the cloud")
  - Mainframes and servers for business, science, government
    - E.g., the machines that run DukeHub, Gmail, Instagram, etc.
  - Google has many thousands of computers (that you don't see)
- Hidden embedded computers
  - Controllers for cars, airplanes, ATMs, toasters, robots, etc.
  - Far and away the largest market for computers!
- Other kinds of computers??

Forces on Computer Architecture



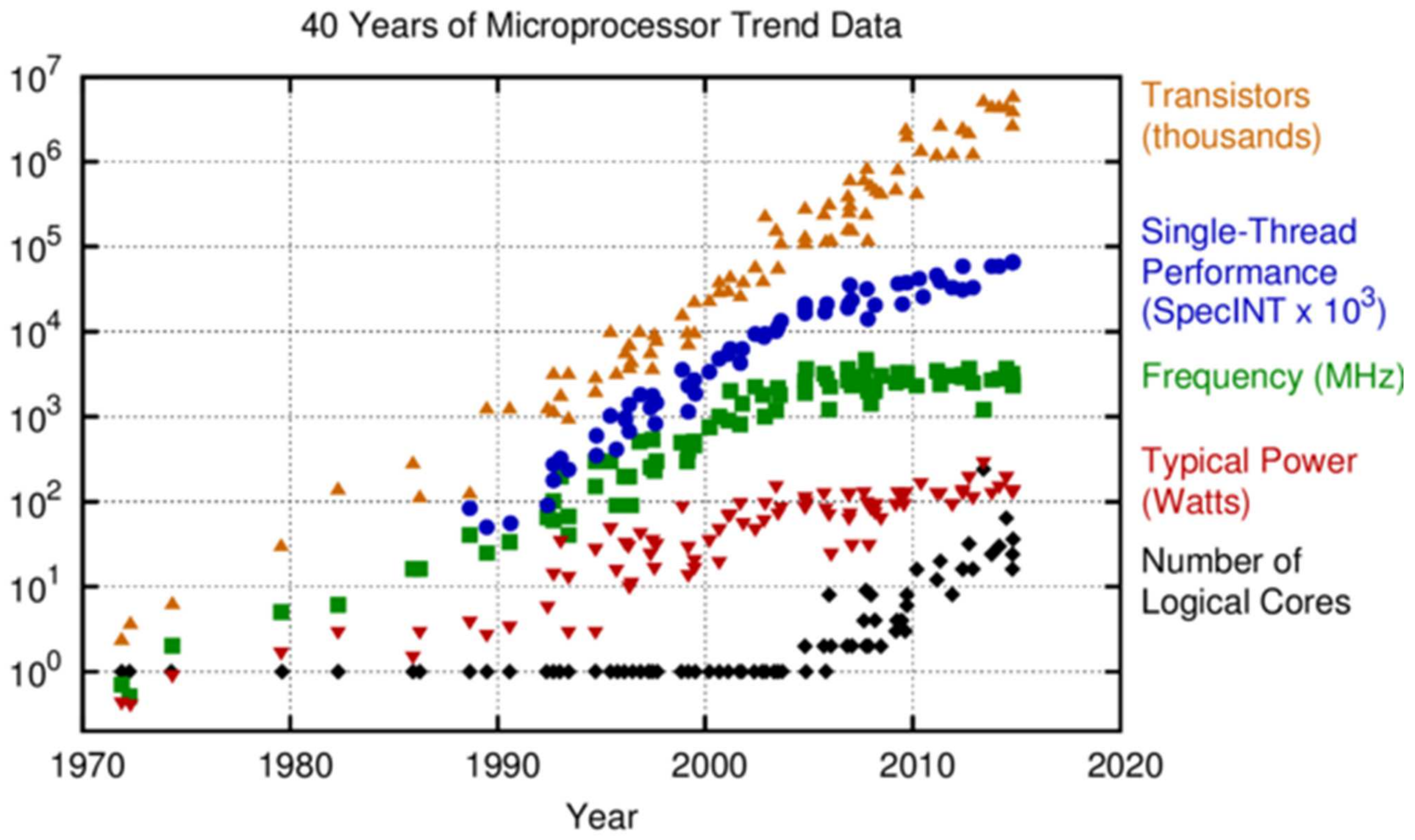
A Very Brief Early History of Computing

- 1645 Blaise Pascal's Calculating Machine
- 1822 Charles Babbage
  - Difference Engine
  - Analytic Engine: Augusta Ada King = first programmer
- < 1946 Eckert & Mauchly
  - ENIAC (Electronic Numerical Integrator and Calculator)
- 1947 John von Neumannn
  - Proposed the Stored Program Computer
  - Virtually all current computers are "von Neumann" machines
- 1949 Maurice Wilkes
  - EDSAC (Electronic Delay Storage Automatic Calculator)

Progress in Commercial Computers

Year	Name	Size (cu. ft.)	Adds/sec	Price
1951	UNIVAC I	1000	1,900	$1,000,000
1964	IBM S/360 Model 50	60	500,000	$1,000,000
1965	PDP-8	8	330,000	$16,000
1976	Cray-1	58	166 million	$4,000,000
1981	IBM PC	desktop	240,000	$3,000
1991	HP 9000 / model 750	desktop	50 million	$7,400
1996	PC with Intel PentiumPro	desktop	400 million	$4,400
2002	PC with Intel Pentium4	desktop/laptop/rack	4 billion	$1-2K
2008	Cell processor	PlayStation3	~200 billion	~$350 (eBay)
2014	Nvidia K40 GPU	Desktop/rack	~4.3 trillion	$4,000
2018	AMD/Radeon Vega 64	Desktop/rack	12.7 trillion	$400

Microprocessor Trends (for Intel CPUs)



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2015 by K. Rupp

What Do Computer Architects Do?

- Full disclosure: I'm a computer architect
- Design new microarchitectures
  - Very occasionally, we design new architectures
- Design computers for ever-changing needs & challenges
  - Tailored to new applications (e.g., image/video processing)
  - Amenable to new technologies (e.g., faster and more plentiful transistors)
  - More reliable, more secure, use less power, etc.
- Computer architecture is engineering, not science
  - There is no one right way to design a computer → this is why there isn't just one type of computer in world
  - This does not mean, though, that all computers are equally good

Course Outline

- Introduction to Computer Architecture
- C Programming and From C to Binary (next!)
- Instruction Sets & Assembly Programming
- Processor Core Design
- Memory Systems
- I/O Devices and Networks
- Pipelined Processor Cores
- Multicore Processors

The Even Bigger Picture

- ECE/CS 250: Basic computer design
  - Finish 1 instruction every 1 very-long "clock cycle"
  - Finish 1 instruction every 1 short cycle (using pipelining)
- ECE/CS 350: Implementing digital computers/systems
- ECE 552/CS 550: High-performance computers + more
  - Finish ~3-6 instructions every very-short cycle
  - Multiple cores each finish ~3-6 instructions every very-short cycle
  - Out-of-order instruction execution, power-efficiency, reliability, security, etc.
- ECE 652/CS 650: Highly parallel computers and other advanced topics
- Should also learn: Operating Systems & Compilers