ECE 250 / CS 250
Computer Architecture

“Overview”

Benjamin Lee

Some slides based on those from Alvin Lebeck, Daniel Sorin, Andrew Hilton, Amir Roth, Gershon Kedem

General Information

- Instructor: Benjamin Lee
  - Office: Hudson 210
  - email: benjamin.c.lee@duke.edu
  - Office Hours: Tu/Th 2:40-4:00pm
    - » If you have classes/work/practice at both these times, email to schedule appt.
- Graduate TAs
  - Heather Duschl (heather.duschl@duke.edu)
  - Mohamed Ibrahim (mohamed.s.ibrahim@duke.edu)
  - Yuxuan Li (lyuxuan@cs.duke.edu)
- Undergraduate TAs
  - 14-15 UTAs
  - Final roster to be determined.

Access to Information and Communication

- Course web page
  - http://people.duke.edu/~bcl15/class/class_ece250spr14.html
  - Assignments, schedule, documentation/resources, etc.
  - Slides available shortly before or after class…(video too.)
- Sakai mostly for submitting homework and posting grades
  - Bulk email announcements sometimes
- Piazza for announcements, Q&A
  - Use this instead of sending email to instructor staff, best way to get a quick answer
  - Questions answered by UTA, grad TA, me, or other students
  - Many eyes on questions & answers, minimizes confusion and/or misinformation
- You are expected to monitor these modes of communication

General Info

- I AM NOT PERFECT
  - Ask questions, if you are confused or need clarification others probably need help too.
  - But, if you fall asleep I'll call you out! ☺
- Tuesday and Thursday
  - I lecture, you ask questions, we generally have a good time
- Wednesday and Friday (Recitation/Lab)
  - You are expected to attend these
  - You must complete the assigned tasks from recitation
  - Smaller sections (25 students)
  - 2 staff (UTAs or TA+UTA)
  - Hands on exercises
    - » Working problems, learning tools, etc.
  - If you don't have a laptop to bring to recitation, contact me

Resources for class

- Textbooks
  - *Computer Organization & Design 4th Ed. revised* (Patterson & Hennessy)
  - *The C Programming Language 2nd Ed.* (Kernighan and Ritchie) – optional but may be useful for learning C
- Course webpage includes links to other resources
  - E.g., recorded lectures from previous semester
- Office hours
  - Use them!
  - Do not expect help at 3am right before a due date
  - Do not expect a TA to stay extra long helping everyone the night before an assignment is due

Grading

- Grade Breakdown
  - 50% Homework (~7 assignments)
  - 12.5% Midterm I
  - 12.5% Midterm II
  - 25% Final
- Late Policy
  - 0-24 hours late: 10% penalty
  - 24-48 hours late: 20% penalty
  - > 48 hours late: No Credit
- No extra credit in this course
- Plan your time accordingly, start early!

Academic Conduct

- Do Not Cheat!
- Academic Conduct
  - Duke Community Standard
  - Studying together in groups is encouraged
  - Common examples of cheating: running out of time on an assignment and then use someone else's solution, person asks to borrow solution "just to take a look", using previous semester results, using solutions from the internet, copying an exam question, line by line help from another student …
  - All submitted work and programming must be your own, unless otherwise stated.
- Zero tolerance for academic misconduct
  - We do use software to compare your code/solutions, and it works well!
  - You'll get a zero, and I'll refer the case to the honor council

Course Problems

- Can’t make midterms / final, other conflicts
  - Tell us early and we will schedule an alternate time
- If you are having problems
  - See me
  - See DUS
  - See Academic Dean (very good resource)

Why Do You Have to Take This Course?

- You want to be a race car driver
- You all know how to drive
- To be successful you don't just drive
- You must "be in touch with your vehicle"
- You have to learn about the vehicle
  - Engine
  - Suspension
  - Tires
- Is it drag racing, monster trucks, NASCAR, endurance
  - Different cars
  - Different style of driving
- Who is going to win the Indy 500, random 15 year old with temporary permit or Jimmie Johnson?

Why Do You Have to Take This Course?

- You want to be a Computer Scientist / Engineer
- You know how to program (Compsci 101, 201)
- To be successful you don’t just program
- You have to understand the machine
    - Hardware: Processor, memory, disk, etc.
    - SW: Operating system, Programming Languages/Compilers
- What kind of computer scientist/engineer?
    - Computer architect, like me ☺
    - Databases, networks
    - Scientific computing (motion of planetary bodies, drug development, computational biology, economics, etc.)
    - Games, virtual reality
    - Embedded: Cell phones, mp3 player, cars
- Who’s code do you want controlling your brakes, airbag, airplane, financial transactions? Someone in CS 101 or a computer scientist / engineer?

Goals of the Course

- By end of semester:
  - You will know how to program in C on Linux
  - You will know how computers work
  - You will design a simple computer
  - You will understand some of the engineering tradeoffs to be made in the design of different types of computers
  - You will understand some of the program/application performance issues related to how software maps onto hardware
  - You will see how abstractions help manage complexity
    - Prof Astrachan, "from the abstract to the ridiculous"
    - Be prepared for follow on systems courses: Operating Systems, Networks, Databases, Digital Systems, Advanced Architecture, Embedded Systems, Compilers, etc.
  - 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!

Administrivia

- Remote Access
- Unix tutorial
- Signup for piazza
- If you need a permission number to register
  - See me after class
  - Email me (even if you see me after class)

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?

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**
- What instructions does a computer need?

Computers Execute Instructions

- What kinds of instructions are there?
  - Arithmetic: add, subtract, multiply, divide, etc.
  - Access memory: 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 python or whatever is trendy these days?
  - None of us write programs in binary (zeros and ones) …

Instruction Sets

- Computers can only execute instructions that are in their specific machine language
- Every type of computer has an instruction set
  - Intel (and AMD) IA-32 (x86): Pentium Core i7, AMD Opteron “Magny Cours”, etc.
  - ARM: In many embedded processors (e.g., smartphones)
  - Intel IA-64: Itanium, Itanium 2
  - PowerPC: In Cell Processor and old Apple Macs
  - SPARC: In computers from Sun Microsystems/Oracle
  - MIPS: MIPS R10000 → this is the example used in the textbook
- No computer directly executes Java, C++, Matlab, etc.

Outline of Introduction

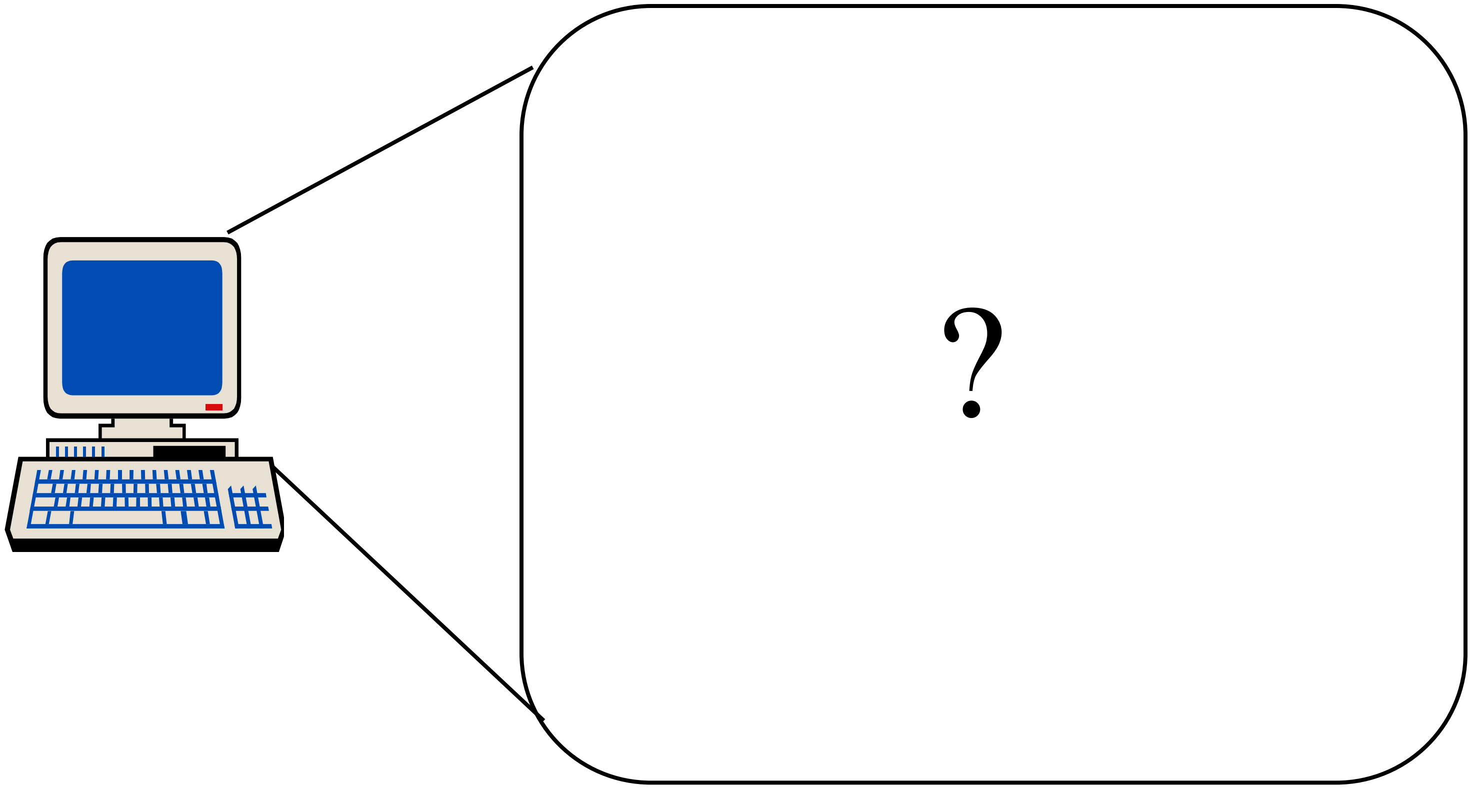
- Administrivia
- What is a computer?
- What is computer architecture?
- Why are there different types of computers?
- What does the rest of this course look like?

What is Computer Architecture?

- Computer architecture defines HW-SW interface
  - Defines set of instructions
  - Defines number of storage locations
- Microarchitecture defines the implementation of the computer architecture
  - Many microarchitectures 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
- So what’s inside one of these machines?

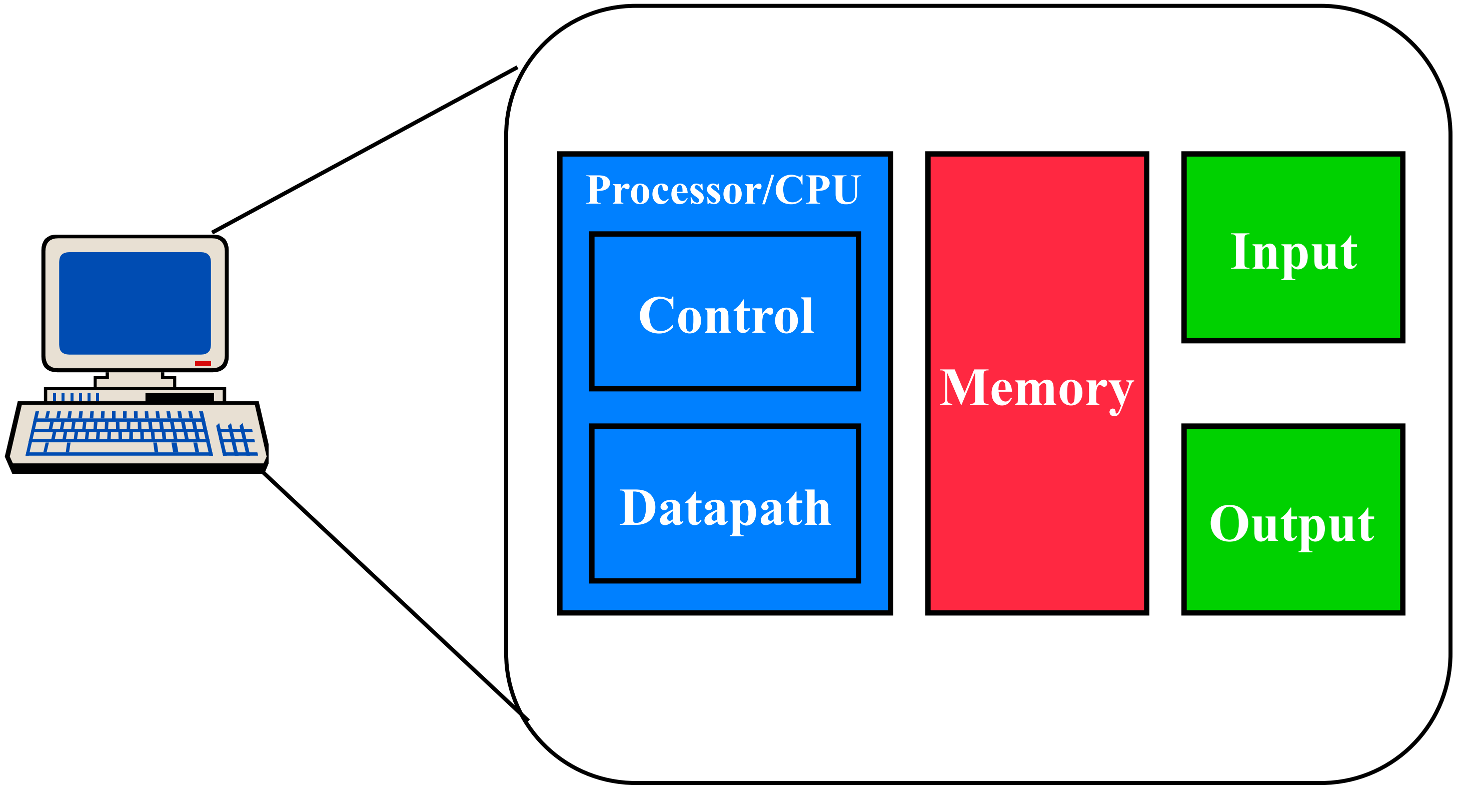
The Big Picture

- What is inside a computer?
- How does it execute a program?

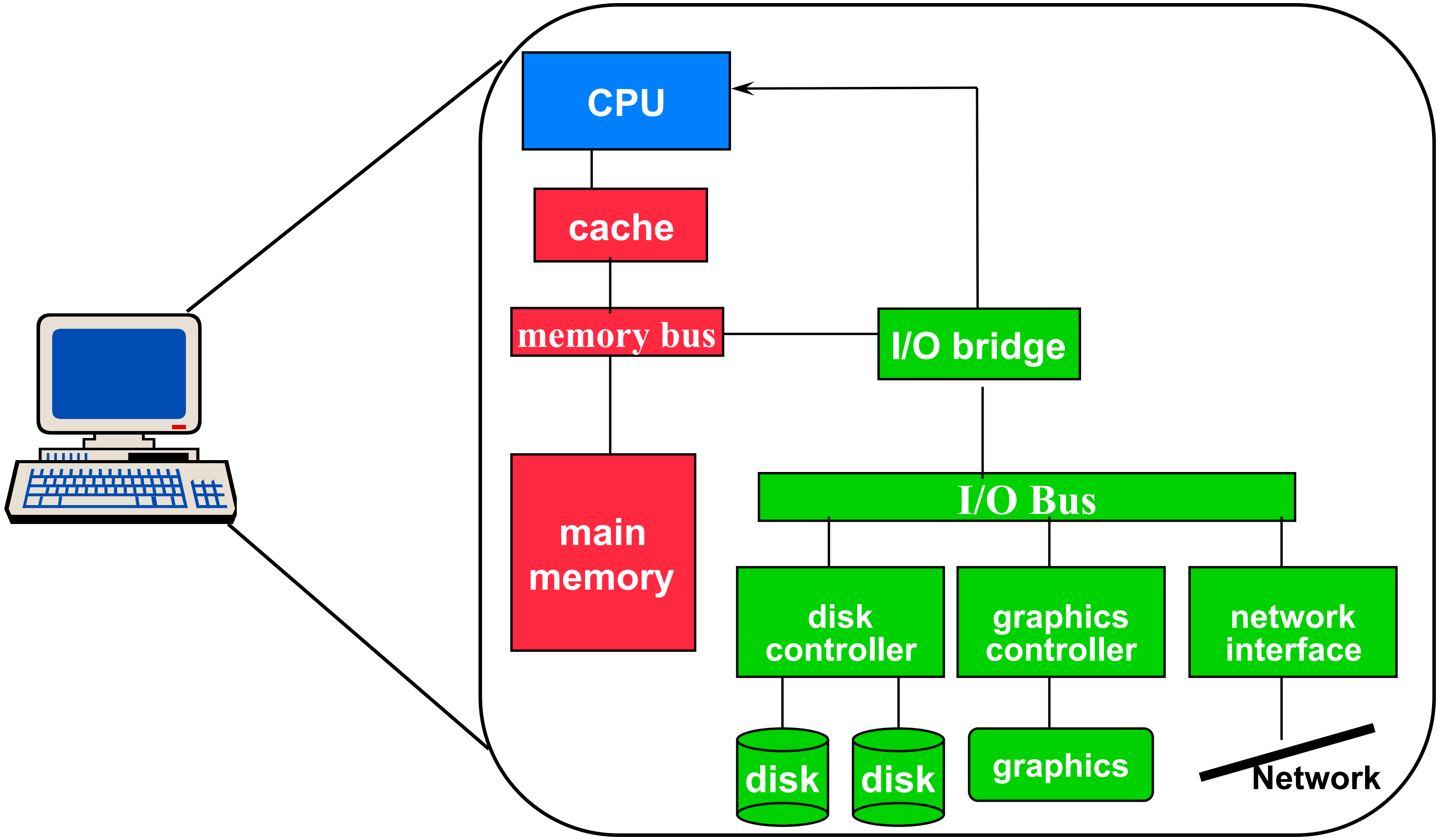


The Inside of a Computer

- The Five Classic Components of a Computer

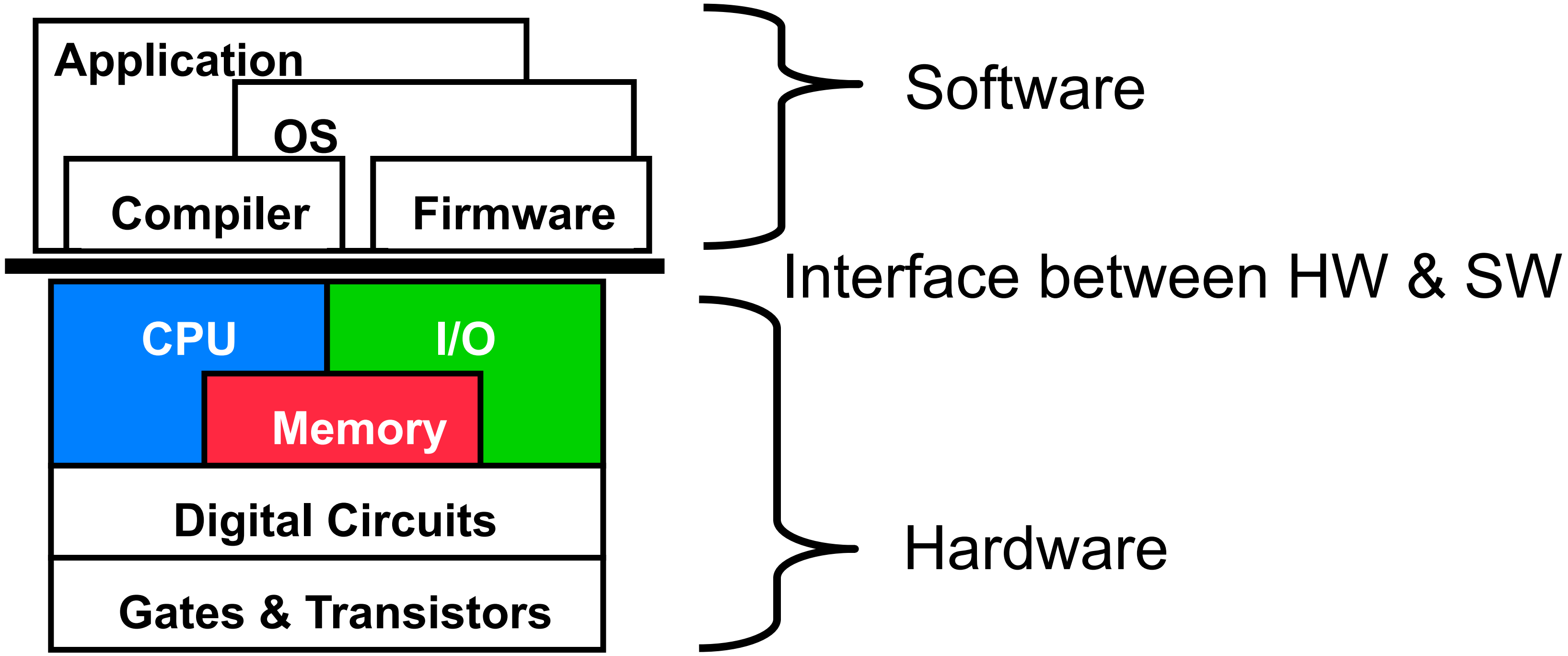


System Organization



What Is CS/ECE 250 All About?

- **Architecture = interface between hardware and software**



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

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

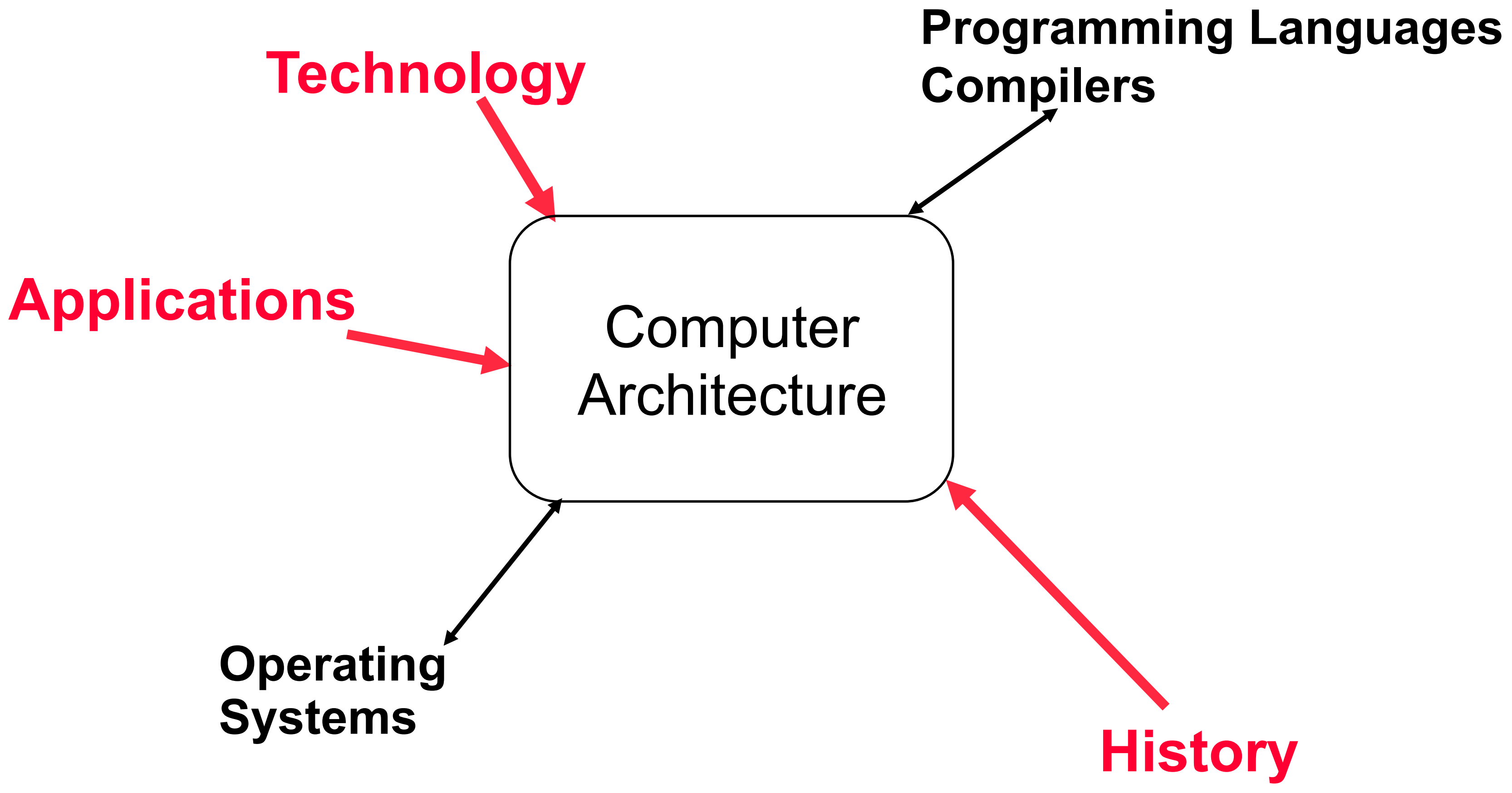
- High performance
  - E.g., NVIDIA Kepler
  - Gaming, Scientific Computing
- Others can achieve decent enough performance without using too much power
  - E.g., ARM (for mobile phones, tablets)
- And yet others can function reliably enough to be trusted with the control of your car's brakes or fly airplanes

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

Kinds of Computers

- “Traditional” personal computers
  - Laptop, desktop, netbook
- Less-traditional personal computers
  - iPad, iPhone, Blackberry, iPod, Xbox, Wii, etc.
- Hidden “big” computers (some are in the “cloud”)
  - Mainframes and servers for business, science, government
    - » E.g., the machines that run Duke email, ACES, etc.
  - Google/Facebook/Microsoft have many thousands of computers (that you don’t see)
- Hidden embedded computers
  - Controllers for cars, airplanes, ATMs, toasters, DVD players, etc.
  - Far and away the largest market for computers!
- Other kinds of computers??

Forces on Computer Architecture



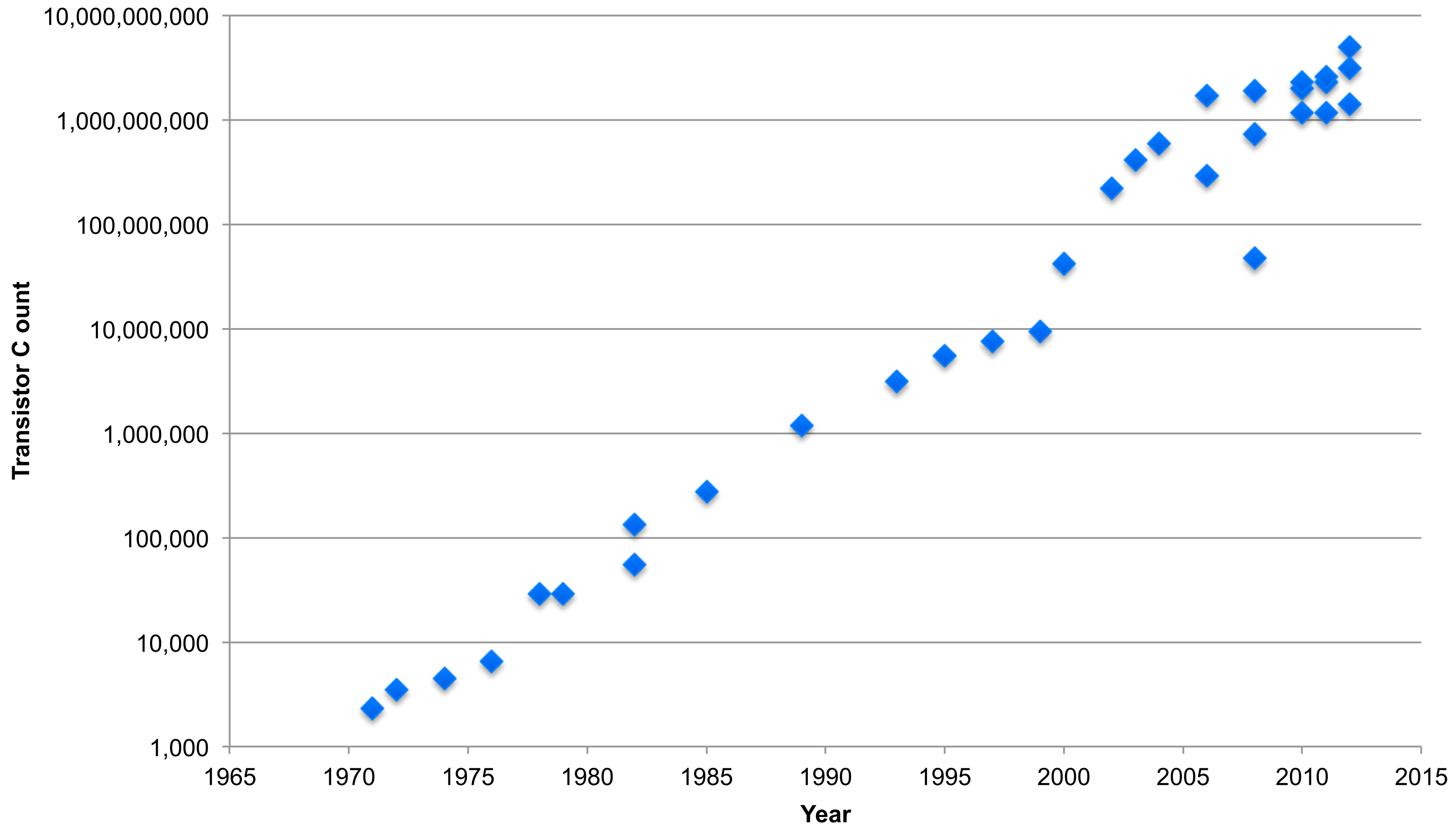
A Very Brief History of Computing

- 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)

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	166,000,000	$4,000,000
1981	IBM PC	1	240,000	$3,000
1991	HP 9000 / model 750	2	50,000,000	$7,4000
1996	Intel Ppro PC	2	400,000,000	$4,400
2005	Intel Pentium4	0.25-2	4,000,000,000	< $1,000
2013	Intel Corei7	0.25-2	17,00,000,000	$800 - $1,000

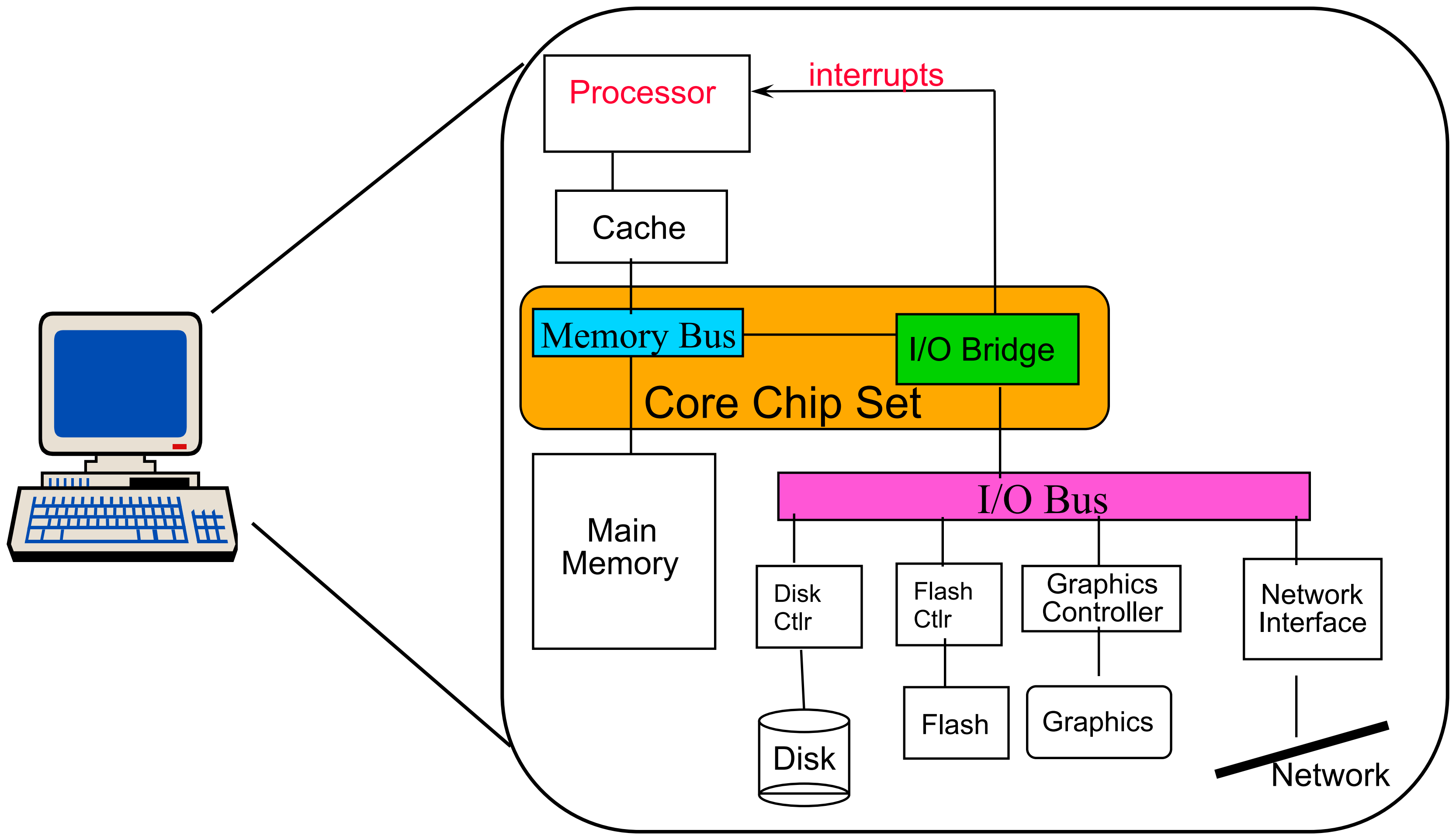
Microprocessor Transistor Trends (Intel CPUs)



What Do Computer Architects Do?

- Full disclosure: I'm a computer architect (but I do software systems and low-level technology too)
- Design new microarchitectures
  - Very occasionally, we design new architectures
- Designs that meet ever-changing needs and 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 and some science
  - There is no one right way to design a computer → this is why there isn't just one type of computer in the world
  - This does not mean, though, that all computers are equally good

System Organization



What You Will Learn In This Course

- The basic operation of a computer
  - Data representations
  - Primitive operations (instructions)
  - Computer arithmetic
  - Instruction sequencing and processing
  - Memory
  - Input/output
  - Doing all of the above, just faster!
- Understand the relationship between abstractions
  - Interface design
  - High-level program to control signals (SW → HW)
  - It's how we manage complexity!
- C programming → why?

Course Outline

- Introduction to Computer Architecture
- C programming language (next!)
- From C to Binary
- Instruction Sets & Assembly Programming
- Processor Core Design
- Memory Systems
- Input/Output
- Pipelined Processor Cores
- Multicore Processors

The Even Bigger Picture

- CS/ECE 250: Basic computer design
  - Finish 1 instruction every 1 very-long clock cycle
  - Finish 1 instruction every 1 short cycle (using pipelining)
- CS/ECE 350: Implementing digital computers/systems
- CS 550/ECE 552: 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.
- CS 650/ECE 652: Highly parallel computers and other advanced topics
- CS/ECE 554: Fault tolerant computers
- Various 590 instances: Topics / seminars