ECE/CS 250
Computer Architecture

Summer 2019

Introduction

Tyler Bletsch
Duke University

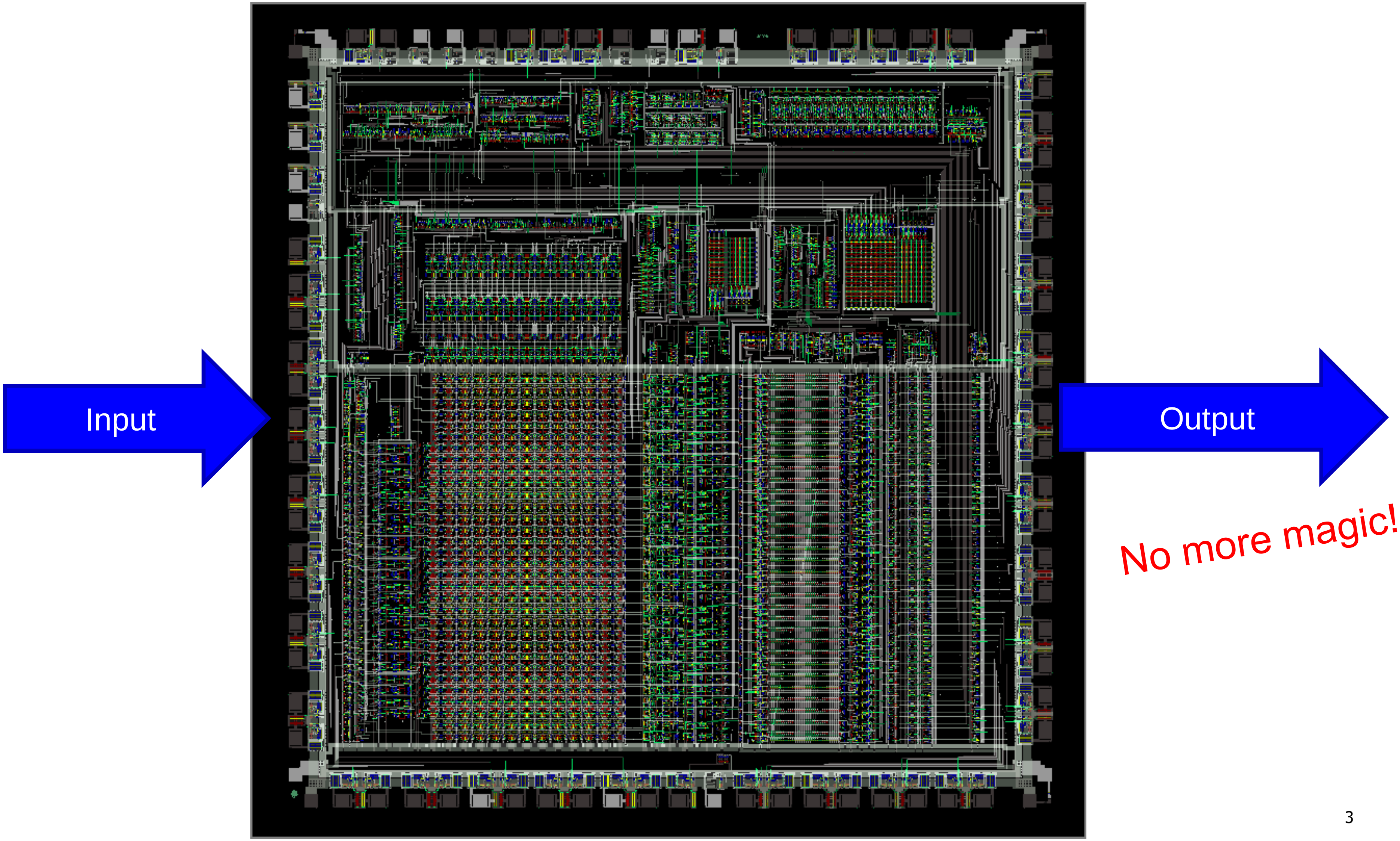
Slides are derived from work by
Daniel J. Sorin (Duke), Andrew Hilton (Duke), Alvy Lebeck (Duke),
Benjamin Lee (Duke), and Amir Roth (Penn)

Course objective: Evolve your understanding of computers

Before

Course objective: Evolve your understanding of computers

After



No more magic!

Instructor and TAs

- Professor: Tyler Bletsch
  - Office: Hudson Hall 106
  - Email: Tyler.Bletsch@duke.edu
  - Office Hours: see course site
- Undergraduate TAs

Undergrad Teaching Assistants

- Undergraduate TAs (UTAs)
  - Awesome undergrads who aced this class

- Will help with
  - Answering email questions about homeworks
  - Holding office hours to help with tools and software

- Will NOT bail you out at 3am

Getting Info

- **Course Web Page**: static info

http://people.duke.edu/~tkb13/courses/ece250/

  - Syllabus, schedule, slides, assignments, rules/policies, prof/TA info, office hour info
  - Links to useful resources

- **Piazza**: questions/answers
  - Post all of your questions here
  - Questions must be "public" unless good reason otherwise
  - **No code** in public posts!

- **Sakai**: just assignment submission and gradebook

Getting Answers to Questions

- What do you do if you have a question?

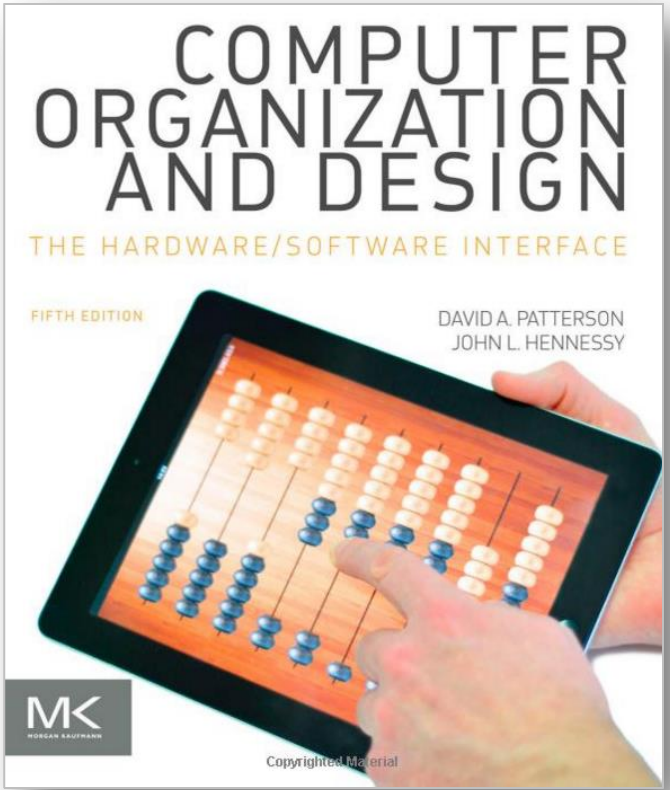
1. Check the course website
2. Check Piazza
   - If you have questions about homeworks, use Piazza – then everyone can see the answer(s) posted there by me, a TA, or your fellow classmate
   - Professor and TAs will NOT answer direct emails about homeworks or anything that pertains to more than 1 student

- Contact TA directly if: grading issue
- Contact professor directly if issue that is specific to you and that can't be posted on Piazza (e.g., missing exam)

Textbook

- Text: David A. Patterson and John L. Hennessy. Computer Organization and Design: The Hardware/Software Interface, 5th edition, Morgan-Kaufmann.
  - Not the "ARM edition" or "Revised Printing" or whatever

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



If you go to **addall.com**, you can search all online booksellers at once.
Amazon price for text: $48 used. Addall found it for $33.

Other Resources

- There are many online resources, including:
  - Unix tutorials
  - C programming tutorials
  - Videos of Prof. Sorin (Duke ECE/CS) teaching this course
  - Videos of Prof. Hilton (Duke ECE/CS) teaching C programming
  - Coursera course on computer architecture
  - Etc.
- Many useful links on course website
- Feel free to use these materials, but none are required

Workload

- 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**
  - During recitations, work on exercises to help you learn skills necessary for homeworks and exams. Can also get homework help once done
  - UTAs will help students during recitations
  - Bring a laptop to work on – if you don't have one, please with a partner who has one or contact me about getting a loaner

Lecture vs. Recitation

- Lecture:
  - Learning the theory the underlies computers
  - **Necessary to achieve understanding and do well in the course**
  - Attendance expected but not tracked
- Recitation:
  - Learning practical skills needed to understand and design computers
  - **Necessary to achieve understanding and do well in the course**
  - Attendance required. Grading:
    - Students attending and making *good faith effort* will receive full credit for the day.

Lecture vs. Recitation

- If you attend lecture but not recitation:
  - You won't know how to *do* the assignment

Lecture vs. Recitation

- If you attend recitation but not lecture:
  - You won't know <u>*how*</u> to do the assignment

Lecture vs. Recitation

- If you attend recitation AND lecture:
  - Your hands will turn into creepy robot hands but you'll probably get a good grade

Grading Breakdown

HW0!

Assignment	%
Homeworks	55%
Recitation	5%
Midterm Exam	15%
Final Exam	25%

Partial credit is available – provide detail in your answers to seek it!

Late homework submissions incur penalties as follows:

- Submission is 0-24 hours late: total score is multiplied by 0.9
- Submission is 24-48 hours late: total score is multiplied by 0.8
- Submission is more than 48 hours late: total score is multiplied by the Planck constant (in J·s)

$\sim 6.6 \times 10^{-34}$

NOTE: If you feel *in advance* that you may need an extension, contact the instructor.

These assignments are looooooooooong. START EARLY.

Homework Zero

- Due Friday night
- Designed to get you familiar with UNIX in general and Linux in particular
- UNIX skills are for more than this course – there’s a **reason** people use these tools!
- If you’re having trouble, post on Piazza and we can help you.

Grade Appeals

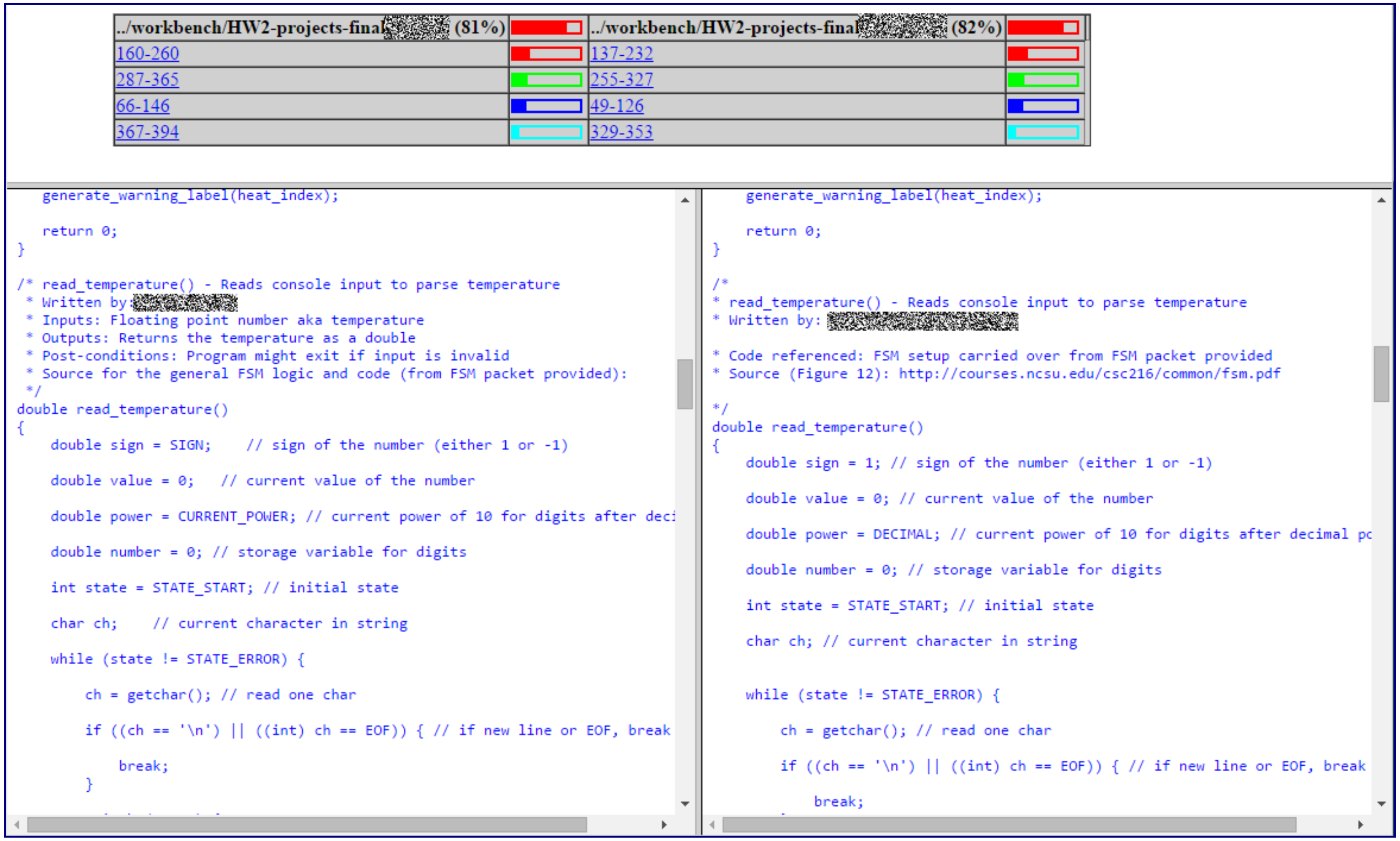
- All regrade requests must be in writing
  - Email the UTA who graded the question (we'll indicate who graded what)
- After speaking with the TA, if you still have concerns, contact the instructor
- All regrade requests must be submitted no later than 1 week after the assignment was returned to you.

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 the 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
  - I've referred over a dozen cases to the Office of Student Conduct; don't be one of them!

- "But I didn't know that was cheating" is not a valid excuse

MOSS:
Measure of Software Similarity



../workbench/HW2-projects-final[redacted] (81%)		../workbench/HW2-projects-final[redacted] (82%)	
160-260		137-232	
287-365		255-327	
66-146		49-126	
367-394		329-353	

```
  generate_warning_label(heat_index);

  return 0;
}

/* read_temperature() - Reads console input to parse temperature
 * Written by:[redacted]
 * Inputs: Floating point number aka temperature
 * Outputs: Returns the temperature as a double
 * Post-conditions: Program might exit if input is invalid
 * Source for the general FSM logic and code (from FSM packet provided):
 */
double read_temperature()
{
    double sign = SIGN;    // sign of the number (either 1 or -1)

    double value = 0;   // current value of the number

    double power = CURRENT_POWER; // current power of 10 for digits after deci

    double number = 0; // storage variable for digits

    int state = STATE_START; // initial state

    char ch;    // current character in string

    while (state != STATE_ERROR) {

        ch = getchar(); // read one char

        if ((ch == '\n') || ((int) ch == EOF)) { // if new line or EOF, break

            break;
        }
```

```
  generate_warning_label(heat_index);

  return 0;
}

/*
* read_temperature() - Reads console input to parse temperature
* Written by: [redacted]

* Code referenced: FSM setup carried over from FSM packet provided
* Source (Figure 12): http://courses.ncsu.edu/csc216/common/fsm.pdf

*/
double read_temperature()
{
    double sign = 1; // sign of the number (either 1 or -1)

    double value = 0; // current value of the number

    double power = DECIMAL; // current power of 10 for digits after decimal pc

    double number = 0; // storage variable for digits

    int state = STATE_START; // initial state

    char ch; // current character in string

    while (state != STATE_ERROR) {

        ch = getchar(); // read one char

        if ((ch == '\n') || ((int) ch == EOF)) { // if new line or EOF, break

            break;
```

Doesn’t care about:
- Comments
- Whitespace
- Naming
- Values

Only cares about code structure.

<u>How to beat it?</u>
Write your own code

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 know how to program in C
- If, at any point, it's not clear why I'm talking about some topic, please ask!

Our Responsibilities

- The instructor and TA will…
  - Provide lectures/recitations at the stated times
  - Set clear policies on grading
  - Provide timely feedback on assignments
  - Be available out of class to provide reasonable assistance
  - Respond to comments or complaints about the instruction provided

- Students are expected to…
  - Receive lectures/recitations at the stated times
  - Turn in assignments on time
  - Seek out of class assistance in a timely manner if needed
  - Provide frank comments about the instruction or grading as soon as possible if there are issues
  - Assist each other *within the bounds of academic integrity*

Advice from past students

I have to tell you about the future!!

From a survey given at the end of past semesters. Unedited. Spring 2016:

- "Start every homework as early as possible. It is almost impossible to judge how long they are going to take until you are a decent way through them. By starting early, you'll relieve some stress, and be able to approach teachers/TA's with tangible progress and questions early on. This makes the whole assignment go faster."
- "Don't be intimidated if the information makes no sense at first. Be patient, use your resources, and take your time. It'll make sense. Going hand in hand with this, if you're going to take your time, start early on assignments to make sure you have that time."

From Summer 2017:

- "Don't complain, it will only make things worse. The course is actually very well-designed, most people who have trouble (myself included the first time I took this course #anonymous) have problems because they're idiots at managing time. Don't think you're a genius and wait until the last minute - just do your work until it's done."
- "Enjoy your work. Get enough sleep. It's a fun class - if it's not, you're probably doing something wrong."

From Fall 2017:

- "Start the homework EARLY! If you do this, you will really enjoy the course, as it covers some awesome material. Moreover, you will be impressed with yourself. You actually make a CPU from scratch... If you start the homework at the last second, you will surely not have a good experience."
- "Start the assignment EARLY and ASK for help EARLY. There is nothing worse than hopelessly waiting for someone to respond to your question in Piazza the night the assignment is due."

From Summer 2018:

- "Taking CS250 in the summer was very challenging but also very rewarding. One critical piece of advice is to **start all of the assignments on the day that they are released**. This will ensure that you don't have any 11:54 submission headaches :). Also, I would say to **reach out and ask for help as much as possible**. ..."
- "I would definitely say **start early** and **get help**. The projects in this class aren't projects you can bang out a few days before or with an all nightery. They need a lot of time and thinking so its best to approach it by working piece or concept at a time. Otherwise you'll fall behind. Also asking for help is a huge part of the class. In this class you can quickly fall behind in the material if you don't understand what's going on, so anything that confuses you should clear up quickly."

See all advice, unedited and raw, here: https://tinyurl.com/ya7mkhmy

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 machine that follows simple instructions deterministically.

- It just does what software tells it to do
  - Software is a series of these **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 whatever the cool kids are using these days?

Instruction Sets

- Computers can only execute instructions that are in their specific machine language
- Every type of computer has a different instruction set that it understands
  - Intel (and AMD) IA-32 (x86): Pentium Core i7, AMD Ryzen, etc.
  - ARM: In many embedded processors (e.g., smartphones)
    - Used by many companies (e.g., Qualcomm)
  - Intel IA-64: Itanium, Itanium 2
  - PowerPC: In Cell Processor (incl. Sony PS3) and old Apple Macs
  - SPARC: In computers from Sun Microsystems/Oracle
  - MIPS: MIPS R10000 → this is the example used in the textbook
- Note: no computer executes Java or C++

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?

Computer Architecture

- **Computer architecture** specifies what the hardware looks like (its interface), so that we can write software to run on it
  - What instructions does it have?
  - Number of storage locations it has?
  - More stuff (covered later…)
- **Important point:** there are many, many different ways to build machines that provide the 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
- What's inside one of these machines?

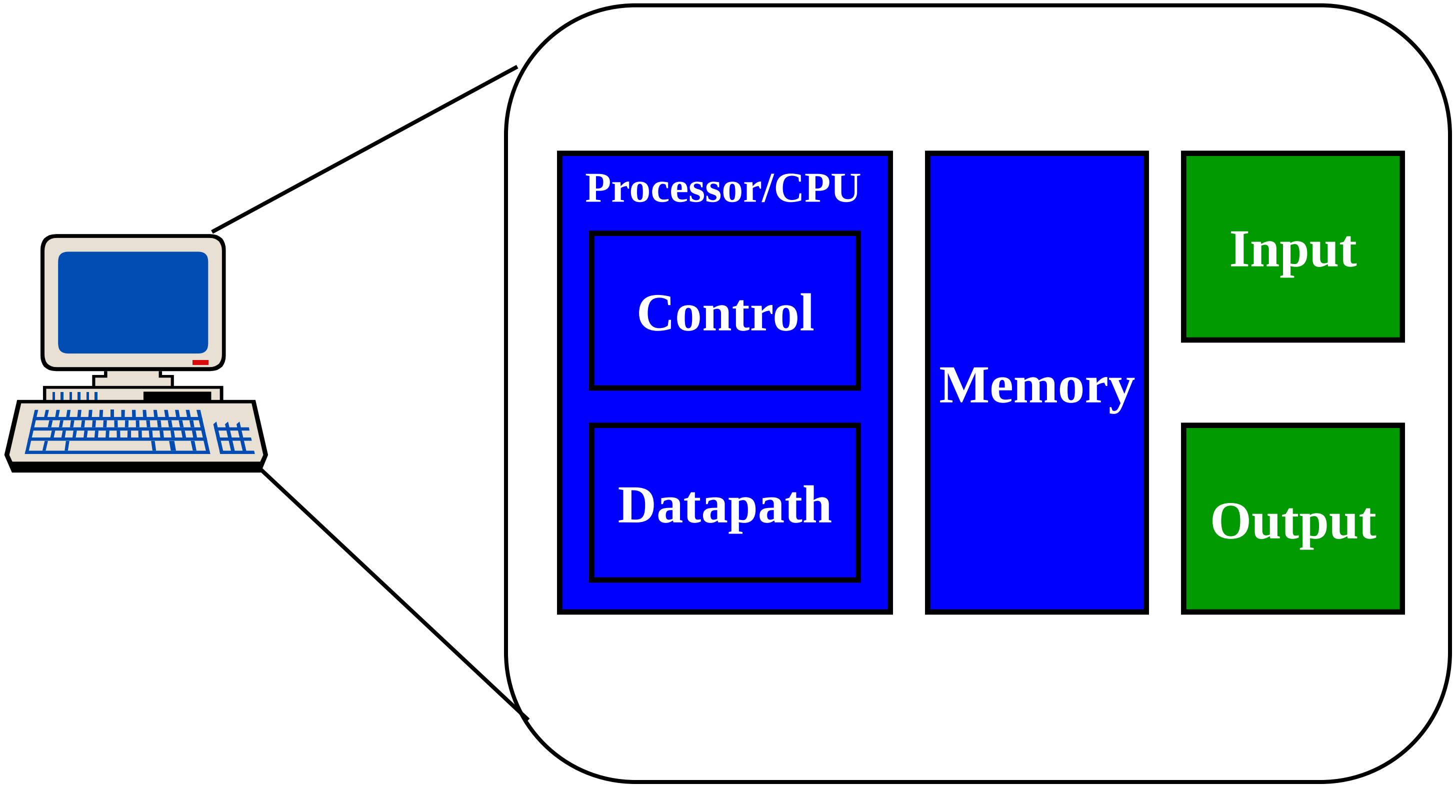
All computers are like fast food restaurants

- Fast Food **Architecture**: the interface
  - Menu
  - How/where to place orders
  - How finished orders are given to customers

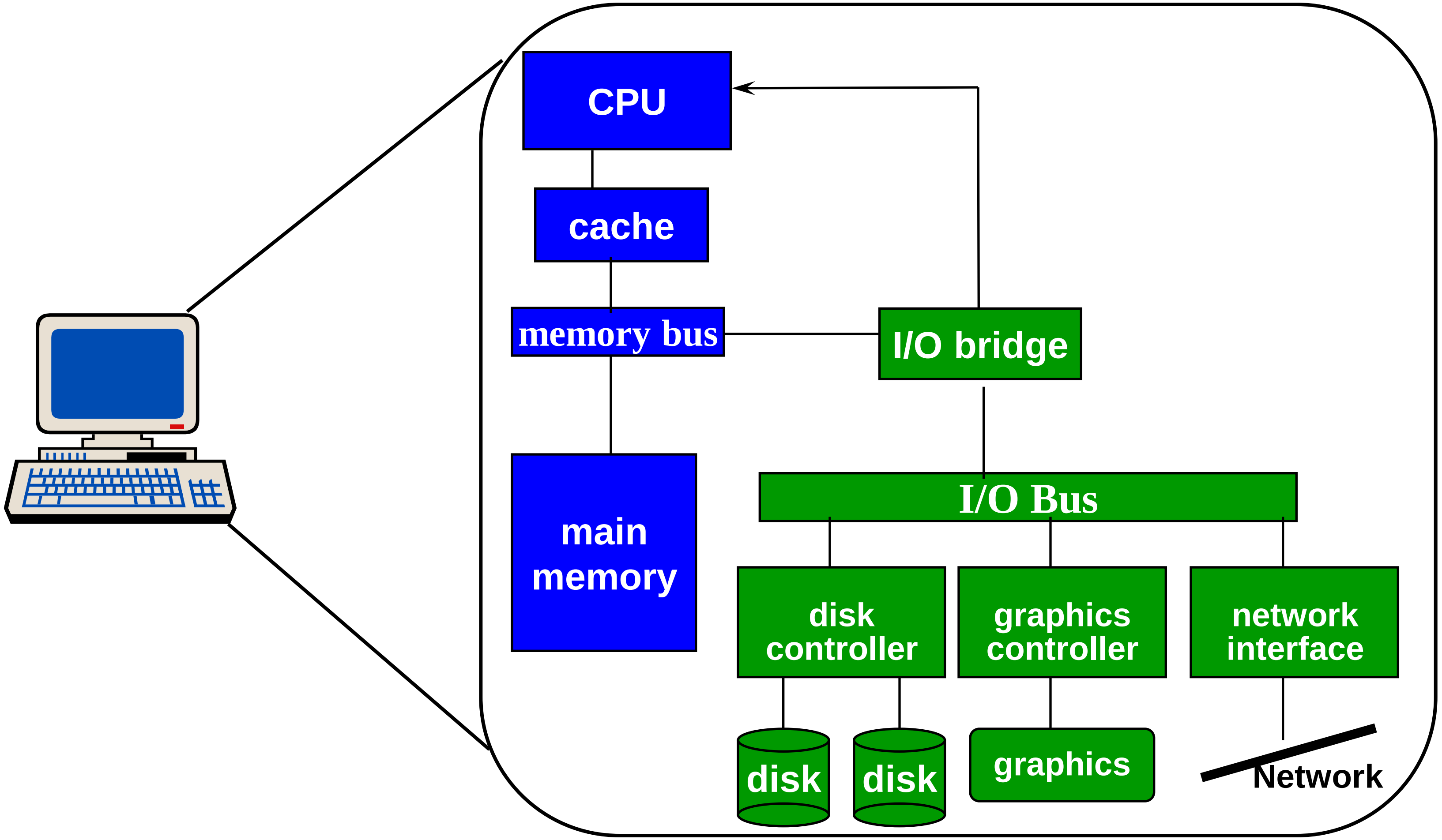
- Fast Food **Microarchitecture**: the implementation
  - What ingredients are used
  - What appliances are available
  - How many employees you have and what they do

The Inside of a Computer

- The Five Classic Components of a Computer

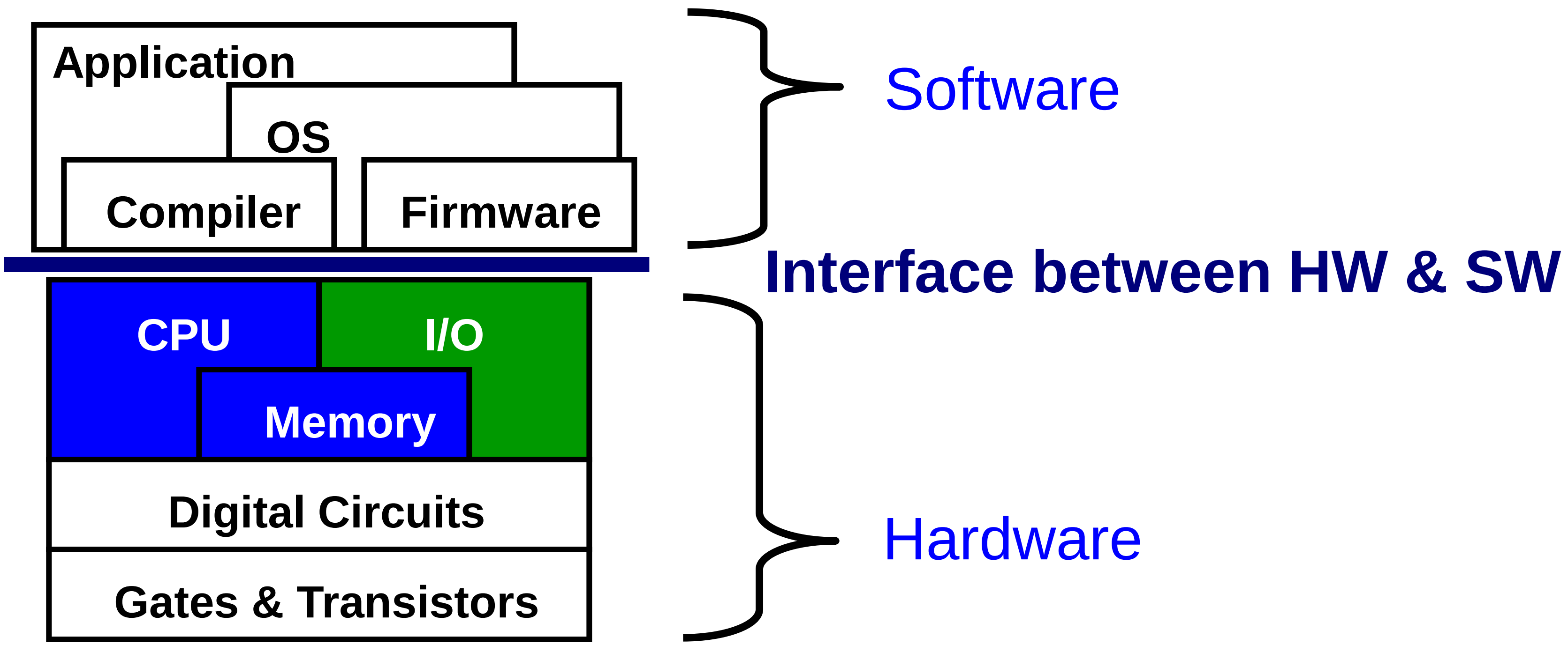


System Organization



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 for high-performance gaming
  - E.g., Hybrid CPU/GPU in XBox One and PS4
- Some for power-efficiency at acceptable performance (laptop)
  - E.g., Intel Pentium M (for Mobile)
- Some for extreme reliability
  - E.g., the CPU that runs your car’s brakes

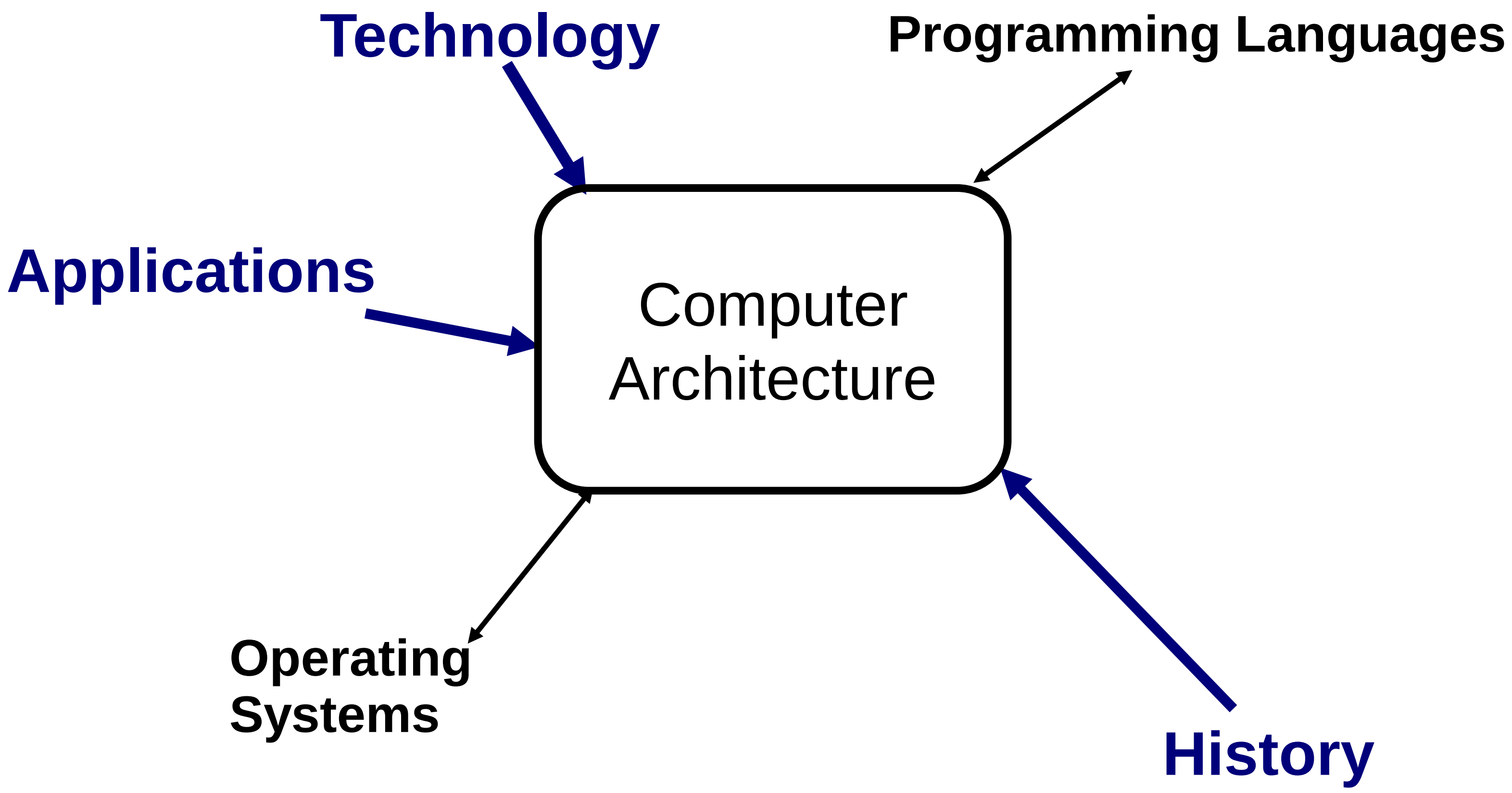
What computers do you use?

Which of those computers do you own?

Kinds of Computers

- "Traditional" personal computers
    - Laptop, desktop, netbook
- Less-traditional personal computers
    - iPad, iPhone, Samsung/Android smartphone, iPod, Xbox, 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 has 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



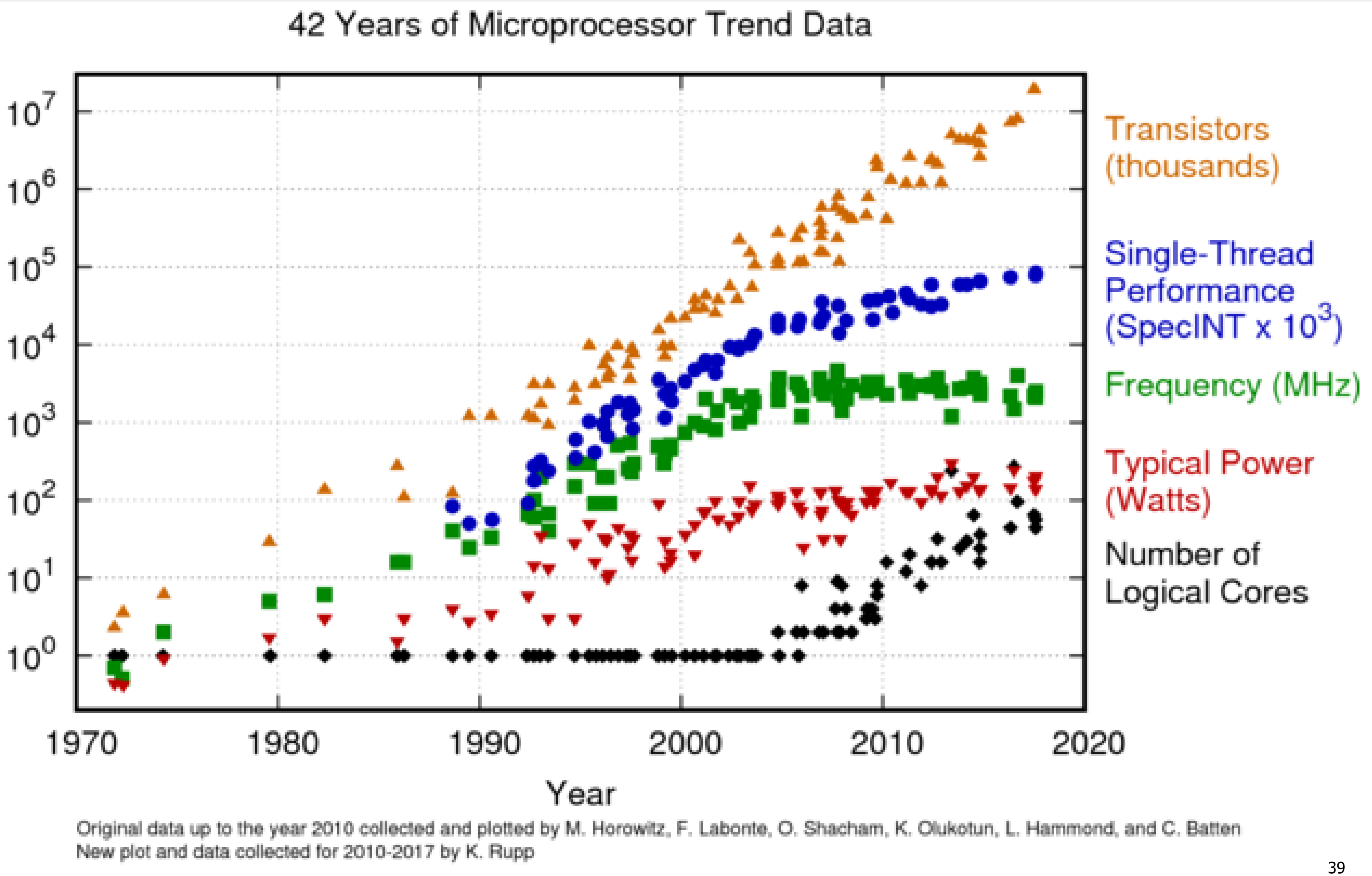
A Very Brief History of Computing

- 1645 Blaise Pascal’s Calculating Machine
- 1822 Charles Babbage
  - Difference Engine
  - Analytic Engine: Augusta Ada King (Lovelace), 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)

Some 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

Microprocessor Trends (for Intel CPUs)



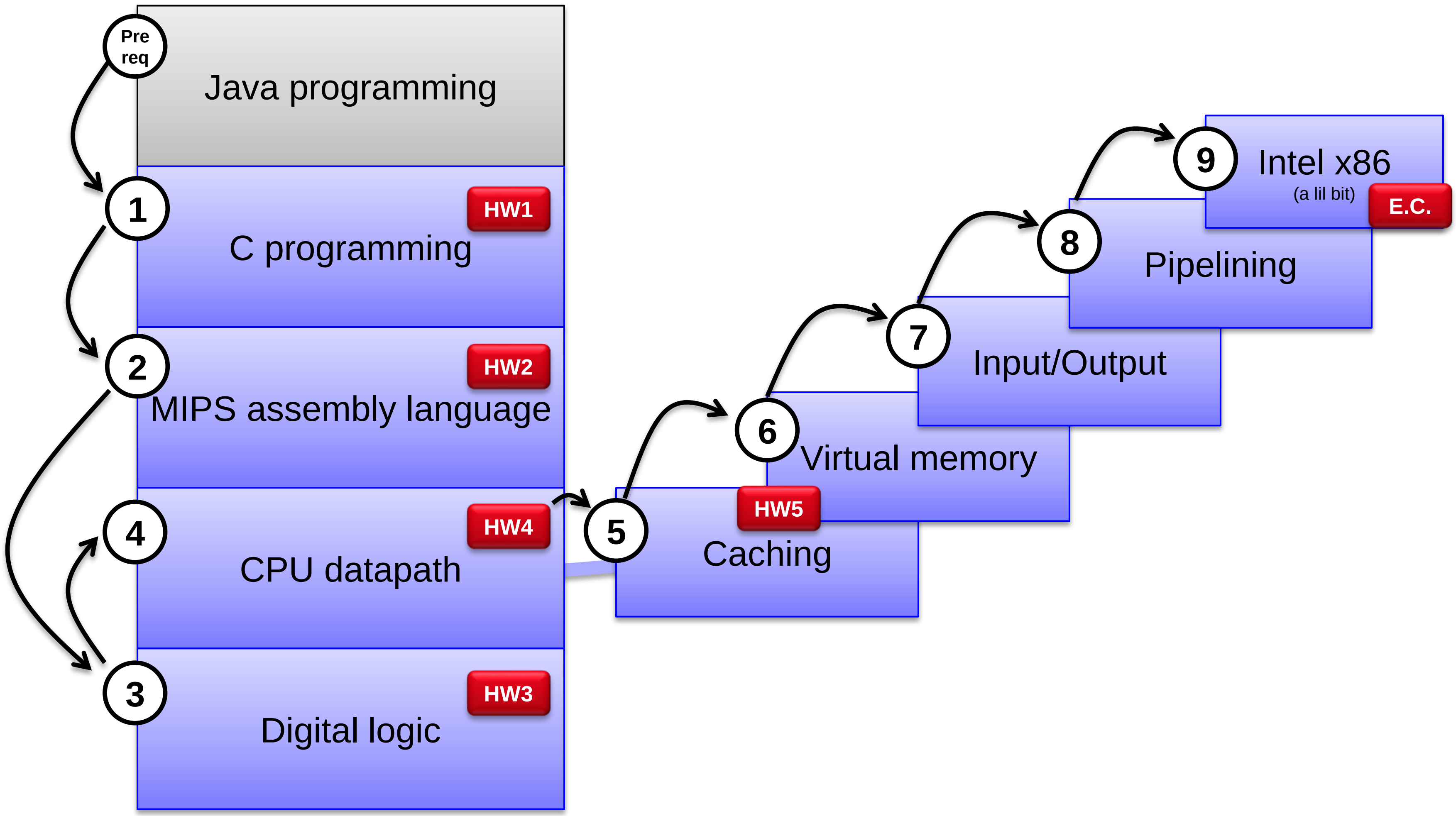
What Do Computer Architects Do?

- Design new microarchitectures
  - Very occasionally, we design new architectures
- Design computers 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, not 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

What You Will Learn In This Course

- The basic operation of a computer
  - 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)
- C programming → why?

Course Outline



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
- ECE 554/CS ???: Fault tolerant computers