

18-447

Computer Architecture

Lecture 1: Introduction and Basics

Prof. Onur Mutlu

Carnegie Mellon University

Spring 2012, 1/18/2012

I Hope You Are Here for This

18-213/243

- How does an assembly program end up executing as digital logic?
- **What happens in-between?**
- How is a computer designed using logic gates and wires to satisfy specific goals?

18-240

“C” as a model of computation

Programmer’s view of a computer system works

*Architect/microarchitect’s view:
How to design a computer that meets system design goals.
Choices critically affect both the SW programmer and the HW designer*

HW designer’s view of a computer system works

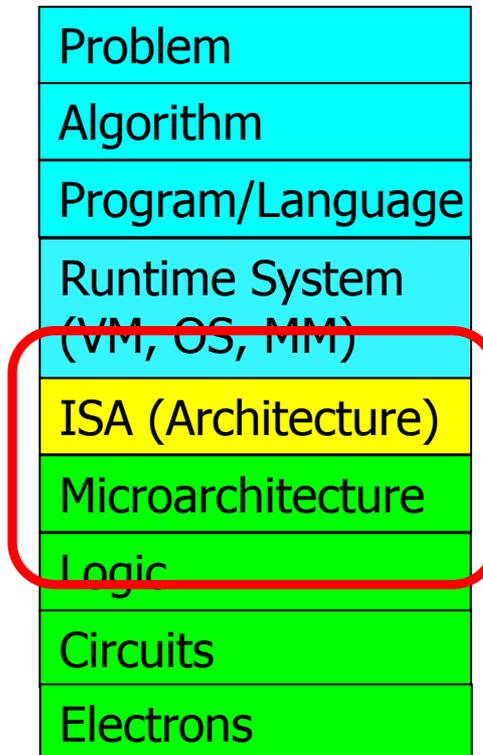
Digital logic as a model of computation

Levels of Transformation

“The purpose of computing is insight” (*Richard Hamming*)

We gain and generate insight by solving problems

How do we ensure problems are solved by electrons?



The Power of Abstraction

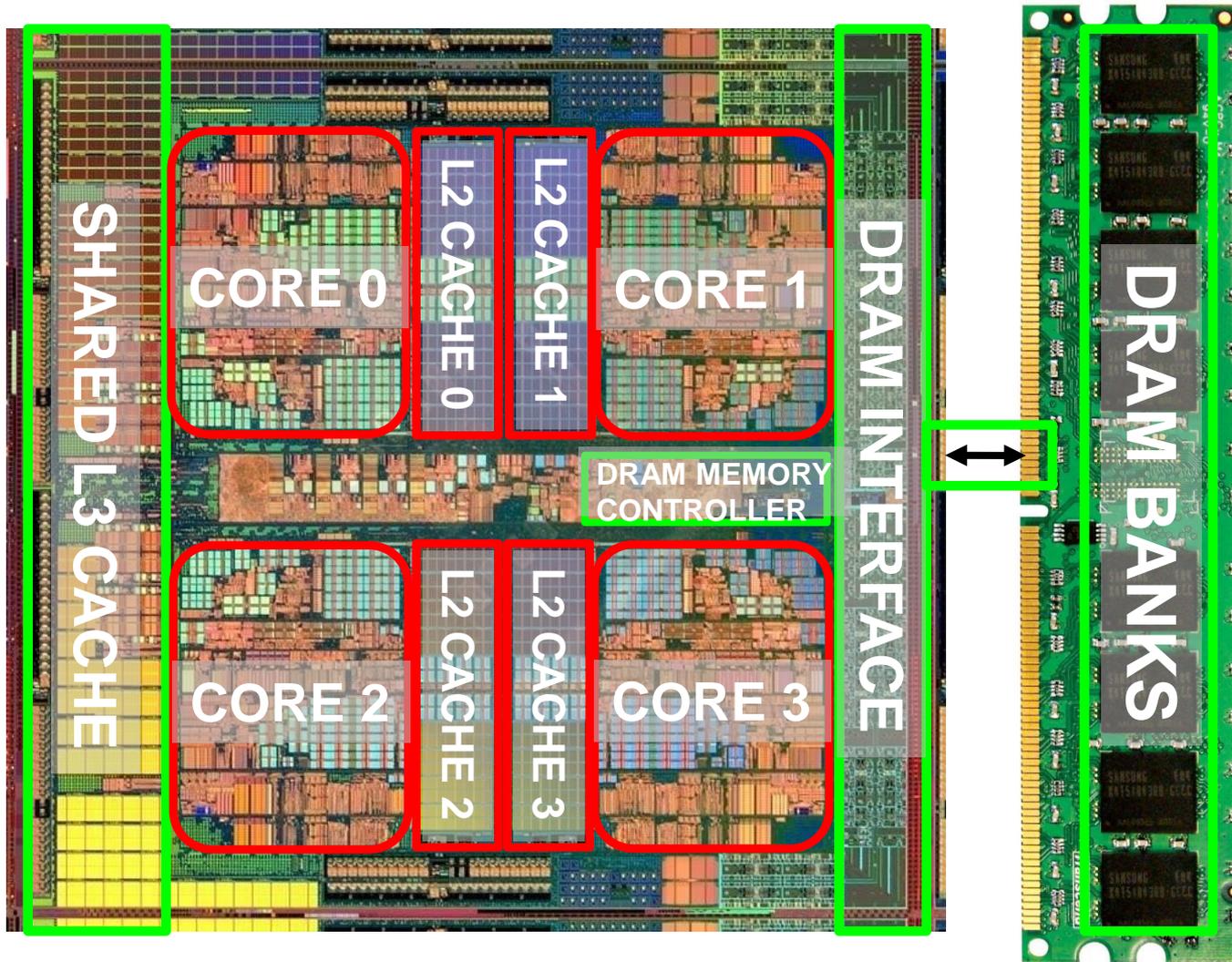
- **Levels of transformation create abstractions**
 - Abstraction: A higher level only needs to know about the interface to the lower level, not how the lower level is implemented
 - E.g., programmer does not really need to know what the ISA is and how a computer executes instructions
- **Abstraction improves productivity**
 - No need to worry about decisions made in underlying levels
 - E.g., programming in Java vs. C vs. assembly vs. binary vs. by specifying control signals of each transistor every cycle
- Then, why would you want to know what goes on underneath or above?

Crossing the Abstraction Layers

- As long as everything goes well, not knowing what happens in the underlying level (or above) is not a problem.
 - What if
 - The program you wrote is running slow?
 - The program you wrote does not run correctly?
 - The program you wrote consumes too much energy?
 - What if
 - The hardware you designed is too hard to program?
 - The hardware you designed is too slow b/c it does not provide the right primitives to the software?
 - One goal of this course is to understand how a processor works underneath the software layer and how decisions made in hardware affect the software/programmer
-

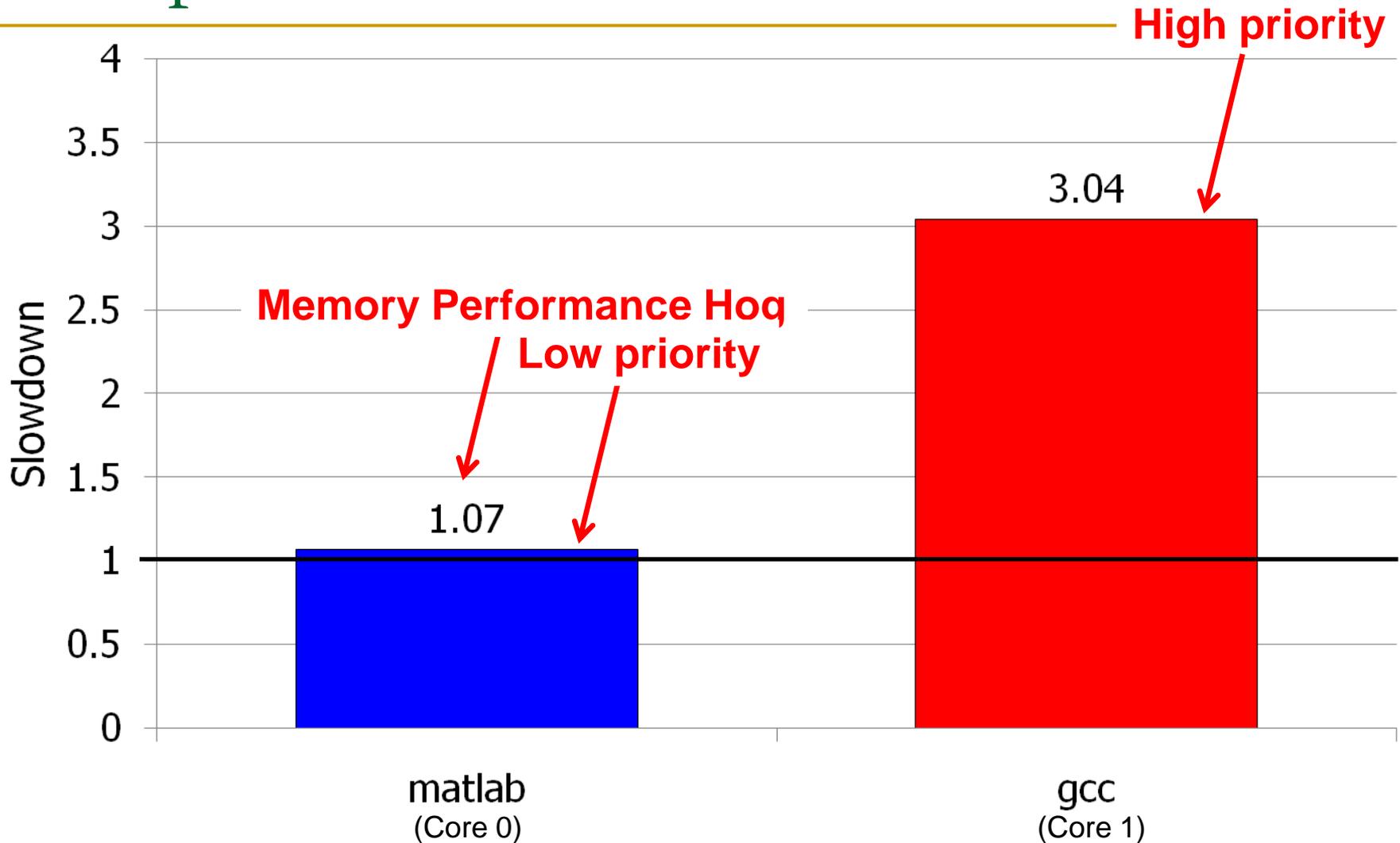
An Example: Multi-Core Systems

Multi-Core
Chip



*Die photo credit: AMD Barcelona

Unexpected Slowdowns in Multi-Core

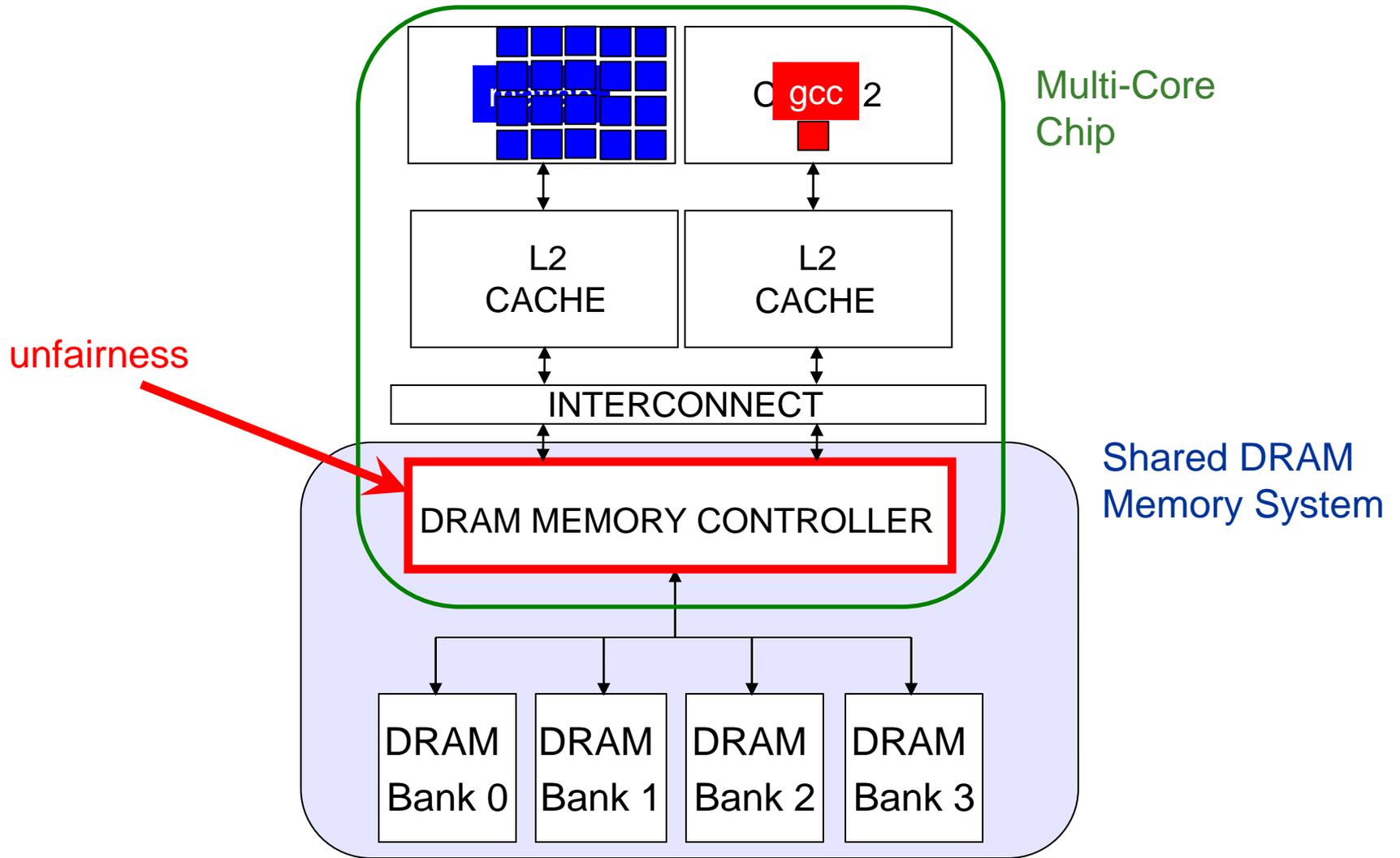


Moscibroda and Mutlu, "Memory performance attacks: Denial of memory service in multi-core systems," USENIX Security 2007.

A Question or Two

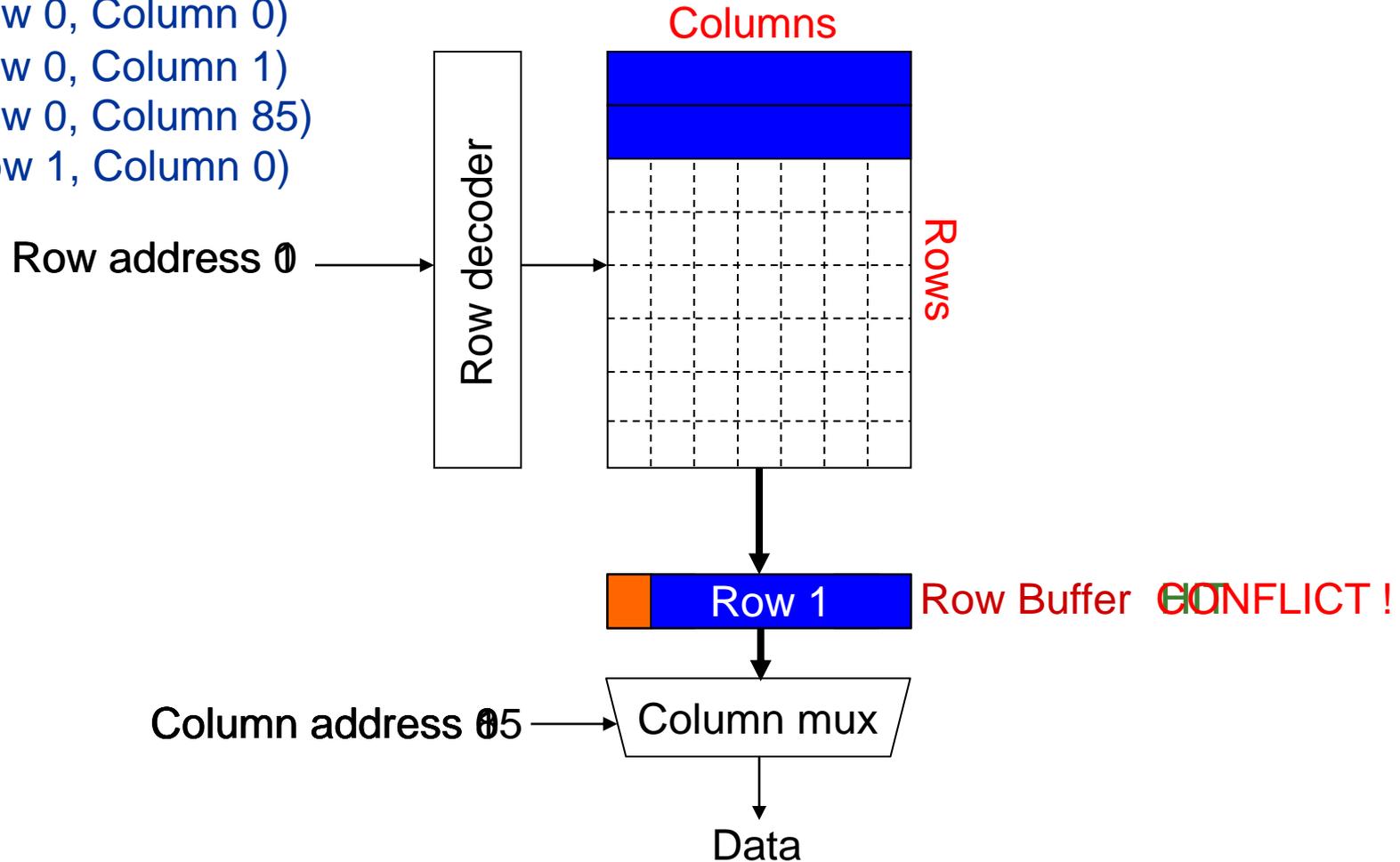
- Can you figure out why there is a disparity in slowdowns if you do not know how the processor executes the programs?
- Can you fix the problem without knowing what is happening “underneath”?

Why the Disparity in Slowdowns?



DRAM Bank Operation

Access Address:
(Row 0, Column 0)
(Row 0, Column 1)
(Row 0, Column 85)
(Row 1, Column 0)



DRAM Controllers

- A row-conflict memory access takes significantly longer than a row-hit access
- Current controllers take advantage of the row buffer
- Commonly used scheduling policy (FR-FCFS) [Rixner 2000]*
 - (1) Row-hit first: Service row-hit memory accesses first
 - (2) Oldest-first: Then service older accesses first
- This scheduling policy aims to maximize DRAM throughput

*Rixner et al., “Memory Access Scheduling,” ISCA 2000.

*Zuravleff and Robinson, “Controller for a synchronous DRAM ...,” US Patent 5,630,096, May 1997.

The Problem

- Multiple threads share the DRAM controller
- DRAM controllers designed to maximize DRAM throughput
- DRAM scheduling policies are thread-unfair
 - Row-hit first: unfairly prioritizes threads with high row buffer locality
 - Threads that keep on accessing the same row
 - Oldest-first: unfairly prioritizes memory-intensive threads
- DRAM controller vulnerable to denial of service attacks
 - Can write programs to exploit unfairness

Now That We Know What Happens Underneath

- How would you solve the problem?

Agenda

- Intro to 18-447
 - Course logistics, info, requirements
 - What 447 is about
 - Lab assignments
 - Homeworks

- Assignments for the next two weeks
 - Homework 0 (due Jan 23)
 - Homework 1 (due Jan 30)
 - Lab 1 (due Feb 3)

- Basic concepts in computer architecture

Handouts

- Make sure you get a copy of
 - The Syllabus
 - Homework 0

Course Info: Who Are We?

- Instructor: Prof. Onur Mutlu

- onur@cmu.edu

- Office: Hamerschlag Hall A305

- Office Hours: W 2:30-3:30pm (or by appointment)

- <http://www.ece.cmu.edu/~omutlu>

- PhD from UT-Austin, worked at Microsoft Research, Intel, AMD

- Research and teaching interests:

- Computer architecture, hardware/software interaction

- Many-core systems

- Memory and storage systems

- Improving programmer productivity

- Interconnection networks

- Hardware/software interaction and co-design (PL, OS, Architecture)

- Fault tolerance

- Hardware security

- Algorithms and architectures for genomics and embedded systems



Course Info: Who Are We?

■ Teaching Assistants

□ Chris Fallin

- cfallin@ece.cmu.edu
- Office hours: Fri 2-4pm



□ Lavanya Subramanian

- lsubrama@ece.cmu.edu
- Office hours: Thu 11:30am-1:30pm



□ Abeer Agrawal

- abeera@andrew.cmu.edu
- Office hours: Mon 3:30-4:30pm, Tue 6-7pm



Your Turn

- Who are you?
- Homework 0
 - Your opportunity to tell us about yourself
 - Due Jan 23, before class
 - Return hard copy
 - Attach your picture
- All grading predicated on receipt of Homework 0

Where to Get Up-to-date Course Info?

- Website: <http://www.ece.cmu.edu/~ece447>
 - Lecture notes
 - Project information
 - Homeworks
 - Course schedule, handouts, papers, FAQs
- Blackboard
 - Upload homeworks here except for HW0
- Your email
- Me and the TAs

Lecture and Lab Locations, Times

■ Lectures:

- ❑ MW 12:30-2:20pm
- ❑ Hamerschlag Hall B103
- ❑ Attendance is for your benefit to learn
- ❑ Some days, we might have discussion sessions

■ Labs:

- ❑ T 10:30am-1:20pm
- ❑ Th 1:30-4:20pm
- ❑ F 6:30-9:20pm
- ❑ Hamerschlag Hall 1112
- ❑ Attendance not required except for check-off of labs
- ❑ You can attend any lab session to ask questions or check off

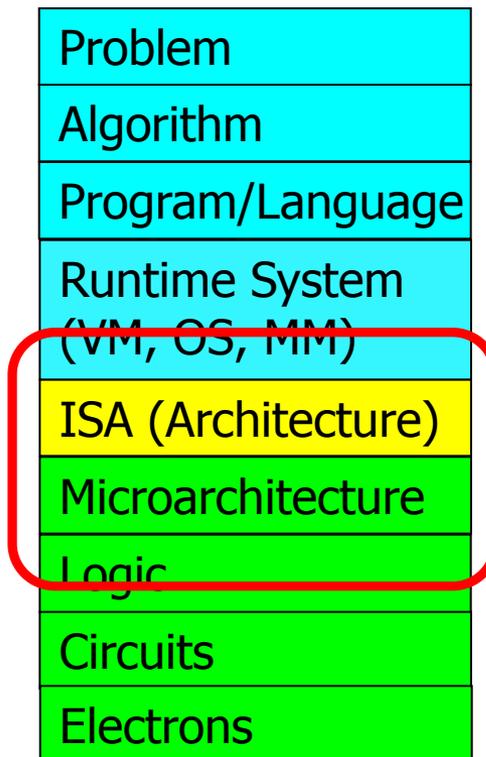
Tentative Course Schedule

- Tentative schedule is in syllabus
- To get an idea of topics, you can look at older versions:
 - www.ece.cmu.edu/~ece447/old
- But don't believe the "static" schedule
- Systems that perform best are usually dynamically scheduled
 - Static vs. Dynamic scheduling
 - Compile time vs. Run time

What Will You Learn

- **Computer Architecture:** The science and art of designing, selecting, and interconnecting hardware components and designing the hardware/software interface to create a computing system that meets functional, performance, energy consumption, cost, and other specific goals.
- **Traditional definition:** “The term *architecture* is used here to describe the attributes of a system as seen by the programmer, i.e., the conceptual structure and functional behavior as distinct from the organization of the dataflow and controls, the logic design, and the physical implementation.” *Gene Amdahl*, IBM Journal of R&D, April 1964

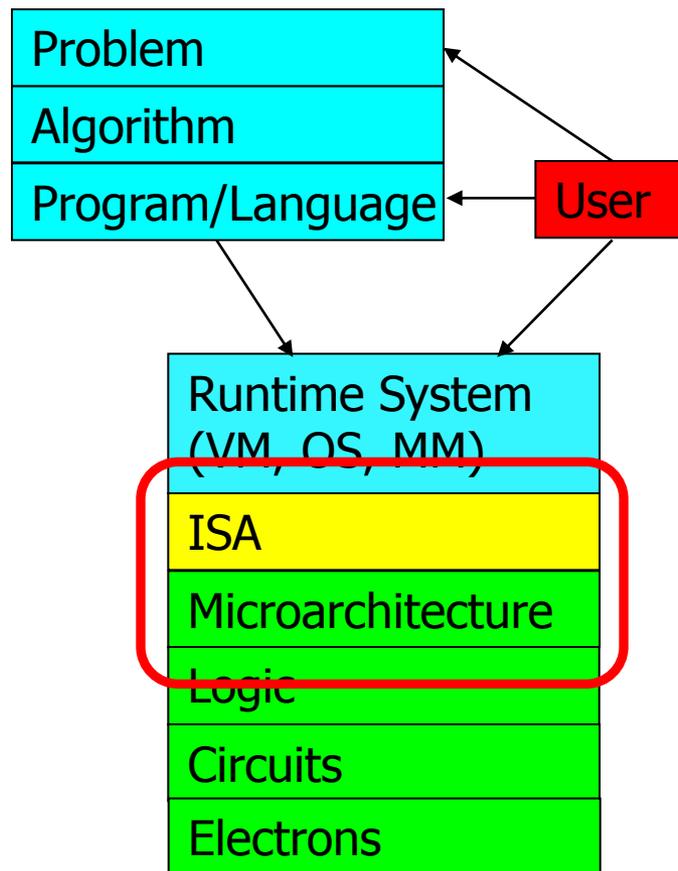
Computer Architecture in Levels of Transformation



- Read: Patt, "Requirements, Bottlenecks, and Good Fortune: Agents for Microprocessor Evolution," Proceedings of the IEEE 2001.

Levels of Transformation, Revisited

- A user-centric view: computer designed for users



- The stack should be optimized for user

What Will You Learn?

- Fundamental principles and tradeoffs in designing the hardware/software interface and major components of a modern programmable microprocessor
 - Focus on state-of-the-art (740 has more research focus)
 - Trade-offs and how to make them
- How to design, implement, and evaluate a functional modern processor
 - Semester-long lab assignments
 - A combination of RTL implementation and higher-level simulation
 - Focus is on functionality (740 has a focus on “doing better”)
- How to dig out information
- How to work even harder!

Course Goals

- Goal 1: To familiarize those interested in computer system design with both **fundamental operation principles and design tradeoffs** of **processor, memory, and platform architectures** in today's systems.
 - Strong emphasis on fundamentals and design tradeoffs.
- Goal 2: To provide the necessary **background and experience to design, implement, and evaluate a modern processor** by **performing hands-on RTL and C-level implementation**.
 - Strong emphasis on functionality and hands-on design.

A Note on Hardware vs. Software

- This course is classified under “Computer Hardware”
- However, you will be much more capable if you master both hardware and software (and the interface between them)
 - Can develop better software if you understand the underlying hardware
 - Can design better hardware if you understand what software it will execute
 - Can design a better computing system if you understand both
- This course covers the HW/SW interface and microarchitecture
 - We will focus on tradeoffs and how they affect software

What Do I Expect From You?

- Required background: 240 (digital logic, RTL implementation, Verilog), 213/243 (systems, virtual memory, assembly)
 - **Learn the material thoroughly**
 - attend lectures, do readings, do homeworks
 - Do the work & **work hard**
 - **Ask questions, take notes, participate**
 - Perform the assigned readings
 - **Come to class on time**
 - **Start early – do not procrastinate**
 - If you want feedback, come to office hours
 - Remember “**Chance favors the prepared mind.**” (Pasteur)
-

What Do I Expect From You?

- How you prepare and manage your time is important
- There will be an assignment due almost every week
 - 7 Labs and 6 Homework Assignments
- This will be a heavy course
 - However, you will learn a lot of fascinating topics and understand how a microprocessor actually works (and how it can be made to work better)

How Will You Be Evaluated?

- Six Homeworks: 10%
- Seven Lab Assignments: 30%
- Midterm I: 15%
- Midterm II: 15%
- Final: 30%
- Our evaluation of your performance: 5%
 - Participation+discussion counts

More on Homeworks and Labs

■ Homeworks

- ❑ Do them to truly understand the material, not to get the grade
- ❑ Content from lectures, readings, labs, discussions
- ❑ All homework writeups *must* be your own work, written up individually and independently
 - However, you can discuss with others
- ❑ No late homeworks accepted

■ Labs

- ❑ These will take time.
- ❑ You need to start early and work hard.
- ❑ Assigned lab slots are for check-off only.
- ❑ Labs will be done individually unless specified otherwise.
- ❑ A total of five late lab days per semester allowed.

A Note on Cheating and Academic Dishonesty

- Absolutely no form of cheating will be tolerated
- You are all adults and we will treat you so
- See syllabus, CMU Policy, and ECE Academic Integrity Policy
 - Linked from syllabus
- Cheating → Failing grade (no exceptions)
 - And, perhaps more

Homeworks for Next Two Weeks

- Homework 0
 - Due next Monday (Jan 23), right before lecture
- Homework 1
 - Due Monday Jan 30, right before lecture, on Blackboard
 - MIPS warmup, ISA concepts, basic performance evaluation

Lab Assignment 1

- A functional C-level simulator for a subset of the MIPS ISA
- Due Friday Feb 3, at the end of Friday lab

- Start early, you will have a lot to learn
- Homework 1 and Lab 1 are synergistic
 - Homework questions are meant to help you in the Lab

Readings for Next Time

- Patt, "Requirements, Bottlenecks, and Good Fortune: Agents for Microprocessor Evolution," Proceedings of the IEEE 2001.
- P&P Chapter 1 (Fundamentals)
- P&H Chapters 1 and 2 (Intro, Abstractions, ISA, MIPS)
- Reference material throughout the course
 - MIPS R4000 Manual
 - (less so) x86 Reference Manual

Discussion for Next Time

- MIPS ISA Tutorial
 - Chris Fallin and Lavanya
 - Time(s) to be determined

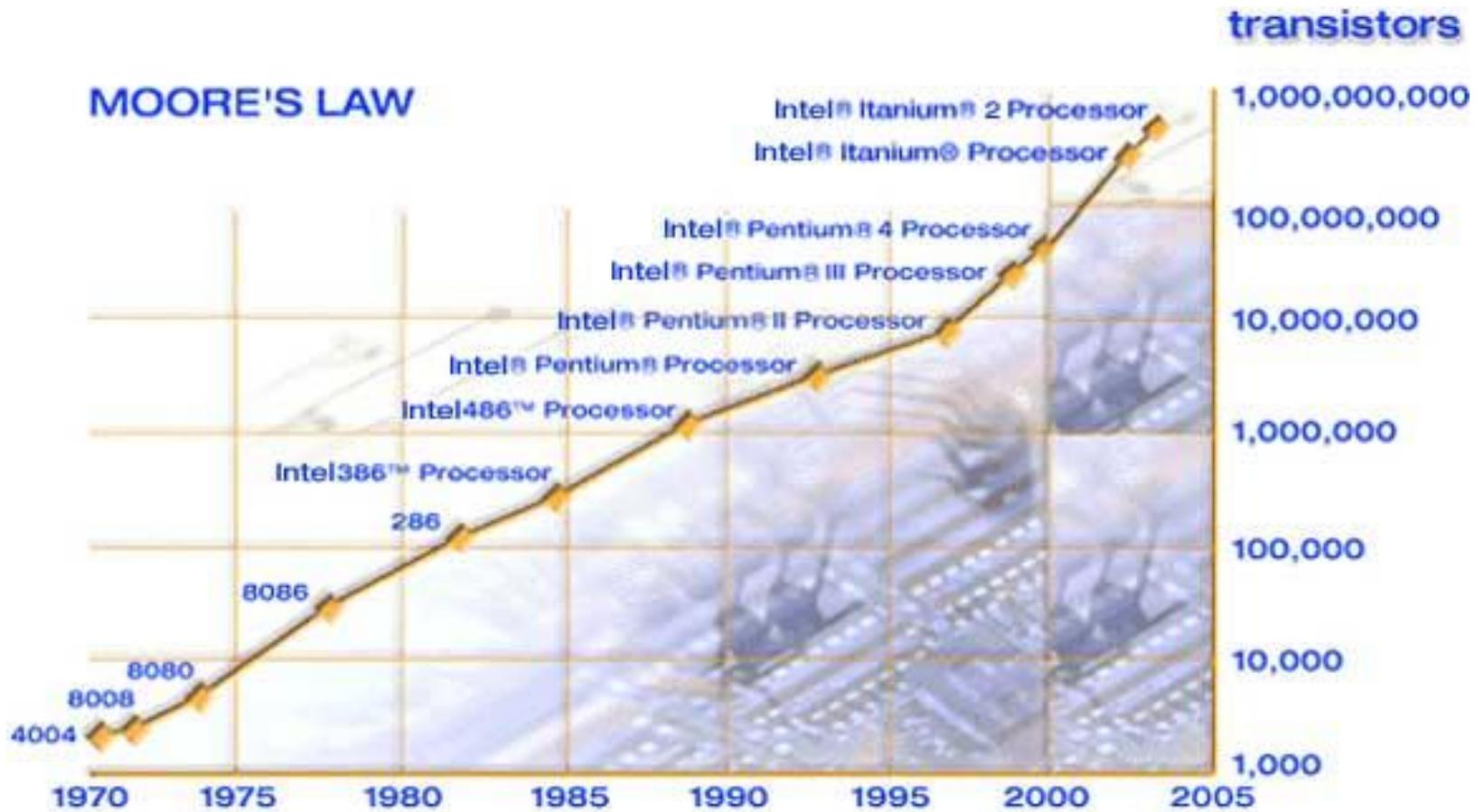
We did not cover the following slides in lecture.
These are for your preparation for the next lecture.

Why Study Computer Architecture?

What is Computer Architecture?

- The science and art of designing, selecting, and interconnecting hardware components and designing the hardware/software interface to create a computing system that meets functional, performance, energy consumption, cost, and other specific goals.
- We will soon distinguish between the terms *architecture*, and *microarchitecture*.

Moore's Law



Moore, “Cramming more components onto integrated circuits,”
Electronics Magazine, 1965.

Why Study Computer Architecture?

- **Enable better systems:** make computers faster, cheaper, smaller, more reliable, ...
 - By exploiting advances and changes in underlying technology/circuits
- **Enable new applications**
 - Life-like 3D visualization 20 years ago?
 - Virtual reality?
 - Personal genomics?
- **Enable better solutions** to problems
 - Software innovation is built into trends and changes in computer architecture
 - > 50% performance improvement per year has enabled
- **Understand why computers work the way they do**

Computer Architecture Today

- Today is a very exciting time to study computer architecture
- Industry is in a large paradigm shift (to multi-core)
- Many problems motivating and caused by the shift
 - Power/energy constraints
 - Complexity of design → multi-core
 - Technology scaling → new technologies
 - Memory wall/gap
 - Reliability wall/issues
 - Programmability wall/problem
- You can revolutionize the way computers are built, if you understand both the hardware and the software (and change each accordingly)
 - Book: Kuhn, “[The Structure of Scientific Revolutions](#)” (1962)