Welcome!

• Syllabus:  
  http://homepage.cs.uiowa.edu/~bdmyers/cs2630_sp17//syllabus/  
  • or linked from ICON  
• Class has been increased to 60 enrollments  
• To the 26 students on the waitlist:  
  • get started on the class, but probably not everyone will make it in 😞  
  • email me if you cannot see the ICON course  
• To the 60 students enrolled:  
  • be considerate to your peers and please make your decision to take the class quickly!  
  • if you take it, be committed to this course
CS 2630
Computer Organization

Meeting 1: Introduction
Brandon Myers
University of Iowa
One of my research projects

Make programming this

...feel like programming this
Why take 2630?

• The esoteric answer: **Computer** Science graduates should have an appreciation for how real **computers** work

• But really...
  
  1. It will be up to you to **design our new computer systems**...computer architects have been panicking for nearly a decade and they are *not* calming down
  
  2. Even if you vow to never, ever, EVER do anything except applications programming...at some point you will be have to **measure a system you’ve built**: performance (latency & throughput), energy usage, reliability, ... To understand how to measure/interpret/improve your system, you need to understand more of the computer
App

High-level language (e.g., C, Java)

Compiler

Operating system (e.g., Linux, Windows)

Instruction set architecture (e.g., MIPS)

Memory system

Processor

I/O system

Datapath & Control

Digital logic

Circuits

Devices (e.g., transistors)

Physics
The reality of our abstractions
Microprocessor Transistor Counts 1971-2011 & Moore's Law

The curve shows transistor count doubling every two years.
Processor performance

Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten
Dotted line extrapolations by C. Moore
Why isn’t multicore working?

Amdahl’s Law

Number of Processors

Speedup

Parallel Portion
- 50%
- 75%
- 90%
- 95%

95%
90%
75%
50%
TL;DR
Multicore will increase performance up to a point, but because of power issues as well as Amdahl’s Law, we can’t bet on the million-core chip. We need entirely new designs for the computers of the near future.

Dark Silicon and the End of Multicore Scaling

Hadi Esmaielzadeh† Emily Blem‡ Renée St. Amant§ Karthikeyan Sankaralingam¶ Doug Burger¶
1University of Washington 2University of Wisconsin-Madison 3The University of Texas at Austin 4Microsoft Research
hadianeh@cs.washington.edu blem@cs.wisc.edu stamant@cs.utexas.edu karu@cs.wisc.edu dburger@microsoft.com

ABSTRACT
Since 2005, processor designers have increased core counts to exploit Moore’s Law scaling, rather than focusing on single-core performance. The failure of Dennard scaling, to which the shift to multicore parts is partially a response, may soon limit multicore scaling just as single-core scaling has been curtailed. This paper models multicore scaling limits by combining device scaling, single-core scaling, and multicore scaling to measure the speedup potential for a set of parallel workloads for the next five technology generations. For device scaling, we use both the ITRS projections and a set

and compiler advances, Moore’s Law, coupled with Dennard scaling [11], has resulted in commensurate exponential performance increases. The recent shift to multicore designs has aimed to increase the number of cores along with transistor count increases, and continue the proportional scaling of performance. As a result, architecture researchers have started focusing on 100-core and 1000-core chips and related research topics and called for changes to the undergraduate curriculum to solve the parallel programming challenge for multicore designs at these scales.

With the failure of Dennard scaling—and thus slowed supply volu-
Processor for augmented reality and visual computing

http://www.movidius.com/applications
Brain-inspired computers
Custom computers for search engines

Project Catapult - Microsoft Research
www.microsoft.com/en-us/research/project/project-catapult
Project Catapult is a Microsoft venture that investigates the use of field-programmable gate arrays (FPGAs) to improve performance, reduce power consumption, and ...
conventional design of a general-purpose processor
Why take 2630?

• The esoteric answer: Computer Science graduates should have an appreciation for how real computers work

• But really...
  
  • 1. It will be up to you to design our new computer systems...computer architects have been panicking for nearly a decade and they are not calming down
  
  • 2. Even if you vow to never, ever, EVER do anything except applications programming...at some point you will be have to measure a system you’ve built: performance (latency & throughput), energy usage, reliability, ... To understand how to measure/interpret/improve your system, you need to understand more of the computer
Why is my program slow?

• Application programmer’s ideal view of the underlying system: “performance is solely determined by the number of operations (remember O(N) from 2230?)”
  • The program is slow, so I’ll fix my algorithm to be O(N^3) instead of O(N^4)
  • It’s still slow, now what?

• Sometimes the programmer needs a more detailed view of the underlying computer system
  • Suppose the program was not using the computer’s **cache** or **parallel processing** effectively
  • Methodology is important, too: What is the **process** for discovering why the program is slow? Scientific method: hypothesize, measure, interpret, repeat...
What is CS2630 about?

- Instruction set architecture (e.g., MIPS)
- Compiler: translating source code (C or Java) Programs to assembly language And linking your code to Library code
- How the software talks To the hardware
- Processor
- Memory system
- I/O system
- Datapath & Control
- How a processor runs MIPS Programs!
- Digital logic
- How switches (1 or 0) can be used to build Interesting functions: from integer arithmetic to programmable computers
Rough schedule of CS2630

• ~week 1-5: How do we represent and store numbers? How do we program a computer?
• ~week 6-8: How do we build complex functions from simpler components like switches?
• ~week 8-10: How do we design a basic processor that runs MIPS programs?
• ~week 10-14: What are the techniques for diagnosing performance issues and improving performance of a processor?
• Bonus: e.g., How do we run more than one program at a time on a processor? How do we represent real numbers?
Let’s look at the syllabus...

• ICON > 2630 Spring 2017 > Syllabus
• Or, 
  [http://homepage.cs.uiowa.edu/~bdmyers/cs2630_sp17//syllabus/](http://homepage.cs.uiowa.edu/~bdmyers/cs2630_sp17//syllabus/)
Textbook

• Harris & Harris, *Digital Design and Computer Architecture*, 2nd ed.

• *(1st edition is okay, too)*
Be successful in CS2630

• Come to class; reading slides is a poor substitute
• Active learning, peer teaching, and other activities to replace the lack of labs/discussions
• Attend Debug Your Brain and/or office hours
• Help your peers on ICON and in class
• Midterm in class (March 6)
• Keep on top of announcements in ICON/website
Peer instruction

• Think, answer, discuss...
• **See syllabus** for information on purchasing a clicker license if you do not yet have one
• Use your smartphone or any web-enabled device!
• Participation counts, *not* right answers

• Too shy to ask a question in class? You can also anonymously ask questions on the ResponseWare app using the chat feature

• Some class meetings involve use of your computer (need at least 1 per pair of students)
  • If bringing a laptop presents a hardship, email me
What to do now

• HW 1 is out
• Quiz 1 is out
• Read the syllabus online
  • go on ICON and take the syllabus survey if you want to vote on policies
• Buy your clicker license (if you don’t yet have one) and go through getting started doc
• Check ICON and reply to the discussion question