Lecture 1: CS/ECE 3810 Introduction

• Today’s topics:
  ▪ Why computer organization is important
  ▪ Logistics
  ▪ Modern trends
Why Computer Organization
Why Computer Organization

• Embarrassing if you are a BS in CS/CE and can’t make sense of the following terms: DRAM, pipelining, cache hierarchies, I/O, virtual memory, …

• Embarrassing if you are a BS in CS/CE and can’t decide which processor to buy: 3 GHz P4 or 2.5 GHz Athlon (helps us reason about performance/power), …

• Obvious first step for chip designers, compiler/OS writers

• Will knowledge of the hardware help you write better programs?
Must a Programmer Care About Hardware?

• Must know how to reason about program performance and energy

• Memory management: if we understand how/where data is placed, we can help ensure that relevant data is nearby

• Thread management: if we understand how threads interact, we can write smarter multi-threaded programs

→ Why do we care about multi-threaded programs?
Example

200x speedup for matrix vector multiplication

- Data level parallelism: 3.8x
- Loop unrolling and out-of-order execution: 2.3x
- Cache blocking: 2.5x
- Thread level parallelism: 14x

Further, can use accelerators to get an additional 100x.
Key Topics

- Moore’s Law, power wall
- Use of abstractions
- Assembly language
- Computer arithmetic
- Pipelining
- Using predictions
- Memory hierarchies
- Reliability
Logistics

• See class web-page
  http://www.cs.utah.edu/~rajeev/cs3810

• TAs: Karl, Surya, Shirley, … ; Office hours: TBA

• Most communication on Canvas; email me directly to set up office hours, or meet me right after class

• Textbook: Computer Organization – HW/SW Interface, Patterson and Hennessy, 5th edition
Course Organization

• 30% midterm, 40% final, 30% assignments

• ~9 assignments – you may skip one; assignments due at the start of class (upload on Canvas)

• Co-operation policy: you may discuss – you may not see someone else’s written matter when writing your solution

• Exams are open-book and open-notes

• Print slides just before class

• Screencast YouTube videos
Microprocessor Performance

50% improvement every year!!
What contributes to this improvement?

Source: H&P Textbook
Power Consumption Trends

- Dyn power $\alpha$ activity $\times$ capacitance $\times$ voltage$^2$ $\times$ frequency

- Voltage and frequency are somewhat constant now, while capacitance per transistor is decreasing and number of transistors (activity) is increasing

- Leakage power is also rising (function of #trans and voltage)

Source: H&P Textbook
Important Trends

• Running out of ideas to improve single thread performance
• Power wall makes it harder to add complex features
• Power wall makes it harder to increase frequency
Important Trends

- Historical contributions to performance:
  1. Better processes (faster devices) ~20%
  2. Better circuits/pipelines ~15%
  3. Better organization/architecture ~15%

In the future, bullet-2 will help little and bullet-1 will eventually disappear!

<table>
<thead>
<tr>
<th>Year</th>
<th>Pentium</th>
<th>P-Pro</th>
<th>P-II</th>
<th>P-III</th>
<th>P-4</th>
<th>Itanium</th>
<th>Montecito</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transistors</td>
<td>3.1M</td>
<td>5.5M</td>
<td>7.5M</td>
<td>9.5M</td>
<td>42M</td>
<td>300M</td>
<td>1720M</td>
</tr>
<tr>
<td>Clock Speed</td>
<td>60M</td>
<td>200M</td>
<td>300M</td>
<td>500M</td>
<td>1500M</td>
<td>800M</td>
<td>1800M</td>
</tr>
</tbody>
</table>

Moore’s Law in action

At this point, adding transistors to a core yields little benefit
What Does This Mean to a Programmer?

• Today, one can expect only a 20% annual improvement; the improvement is even lower if the program is not multi-threaded
  - A program needs many threads
  - The threads need efficient synchronization and communication
  - Data placement in the memory hierarchy is important
  - Accelerators should be used when possible
Challenges for Hardware Designers

• Find efficient ways to

  ▪ boost single-thread performance
  ▪ improve data sharing
  ▪ boost programmer productivity
  ▪ manage the memory system
  ▪ build accelerators for important kernels
  ▪ reduce system energy per instruction
The HW/SW Interface

Application software

----------------------------------

Systems software
(OS, compiler)

----------------------------------

Hardware

a[i] = b[i] + c;

Compiler

lw $15, 0($2)
add $16, $15, $14
add $17, $15, $13
lw $18, 0($12)
lw $19, 0($17)
add $20, $18, $19
sw $20, 0($16)

Assembler

00000010110100000
110100000100010
...

Computer Components

- Input/output devices
- Secondary storage: non-volatile, slower, cheaper
- Primary storage: volatile, faster, costlier
- CPU/processor (datapath and control)
Wafers and Dies

Source: H&P Textbook
Manufacturing Process

• Silicon wafers undergo many processing steps so that different parts of the wafer behave as insulators, conductors, and transistors (switches)

• Multiple metal layers on the silicon enable connections between transistors

• The wafer is chopped into many dies – the size of the die determines yield and cost

• Transistor density increases by 35% per year and die size increases by 10-20% per year… functionality improvements!

• Transistor speed improves linearly with size (complex equation involving voltages, resistances, capacitances)

• Wire delays do not scale down at the same rate as transistor delays
Memory and I/O Technology Trends

• DRAM density increases by 40-60% per year, latency has reduced by 33% in 10 years (the memory wall!), bandwidth improves twice as fast as latency decreases

• Disk density improves by 100% every year, latency improvement similar to DRAM

• Networks: primary focus on bandwidth; 10Mb → 100Mb in 10 years; 100Mb → 1Gb in 5 years
Next Class

• Topics: Performance, MIPS instruction set architecture (Chapter 2)

• Visit the class web-page
  http://www.cs.utah.edu/~rajeev/cs3810
Title

• Bullet