Lecture 2: Performance

• Today’s topics:
  - Technology wrap-up
  - Performance trends and equations

• Reminders: YouTube videos, canvas, and class webpage: https://www.cs.utah.edu/~rajeev/cs3810/
Summary

• Two roadblocks: power and ideas

• Fixed power budget because of cooling constraints; implies that frequency can’t be increased; discourages complex ideas

• End of voltage (Dennard) scaling in early 2010s

• Has led to dark silicon and dim silicon (occasional turbo)
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
  ▪ improve single-thread performance and energy
  ▪ improve data sharing
  ▪ boost programmer productivity
  ▪ manage the memory system
  ▪ build accelerators for important kernels
  ▪ provide security
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

00000010110110000
1101000000100010
...

Hardware Systems software
(OS, compiler) Application software
Computer Components

- Input/output devices

- Secondary storage: non-volatile, slower, cheaper (HDD/SSD)

- Primary storage: volatile, faster, costlier (RAM)

- 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
Processor Technology Trends


- 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
Performance Metrics

• Possible measures:
  ▪ response time – time elapsed between start and end of a program
  ▪ throughput – amount of work done in a fixed time

• The two measures are usually linked
  ▪ A faster processor will improve both
  ▪ More processors will likely only improve throughput
  ▪ Some policies will improve throughput and worsen response time (or vice versa)

• What influences performance?
Execution Time

Consider a system $X$ executing a fixed workload $W$

Performance$_X = 1 / \text{Execution time}_X$

Execution time = response time = wall clock time
- Note that this includes time to execute the workload as well as time spent by the operating system co-ordinating various events

The UNIX “time” command breaks up the wall clock time as user and system time
Speedup and Improvement

- System X executes a program in 10 seconds, system Y executes the same program in 15 seconds
- System X is 1.5 times faster than system Y
- The speedup of system X over system Y is 1.5 (the ratio)  
  \[ \frac{\text{perf X}}{\text{perf Y}} = \frac{\text{exectime Y}}{\text{exectime X}} \]
- The performance improvement of X over Y is 
  \[ 1.5 - 1 = 0.5 = 50\% = \frac{\text{perf X} - \text{perf Y}}{\text{perf Y}} = \text{speedup} - 1 \]
- The execution time reduction for system X, compared to Y is 
  \[ \frac{15-10}{15} = 33\% \]
  The execution time increase for Y, compared to X is 
  \[ \frac{15-10}{10} = 50\% \]
A Primer on Clocks and Cycles
Performance Equation - I

CPU execution time = CPU clock cycles × Clock cycle time
Clock cycle time = 1 / Clock speed

If a processor has a frequency of 3 GHz, the clock ticks 3 billion times in a second – as we’ll soon see, with each clock tick, one or more/less instructions may complete

If a program runs for 10 seconds on a 3 GHz processor, how many clock cycles did it run for?

If a program runs for 2 billion clock cycles on a 1.5 GHz processor, what is the execution time in seconds?
Performance Equation - II

CPU clock cycles = number of instrs \times \text{avg clock cycles per instruction (CPI)}

Substituting in previous equation,

Execution time = clock cycle time \times \text{number of instrs} \times \text{avg CPI}

If a 2 GHz processor graduates an instruction every third cycle, how many instructions are there in a program that runs for 10 seconds?