Lecture 2: Performance

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

- Reminders: YouTube videos, canvas, and class webpage: http://www.cs.utah.edu/~rajeev/cs3810/

- Apply to be a CS tutor: https://www.cs.utah.edu/tutor-app/

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
- Additional performance provided by: more cores, occasional spikes in frequency, accelerators

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!

	Pentium	P-Pro	P-II	P-III	P-4	Itanium	Montecito
Year	1993	95	97	99	2000	2002	2005
Transistors	3.1M	5.5M	7.5M	9.5M	42M	300M	1720M
Clock Speed	60M	200M	300M	500M	1500M	800M	1800M

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

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

$\downarrow$ Compiler

Systems software
(OS, 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)
```

$\downarrow$ Assembler

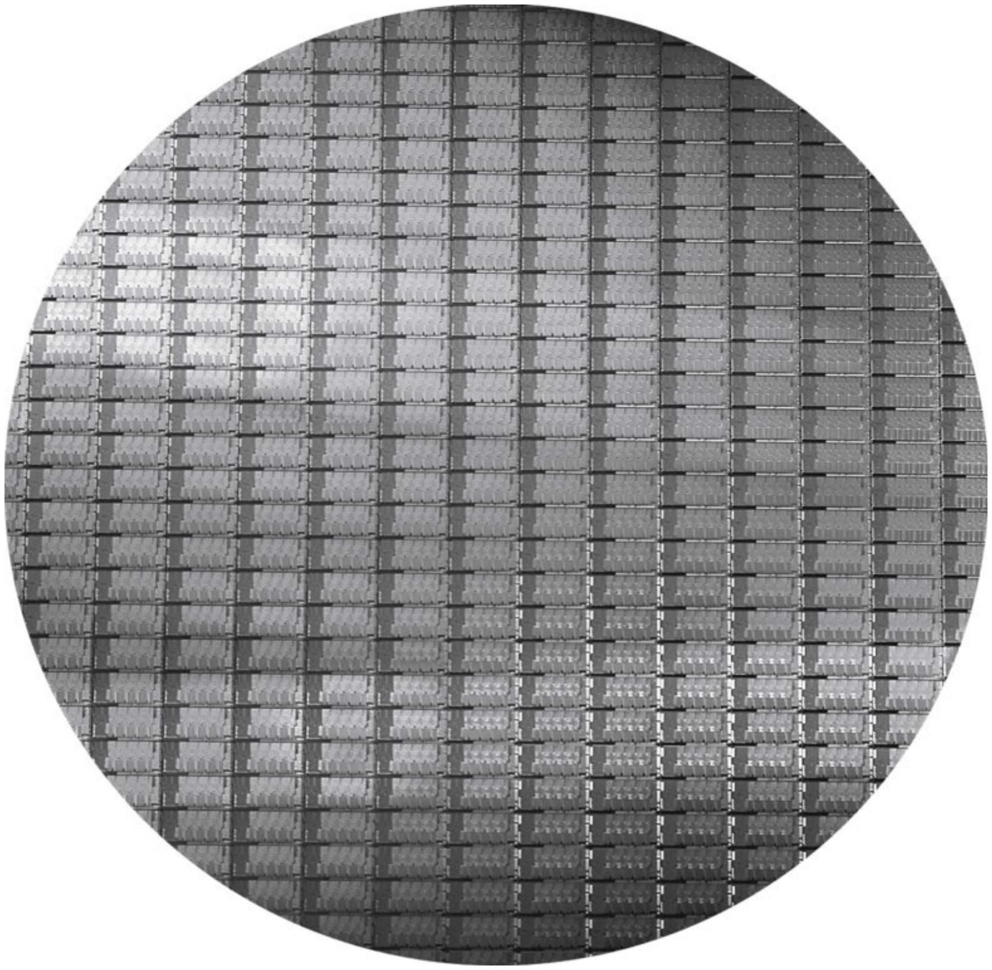
Hardware

```
000000101100000
110100000100010
…
```

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

- Shrinking of transistor sizes: 250nm (1997) → 130nm (2002) → 70nm (2008) → 35nm (2014) → 2019, start of transition from 14nm to 10nm
- 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
- What influences performance?

Execution Time

Consider a system X executing a fixed workload W

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

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)
  = perf X / perf Y = exectime Y / exectime X

- The performance improvement of X over Y is
  1.5 -1 = 0.5 = 50% = (perf X – perf Y) / perf Y = speedup - 1

- The execution time reduction for system X, compared to Y is (15-10) / 15 = 33%
  The execution time increase for Y, compared to X is (15-10) / 10 = 50%

A Primer on Clocks and Cycles

Performance Equation - I

CPU execution time = CPU clock cycles x 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 x avg clock cycles per instruction (CPI)

Substituting in previous equation,

Execution time = clock cycle time x number of instrs x 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?

Factors Influencing Performance

Execution time = clock cycle time x number of instrs x avg CPI

- Clock cycle time: manufacturing process (how fast is each transistor), how much work gets done in each pipeline stage (more on this later)

- Number of instrs: the quality of the compiler and the instruction set architecture

- CPI: the nature of each instruction and the quality of the architecture implementation

Example

Execution time = clock cycle time x number of instrs x avg CPI

Which of the following two systems is better?

- A program is converted into 4 billion MIPS instructions by a compiler ; the MIPS processor is implemented such that each instruction completes in an average of 1.5 cycles and the clock speed is 1 GHz

- The same program is converted into 2 billion x86 instructions; the x86 processor is implemented such that each instruction completes in an average of 6 cycles and the clock speed is 1.5 GHz

Power and Energy

- Total power = dynamic power + leakage power
- Dynamic power α activity x capacitance x voltage^2 x frequency
- Leakage power α voltage
- Energy = power x time

  (joules) (watts) (sec)

Example Problem

- A 1 GHz processor takes 100 seconds to execute a program, while consuming 70 W of dynamic power and 30 W of leakage power. Does the program consume less energy in Turbo boost mode when the frequency is increased to 1.2 GHz?

Example Problem

- A 1 GHz processor takes 100 seconds to execute a program, while consuming 70 W of dynamic power and 30 W of leakage power. Does the program consume less energy in Turbo boost mode when the frequency is increased to 1.2 GHz?

  Normal mode energy = 100 W x 100 s = 10,000 J
  Turbo mode energy = (70 x 1.2 + 30) x 100/1.2 = 9,500 J

  Note:
  - Frequency only impacts dynamic power, not leakage power.
  - We assume that the program's CPI is unchanged when frequency is changed, i.e., exec time varies linearly with cycle time.