

Lecture 2: Performance

- Today's topics:
 - Technology wrap-up
 - Performance trends and equations
- Reminders: YouTube videos, Canvas, Piazza, class webpage:
<https://www.cs.utah.edu/~rajeev/cs3810/>

Summary

$$\text{Dyn Power} \propto V^2$$

- Three roadblocks: power, ideas, technology scaling ^{5nm}
- 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; the end of Moore's Law also imminent ^{Trans density 2x}
- Has led to dark silicon and dim silicon (occasional turbo) ^{2 yrs}
- The rise of accelerators ^{18-24 mos}

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?

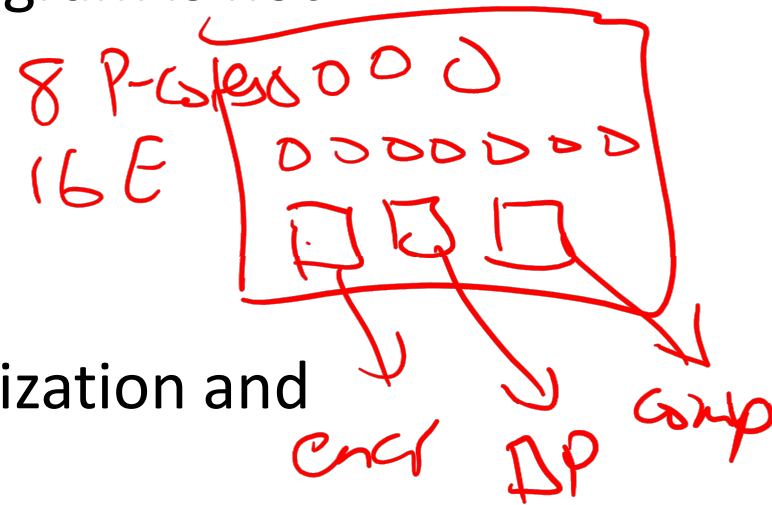
- Today, one can expect only a 20% annual improvement; the improvement is even lower if the program is not multi-threaded

- 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

What is This Course About?



The HW/SW Interface

Application software

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

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

```
000000101100000
110100000100010
...
```

Systems software
(OS, compiler)

Hardware

Intel/AMD

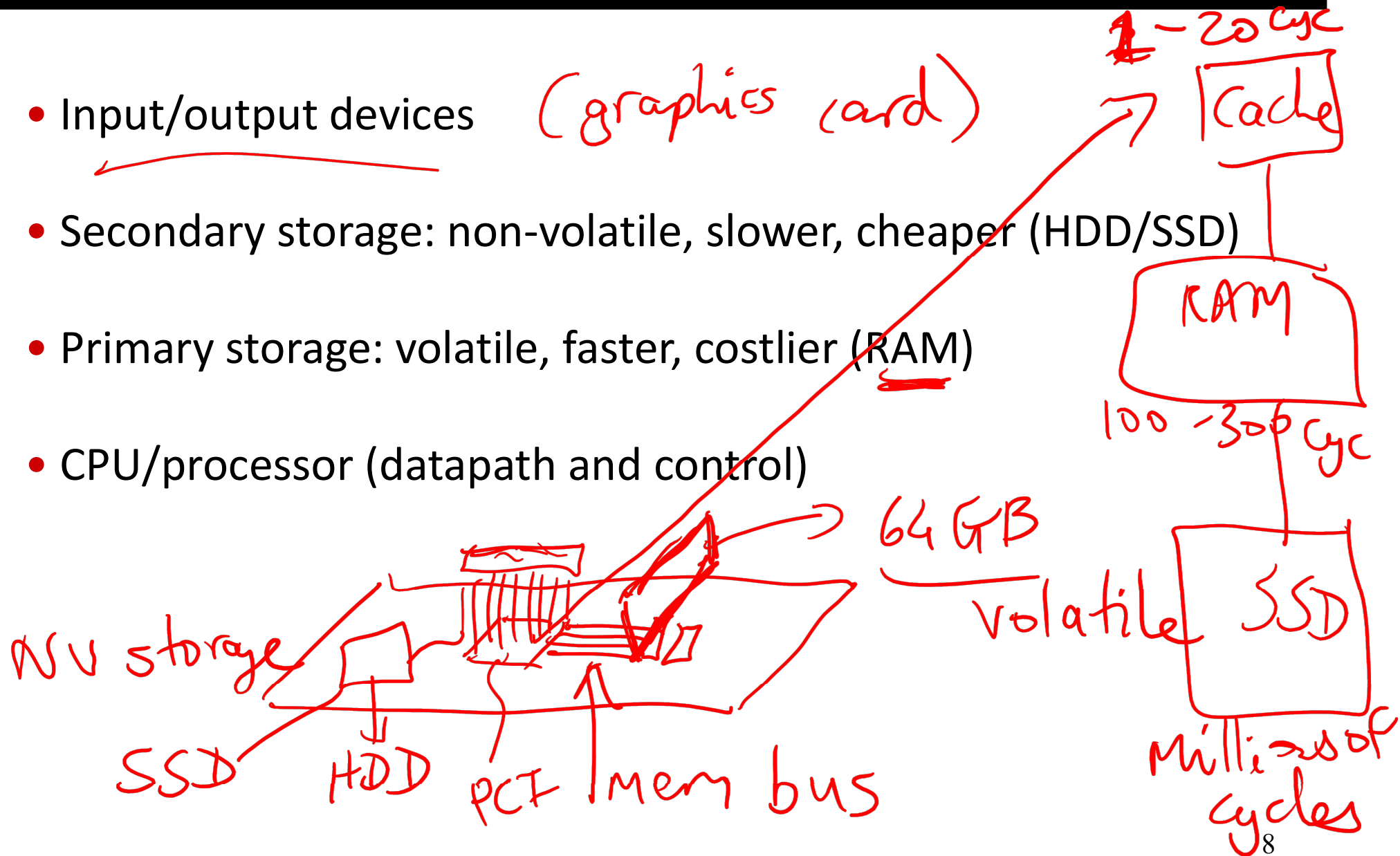
x86

Instr set arch

ISA

Computer Components

- Input/output devices (graphics card)
- Secondary storage: non-volatile, slower, cheaper (HDD/SSD)
- Primary storage: volatile, faster, costlier (RAM)
- CPU/processor (datapath and control)



Wafers and Dies

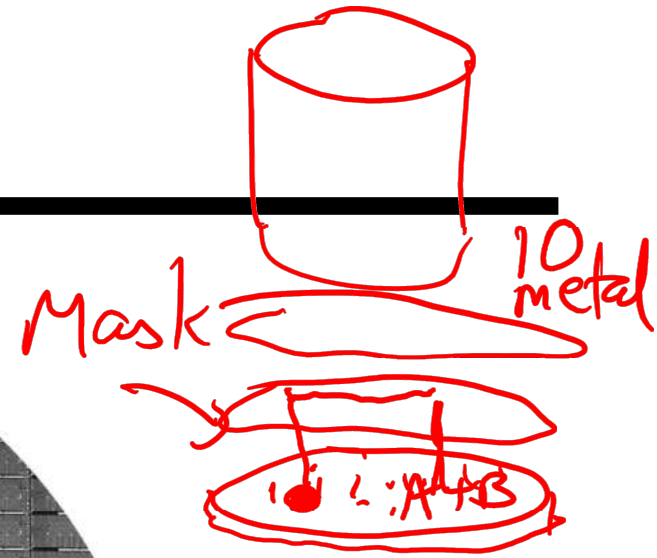
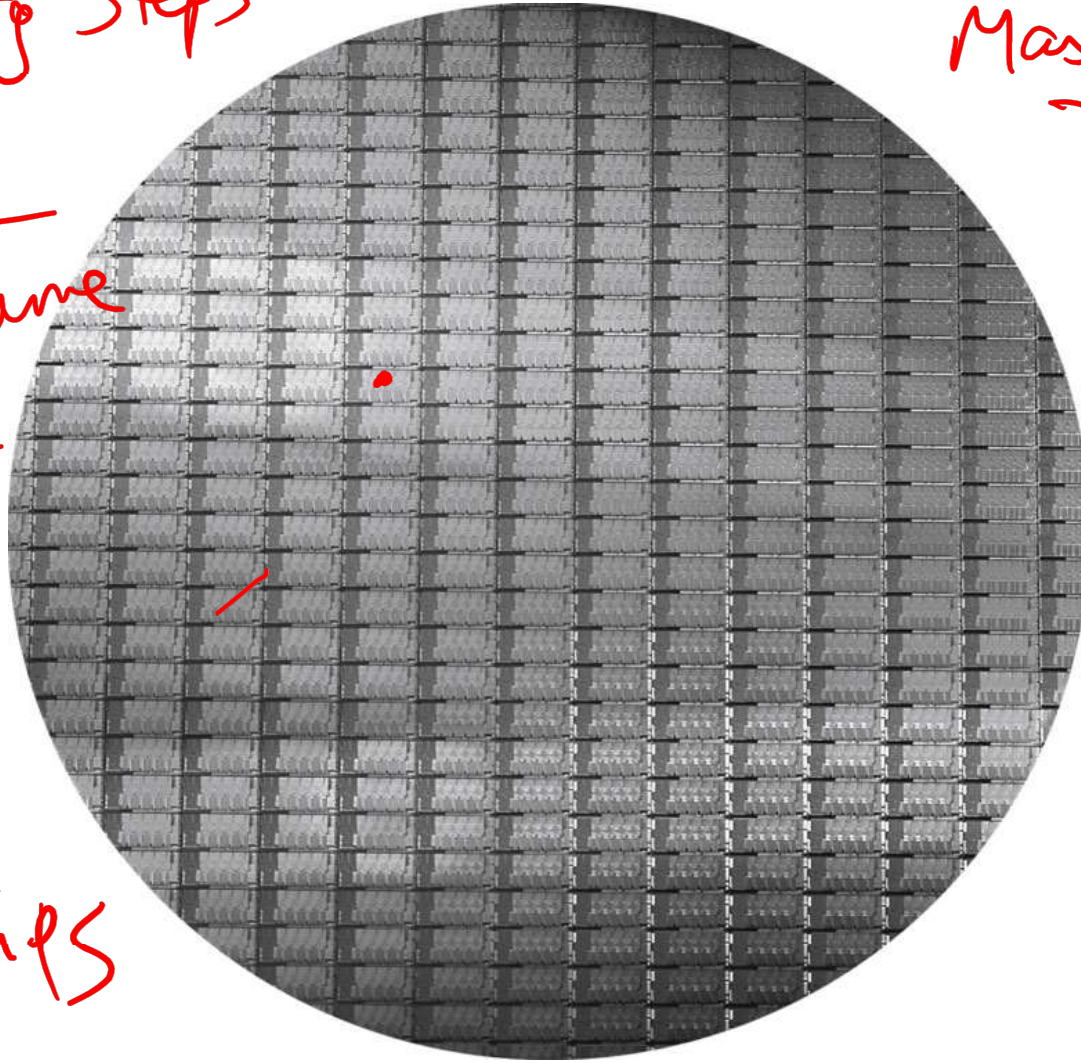
Optics

Cost \propto mfg steps

Cost $\propto \frac{1}{\text{volume}}$

Cost $\propto \frac{\text{size}}{\text{size}}$

Yield is
good chips



Size of
chip $\downarrow 2x$
cost $\downarrow 4x$

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

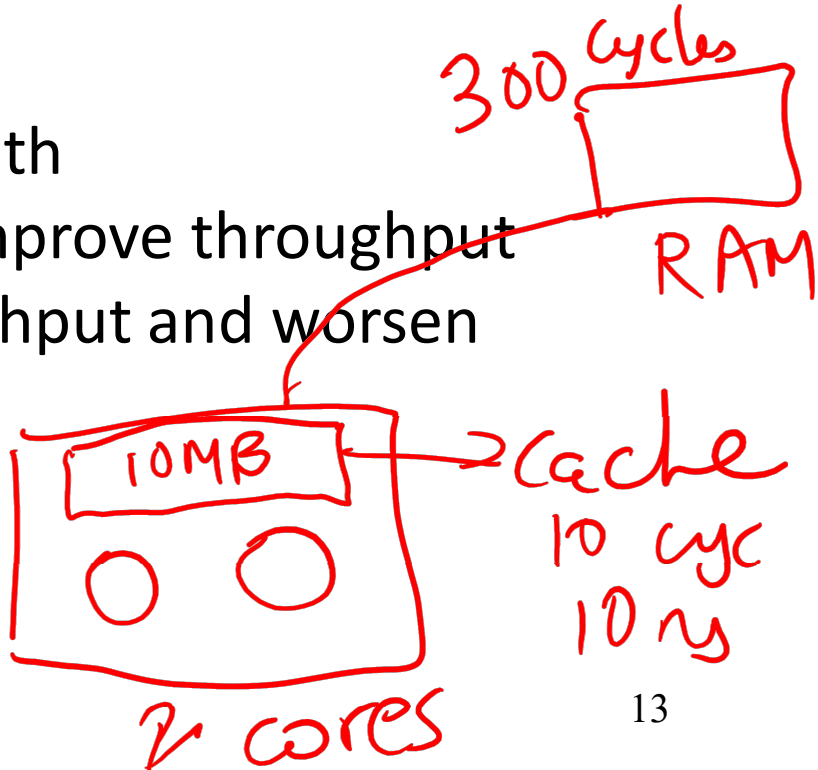
Processor Technology Trends

- Shrinking of transistor sizes: 250nm (1997) → 130nm (2002) → 70nm (2008) → 35nm (2014) → 2019 transition to 10nm, now transitioning to 5nm
- 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) *comp gets faster*
- Wire delays do not scale down at the same rate as transistor delays *com m gets slower*

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: *→ latency*
 - response time – time elapsed between start and end of a program *sec ns*
 - throughput – amount of work done in a fixed time *jobs/sec*
 - 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?
 - 5 sec ← 1 prog on the proc*
 - each 7 sec ← 2 prog on the proc*
- 
- The diagram shows a hand-drawn box representing a processor. Inside the box, there is a smaller box labeled '10MB' and two circles. Below the box, it is labeled '2 cores'. To the right of the box, there is a label 'Cache' with '10 cyc' and '10 ns' written below it. Above the box, there is a label 'RAM' with '300 cycles' written above it. A red arrow points from the 'RAM' label to the top of the processor box.

Execution Time (Response time Latency)

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

$$\frac{13K}{10K} = \underline{\underline{1.3x}}$$

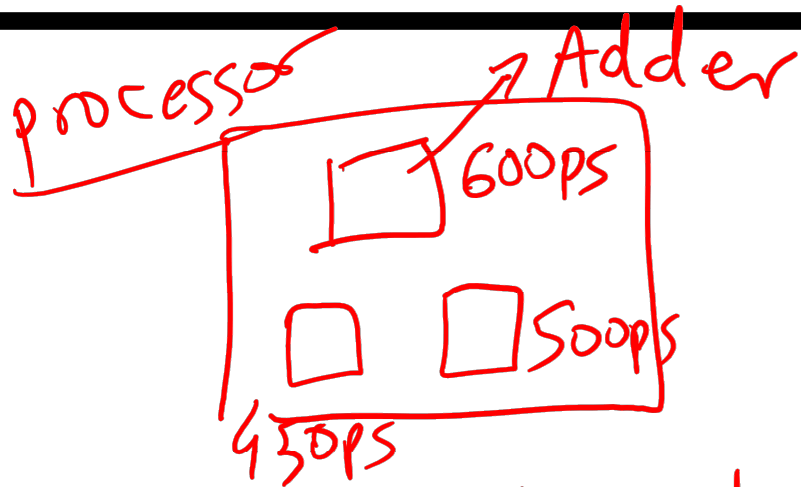
- new*

• 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)
 $= \text{perf X} / \text{perf Y} = \text{exectime Y} / \text{exectime X} = 15/10 = 1.5$

old new sal is 30% higher new sal = old sal old sal
- The performance improvement of X over Y is
 $1.5 - 1 = 0.5 = 50\% = (\text{perf X} - \text{perf Y}) / \text{perf Y} = \text{speedup} - 1 = \frac{13K - 10K}{10K}$
- The execution time reduction for system X, compared to Y is $(15-10) / 15 = 33\%$
 $= \frac{\text{perf X}}{\text{perf Y}} - 1 = \frac{3K}{10K}$

The execution time increase for Y, compared to X is
 $(15-10) / 10 = 50\%$
 $= \text{Speedup} - 1 = 1.5 - 1 = 0.5 = 50\%$
 $= 0.3 = 30\%$

A Primer on Clocks and Cycles (frequency)



$$\text{Hz} = \frac{1}{\text{sec}}$$

$$1 \text{ sec}$$

$$10^{-3} \text{ sec} \quad \frac{1}{1000} \text{ ms}$$

$$10^{-6} \text{ sec} \quad \mu\text{s}$$

$$10^{-9} \text{ sec} \quad \text{ns}$$

$$10^{-12} \text{ sec} \quad \text{ps}$$

CPI
cycles per instr

A hand-drawn timing diagram showing a single clock cycle. It consists of a square wave pulse. Above the pulse, there are two vertical arrows pointing to the rising and falling edges. Below the pulse, a horizontal double-headed arrow spans the width of the pulse and is labeled "1 cycle time = 10⁹". Below this, the text "600ps" is crossed out and "700ps" is written. To the right of the diagram, the following calculations are shown:

$$= \frac{1}{700 \times 10^{-12}}$$

$$= \frac{1}{700 \times 10^{-3} \times 10^{-9}}$$

$$= \frac{1}{0.7} = 1.42 \text{ GHz}$$

Measure of perf =
clock speed = # cycles in

$$= \frac{1}{\text{cycle time (sec)}} \quad 1 \text{ second}$$

Performance Equation - I

CPU execution time = CPU clock cycles x Clock cycle time

Clock cycle time = $1 / \text{Clock speed}$

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?