Computer Architecture

Lecture 2 - Performance

Fall 2011

Reading: 1.4 – 1.10

Homework: 1.3, 1.4, 1.5

Roadmap for the term: major topics

- Computer Systems Overview
- Technology Trends
- Instruction sets (and Software)
- Logic & Arithmetic
- Performance
- Processor Implementation
- Memory Systems
- Input/Output

Performance Outline

- **Motivation**
- **Defining Performance**
- **Common Performance Metrics**
- **Benchmarks**
- **Amdahl's Law**

Performance

- **Goal: Learn to "measure, report and summarize" performance of a computer system**
- **Why study performance?**
  - **To make intelligent decisions when choosing a system**
  - **To make intelligent decisions when designing a system**
  - **Understand impact of implementation decisions**
- **Challenges**
  - **How do we measure performance accurately?**
  - **How do we compare performance fairly?**

What's a good measure of performance?

- **Execution Time (response time, latency)**
  - How long it takes to complete a single task
  - Example: "how long does it take to break an MP3 file?"
- **Throughput**
  - How many tasks are completed per unit time
  - Example: "how many MP3 files can I break per hour?
- **The measure we use depends on the application**

Computer Performance

- Response Time (latency)
  - How long does it take for my job to run?
  - How long does it take to execute a job?
  - How long must I wait for the database query?
- Throughput
  - How many jobs can the machine run at once?
  - What is the average execution rate?
  - How much work is getting done per unit time?

Execution Time vs. Throughput

- **Analogy: passenger airplanes (book Figure 1.13)**
  - **Concorde - fastest "response time" for an individual user**
  - **Boeing 747 - highest passenger throughput**

Airplane	Passenger Capacity	Cruising Range (miles)	Cruising Speed (mph)	Passenger Throughput (passengers x mph)
Boeing 777	375	4630	610	228,750
Boeing 747	470	4150	610	286,700
Concorde	132	4000	1350	178,200
DC-8-50	146	8720	544	79,424

Execution Time

- **Elapsed Time**
  - **counts everything**
    (disk and memory accesses, I/O , etc.)
  - **a useful number, but often not good for comparison purposes**

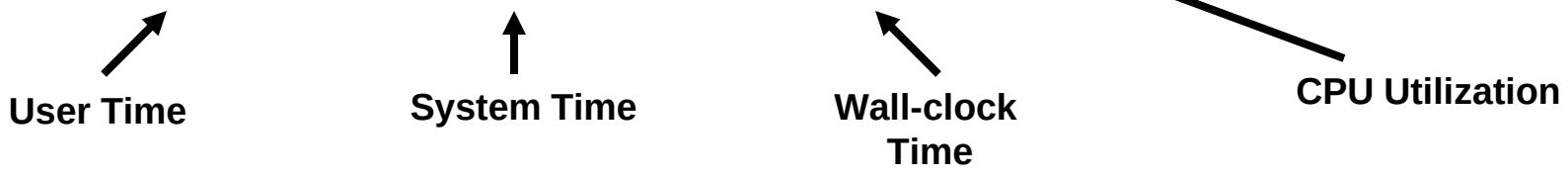
Execution Time

- CPU time
  - doesn't count I/O or time spent running other programs
  - can be broken up into system time, and user time
  - Our focus: user CPU time
  - time spent executing the lines of code that are "in" our program

Measuring Execution Time

- **Wall-clock time or elapsed time**
  - **Includes I/O waiting**
  - **Includes time while OS runs other jobs**
- **CPU time - measured by OS**
  - **User time - time spent in user program during execution**
  - **System time - time spent in OS during execution**
- **Measuring CPU time: Unix/Linux `time` command**

```
% time myprog
        ...
90.7u     12.9s     2:39     65%
```



Performance Outline

- Motivation
- Defining Performance
- Common Performance Metrics
- Benchmarks
- Amdahl's Law

Defining Performance using Execution Time

- **For a <u>given program</u> on machine X:**

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

- **Comparing performance of machines:**

  $\text{Performance}_X > \text{Performance}_Y$ if

  $\text{Execution Time}_X < \text{Execution Time}_Y$

Defining Performance (cont'd)

- We say "X is *n* times faster than Y" if:

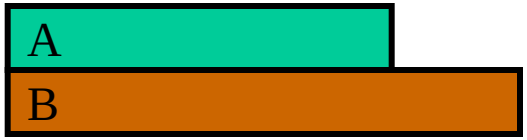
$$\frac{\text{Performance}_X}{\text{Performance}_Y} = n$$

$$\frac{\text{Performance}_X}{\text{Performance}_Y} = \frac{\text{Execution time}_Y}{\text{Execution time}_X} = n$$

Use this definition to compare performance in homework & exam problems!

Example - Performance

Machine	Execution Time
A	15 seconds
B	20 seconds



- **Which machine is faster?** A

- **By how much?**

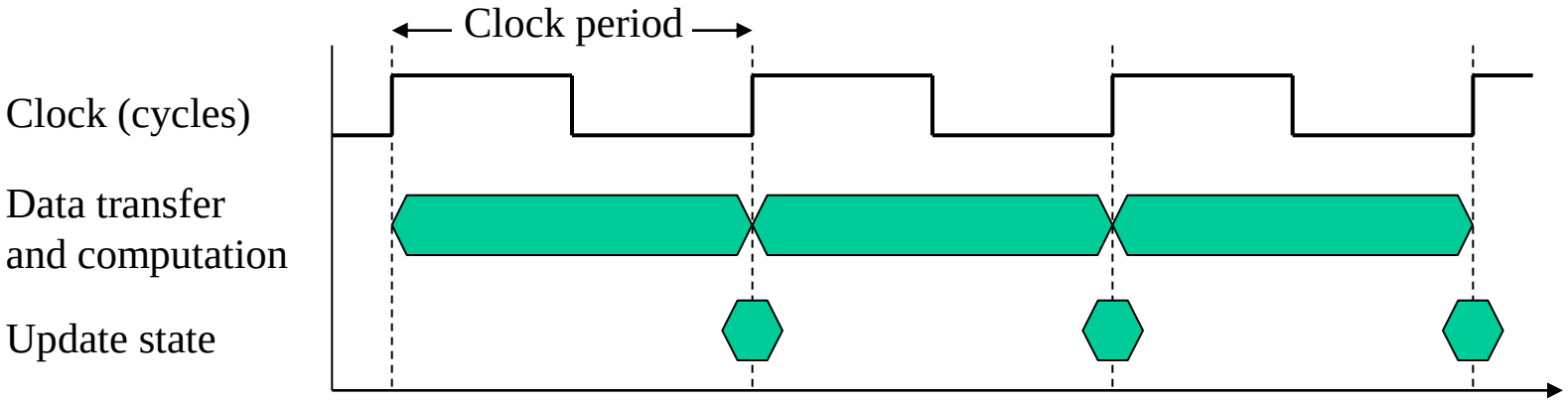
$$n = \frac{\text{Performance}_A}{\text{Performance}_B} = \frac{\text{Execution time}_B}{\text{Execution time}_A} = \frac{20 \text{ sec.}}{15 \text{ sec.}} = 1.33$$

Performance Outline

- Motivation
- Defining Performance
- Common Performance Metrics
- Benchmarks
- Amdahl's Law

CPU Clocking

- **Operation of digital hardware governed by a constant-rate clock**

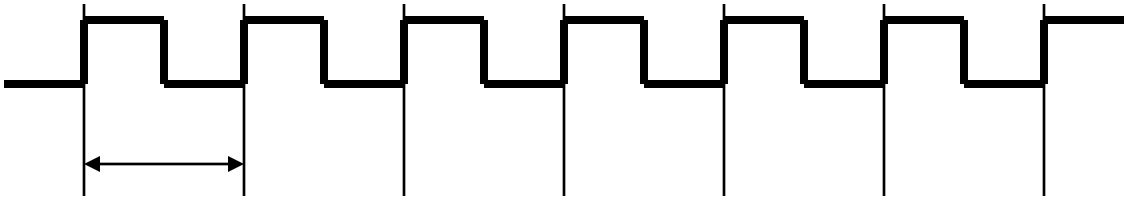


  - Clock period: duration of a clock cycle
    - e.g., 250ps = 0.25ns = 250×10^{-12}s
  - Clock frequency (rate): cycles per second
    - e.g., 4.0GHz = 4000MHz = 4.0×10^{9}Hz

Clocks and Performance

- Every processor has a clock
  - Clock frequency - MHz (or GHz)
  - Clock cycle time / period - ns (or ps)
- How do we relate clock to program performance?

Clock cycle time = 1/clock rate



1 Hz means that an event repeats once per second

Clock cycle time

$$\text{CPU time for a program} = \frac{\text{clock cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{clock cycle}}$$

CPU time for a program = # CPU clock cycle for a program x clock cycle time

CPU time for a program = # CPU clock cycle for a program / clock rate.

Example - Clock Cycles

Machine	Execution Time	Clock Freq.
A	15 seconds	900 MHz
B	20 seconds	600 MHz

- **How many clock cycles does A execute?**

CPU time for a program = CPU clock cycle for a program / clock rate.

=> CPU clock cycle for a program = CPU time for a program * clock rate

- **How many clock cycles does B execute?**

$$20 \text{ seconds} \times 600 \times 10^6 \frac{\text{cycles}}{\text{second}} = 1.2 \times 10^{10} \text{ cycles}$$

Consider a program that runs in 10 seconds on computer, A, which has a 4 GHZ clock. We are trying to help a computer designer build a computer, B, which will run this program in 6 seconds. The designer has determined that a substantial increase in the clock rate is possible, but this increase will affect the rest of the CPU design, causing computer B to require 1.2 times as many clock cycles as computer A for this program. What clock rate should we tell the designer to target?

Answer:

Performance equation

$$\text{CPU time} = \frac{\text{clock cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{clock cycle}} = \frac{\text{clock cycles}}{\text{clock frequency}}$$

First step: find clock cycles executed by Computer A

$$\text{CPU Time}_A = 10 \text{ seconds} = \frac{\text{clock cycles}_A}{400 \times 10^6 \frac{\text{cycles}}{\text{second}}}$$

$$\text{clock cycles}_A = 10 \text{ seconds x } 400 \times 10^6 \frac{\text{cycles}}{\text{second}}$$

$$= 40 \text{ x } 10^9 \text{ cycles}$$

Answer: (continued)

Second step: find clock cycles executed by Computer B

$$\text{clock cycles}_B = 1.2 \times \text{clock cycles}_A = 4800 \times 10^6 \text{ cycles}$$

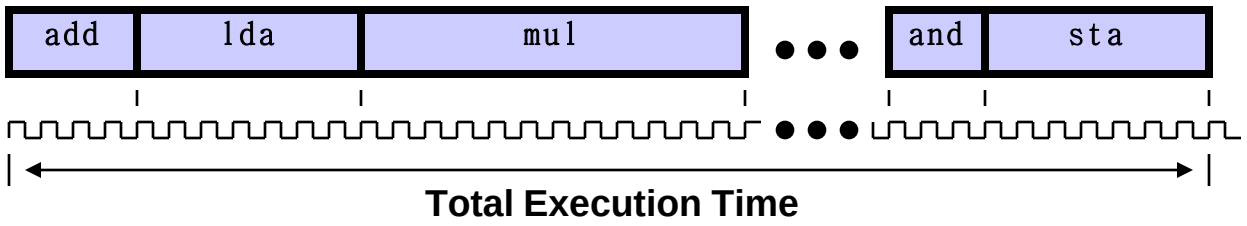
Third step: given clock cycles and CPU time, solve for clock rate of Computer B

$$\text{CPU Time}_B = 6 \text{ seconds} = \frac{4800 \times 10^6 \text{ cycles}}{\text{Clock rate}_B}$$

$$\text{Clock rate}_B = \frac{4800 \times 10^6 \text{ cycles}}{6 \text{ seconds}} = 800 \text{MHz}$$

Clock Cycles per Instruction (CPI)

- Consider the following instructions
  - ADDA - 3 cycles
  - MUL - 10 cycles
  - IDIV - 41 cycles
- More complex processors have other issues…
  - Pipelining - parallel execution, but sometimes stalls
  - Memory system issues: cache misses, page faults, etc.
- How can we combine these into an overall metric?



Definition:
Clock Cycles per Instruction (CPI)

- **Average number of clock cycles per instruction**
- **Measured for an entire program**

$$\text{CPI} = \text{Average Cycles per Instruction} = \frac{\#\ \text{of clock cycles}}{\#\ \text{of instructions}}$$

- => # of clock cycles = CPI * # of instructions.
- => execution time = # of clock cycles / clock rate
  = CPI * # of instructions / clock rate.

Example - CPI

Machine	Clock Cycles	Instructions
A	1.35×10^{10}	2.6×10^{9}
B	1.2×10^{10}	3.0×10^{9}

- **What is the CPI of A?**

- **What is the CPI of B?**

Definition - MIPS

- **MIPS - millions of instructions per second**

$$\text{MIPS} = \frac{\frac{\#\ \text{of instructions}}{\text{benchmark program}} \times \frac{\text{benchmark program}}{\text{execution time}}}{1{,}000{,}000}$$

- **Once used as a general metric for performance**
  - **But, not useful for comparing different architectures**
  - **Often ridiculed as "meaningless indicator of performance"**

Example - MIPS

Machine	Execution Time	Instructions
A	15 seconds	2.6×10^9
B	20 seconds	3.0×10^9

- **What is the MIPS of A?**
- **What is the MIPS of B?**

Assume an instruction set implemented in 2 architectures:

For some program, Computer A has:

clock cycle time = 250 ps

CPI = 2.0

For the same program, Computer B has:

clock cycle time = 500 ps

CPI = 1.2

Which computer is faster?

Answer:

Which is faster ? … Clearly A is faster:

$$\frac{\text{CPU performance A}}{\text{CPU performance B}} = \frac{\text{Execution time B}}{\text{Execution time A}}$$

$$= \frac{600 \times I\,ps}{500 \times I\,ps} = 1.2 \text{ times}$$

Relating the Metrics - The Performance Equation

- **The "Iron Law" of Performance:** **a complete and reliable measure of computer performance is time.**

$$\text{CPU time} = \frac{\#\,\text{instructions}}{\text{program}} \times \frac{\text{clock cycles}}{\text{instruction}} \times \frac{\text{seconds}}{\text{clock cycle}}$$

So, to improve performance (everything else being equal) you can either *increase* or *decrease*?

decrease the # of required cycles for a program, or
decrease the clock cycle time or, said another way,
increase the clock rate.

Self Test !: Clock Cycles and Performance - Example

- **Program runs on Computer A:**
  - **CPU Time: 10 seconds**
  - **Clock: 400MHz**
- **Computer B can run clock faster**
  - **But, requires 1.2X clock cycles to perform same task**
  - **Desired CPU Time: 6 Seconds**
  - **What should the clock frequency be to reach this target?**
- **Key to approach: Performance equation**

$$\text{CPU time} = \frac{\text{clock cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{clock cycle}} = \frac{\text{clock cycles}}{\text{clock frequency}}$$

Remember:

CPU clock cycles = Instructions for a program

x Average clock cycles per Instruction (CPI)

CPU time = Instruction count x CPI x clock cycle time

$$CPU\ \ time = \frac{Instruction\ count \times CPI}{Clock\ rate}$$

Basic measurment at different levels in the computer and what is measured:

Components of Performance	Units of Measure
CPU execution time for a program	Seconds for the program
Instruction count	Instructions executed for the program
Clock Cycles per Instruction (CPI)	Average number of clock cycles per instruction
Clock cycle time	Seconds per clock cycle

More on CPU Clock Cycles

If a program contains different type of instruction using different clock cycle counts, then

$$CPU \text{ clock cycles} = \sum_{i=1}^{n} (CPU_i \times C_i)$$

CPU_i : the average number of cycles per instructions for that instruction class

C_i : the count of the number of instructions of class i executed.

n : the number of instruction classes.

Homework

- **Instruction Classes:**
  - **Add**
  - **Multiply**
- **Average Clock Cycles per Instruction:**
  - **Add 1 clock cycle**
  - **Mul 3 clock cycles**
- **Program A executed:**
  - **10 Add instructions**
  - **5 Multiply instructions**

What is the CPU clock cycle times?

What is the average CPI?

EXAMPLE

Comparing Code Segments

A compiler designer is trying to decide between two code sequences for a particular computer. The hardware designers have supplied the following facts:

	CPI for this instruction class		
	A	B	C
CPI	1	2	3

For a particular high-level-language statement, the compiler writer is considering two code sequences that require the following instruction counts:

Code sequence	Instruction counts for instruction class		
	A	B	C
1	2	1	2
2	4	1	1

Which code sequence executes the most instructions? Which will be faster? What is the CPI for each sequence?

ANSWER

Sequence 1 executes 2 + 1 + 2 = 5 instructions. Sequence 2 executes 4 + 1 + 1 = 6 instructions. So sequence 1 executes fewer instructions.

We can use the equation for CPU clock cycles based on instruction count and CPI to find the total number of clock cycles for each sequence:

$$\text{CPU clock cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times C_i)$$

This yields

$$\text{CPU clock cycles}_1 = (2 \times 1) + (1 \times 2) + (2 \times 3) = 2 + 2 + 6 = 10 \text{ cycles}$$

$$\text{CPU clock cycles}_2 = (4 \times 1) + (1 \times 2) + (1 \times 3) = 4 + 2 + 3 = 9 \text{ cycles}$$

ANSWER

So code sequence 2 is faster, even though it actually executes one extra instruction. Since code sequence 2 takes fewer overall clock cycles but has more instructions, it must have a lower CPI. The CPI values can be computed by

$$\text{CPI} = \frac{\text{CPU clock cycles}}{\text{Instruction count}}$$

$$\text{CPI}_1 = \frac{\text{CPU clock cycles}_1}{\text{Instruction count}_1} = \frac{10}{5} = 2$$

$$\text{CPI}_2 = \frac{\text{CPU clock cycles}_2}{\text{Instruction count}_2} = \frac{9}{6} = 1.5$$

The above example shows the danger of using only one factor (instruction count) to assess performance. When comparing two computers, you must look at all three components, which combine to form execution time.

Performance Tradeoffs

- **Program Instruction count - impacted by**
  - **Instructions available to perform basic tasks (architecture)**
  - **Quality of compiler code generator**
- **CPI - impacted by**
  - **Chip implementation**
  - **Quality of compiler code generator**
  - **Memory system performance**
- **Clock Rate**
  - **Delay characteristics of IC technology**
  - **Logical structure of implementation**

Performance Outline

- Motivation
- Defining Performance
- Common Performance Metrics
- Benchmarks
- Amdahl's Law

Benchmarks - Programs to Evaluate Performance

- The book defines performance in terms of a specific program
- But, which program should you use?
  - Ideally, "real" programs
  - Ideally, programs you will use
  - But what if you don't know or don't have time to find out?
- Alternative - Benchmark Suites

Benchmark Suites

- Use a collection of small programs
- Summarize performance … how?
  - Total Execution Time
  - Arithmetic Mean
  - Weighted Arithmetic Mean
  - Geometric Mean

Total Execution Time

- **Suppose we have two benchmarks:**

	Computer A	Computer B
Program 1 (seconds)	1	10
Program 2 (seconds)	1000	100
Total time (seconds)	1001	110

- **How do we compare computers A and B?**
  - **A is 10 times faster than B for Program 1**
  - **B is 10 times faster than A for Program 2**
- **Summarizing performance: total execution time**

$$\frac{\text{Performance}_B}{\text{Performance}_A} = \frac{\text{Execution time}_A}{\text{Execution time}_B} = \frac{1001}{110} = 9.1$$

- **Reasonable comparison for even workload**

Summarizing Performance

- **Arithmetic Mean** $\text{AM} = \frac{1}{n}\sum_{i=1}^{n}\text{Time}_i$
- **Example:**

$$\text{AM}_A = \frac{1}{2}(1+1000) = 500.5$$

$$\text{AM}_B = \frac{1}{2}(10+100) = 55$$

- **A smaller mean indicates a smaller average execution time; and thus improved performance.**
- **Weighted Arithmetic Mean - use when some programs run more often than others**

$$\text{WAM} = \sum_{i=1}^{n}\text{Weight}_i \times \text{Time}_i$$

- **Example:**
  - **Program 1 is 80% of workload** $\text{WAM}_A = 0.8\times 1 + 0.2\times 1000 = 200.8$
  - **Program 2 is 20% of workload** $\text{WAM}_B = 0.8\times 10 + 0.2\times 100 = 28$

The SPEC Benchmark Suite

- System Performance Evaluation Corporation
  - Founded by workstation vendors with goal of realistic, standardized performance test
  - Philosophy: fair comparison between real systems
  - Multiple versions - most recent is SPEC2006
- Basic approach
  - Measure execution time of several small programs
  - Normalize each to performance on a reference machine
  - Combine performances using geometric mean

$$\text{Exec. Time Ratio} = \frac{\text{Exec. Time}_{ref}}{\text{Exec. Time}_{test}} \qquad \text{GM} = \sqrt[n]{\prod_{i=1}^{n} \text{Exec. Time Ratio}}$$

More about SPEC

- Separate integer & floating point suites
  - SPECint - integer performance
  - SPECfp - floating point performance
- Several Versions
  - SPEC89 - initial version
  - SPEC95 - see Figure 2.6 in book
  - SPEC CPU 2000 - current
    - CINT 2000
    - CFP 2000

Some Not-so-Recent SPEC Data

- **Some SPEC95 numbers:**

Processor	Clock	SPECint95	SPECfp95
Intel Pentium II	266 MHz	9.5	6.4
Intel Pentium III	734 MHz	35.6	30.4
AMD Athlon K7	800 MHz	35.0	26.1
Motorola PPC G4	450 MHz	21.4	20.4
Sun Sparc Ultra III	600 MHz	35	60
Compaq Alpha 21264	667 MHz	44	66

Source: Berkeley CPU Information Center

- **SPEC2000 Results - see `http://www.spec.org`**

Performance Outline

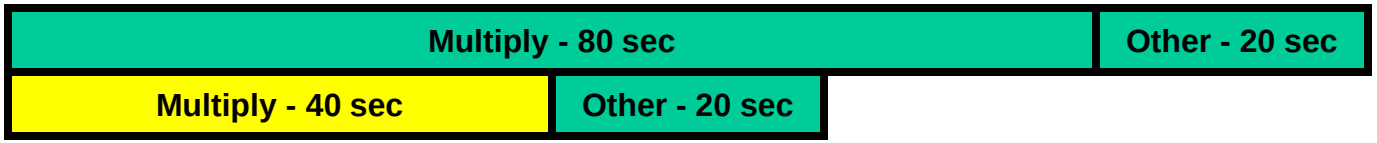
- Motivation
- Defining Performance
- Common Performance Metrics
- Benchmarks
- Amdahl's Law

Amdahl's Law

- **Improving one part of performance by a factor of N doesn't increase overall performance by N**

$$\text{Execution time after improvement} = \left(\frac{\text{Execution time affected by improvement}}{\text{Amount of improvement}} + \text{Execution Time Unaffected} \right)$$

- **Book example: Suppose a program executes in 100 seconds where:**
  - **80 seconds are spent performing multiply operations**
  - **20 seconds are spent performing other operations**
- **What happens if we speed up multipy *n* times?**



Amdahl's Law Example (cont'd)

- **Execution time after speeding up multiply:**

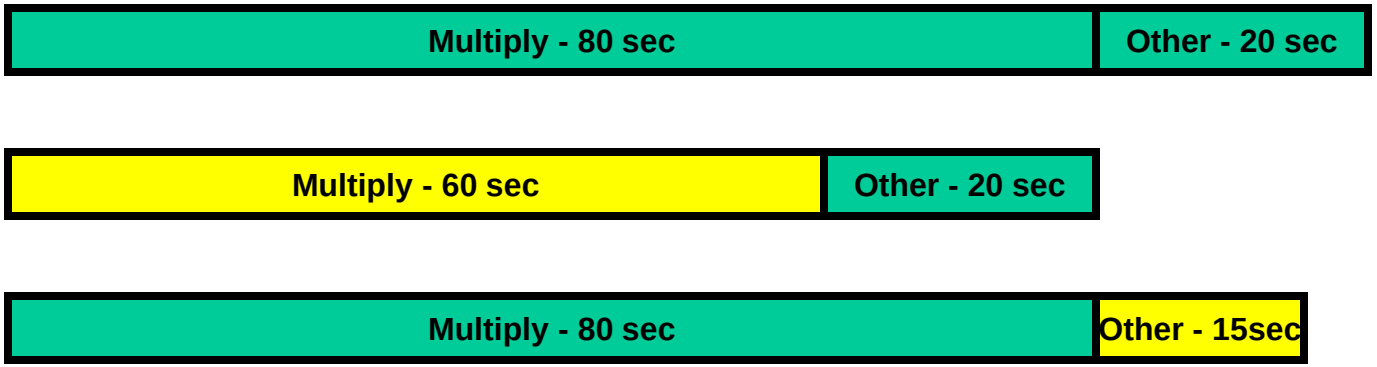
$$\text{Execution time after improvement} = \left(\frac{\text{Execution time affected by improvement}}{\text{Amount of improvement}} + \text{Execution Time Unaffected} \right)$$

$$\text{Execution time after improvement} = \left(\frac{80 \text{ seconds}}{n} + 20 \text{ seconds} \right)$$

- **Bottom line: no matter what we do to multiply, execution time will always be >20 seconds!**

Amdahl's Law Corollary

- **Make the common case fast**
  - In our example, biggest gains when we speed up multiply
  - Speeding up "other instructions" is not as valuable



Amdahl's Law

What about this!

How much do we have to improve the speed of multiplication if we want the program to run 4 times faster?"

Amdahl's Law

Execution time after improvement

$$= \frac{80\ \text{seconds}}{\text{n}} + (100 - 80\ \text{seconds})$$

$$\textit{Expected}\quad \textit{Execution}\ \text{time after improvement} = \frac{100}{4} = 25$$

$$= \frac{80}{n} + 20, \qquad \text{n} = 16$$

MIPS (millions instructions per second)

$$MIPS = \frac{\text{Instruction count}}{\text{Execution time} \times 10^6}$$

Code from	Instruction Counts (in billions) for each instruction set		
	A (1 CPI)	B (2 CPI)	C (3 CPI)
Compiler 1	5	1	1
Compiler 2	10	1	1

Clock rate = 4GHz A,B,C : Instruction Classes

- **Which code sequence will execute faster according to MIPS?**
- **According to execution time?**

Execution time & MIPS (1)

CPU clock cycles1 = (5 x 1+1 x 2+1 x 3) x 10^9 = 10 x 10^9

CPU clock cycles2 = (10 x 1+1 x 2+1 x 3) x 10^9 = 15 x 10^9

$$\text{Execution time1} = \frac{10 \times 10^9}{4 \times 10^9} = 2.5 \text{seconds}$$

$$\text{Execution time2} = \frac{15 \times 10^9}{4 \times 10^9} = 3.75 \text{ seconds}$$

Execution time & MIPS (2)

$$MIPS_1 = \frac{(5+1+1)\times 10^9}{2.5\,\text{seconsd}\times 10^6} = 2800$$

$$MIPS_2 = \frac{(10+1+1)\times 10^9}{3.75\times 10^6} = 3200$$

So, the code from compiler 2 has higher MIPS rating, but the code from compiler 1 runs faster!!

→ This example shows how MIPS can fail to give a true picture of performance!!!

Roadmap for the term: major topics

- Computer Systems Overview
- Technology Trends
- Instruction sets (and Software)
- Logic & Arithmetic
- Performance
- Processor Implementation
- Memory Systems
- Input/Output