CprE 381: Computer Organization and Assembly Level Programming

Performance

Henry Duwe
Electrical and Computer Engineering
Iowa State University

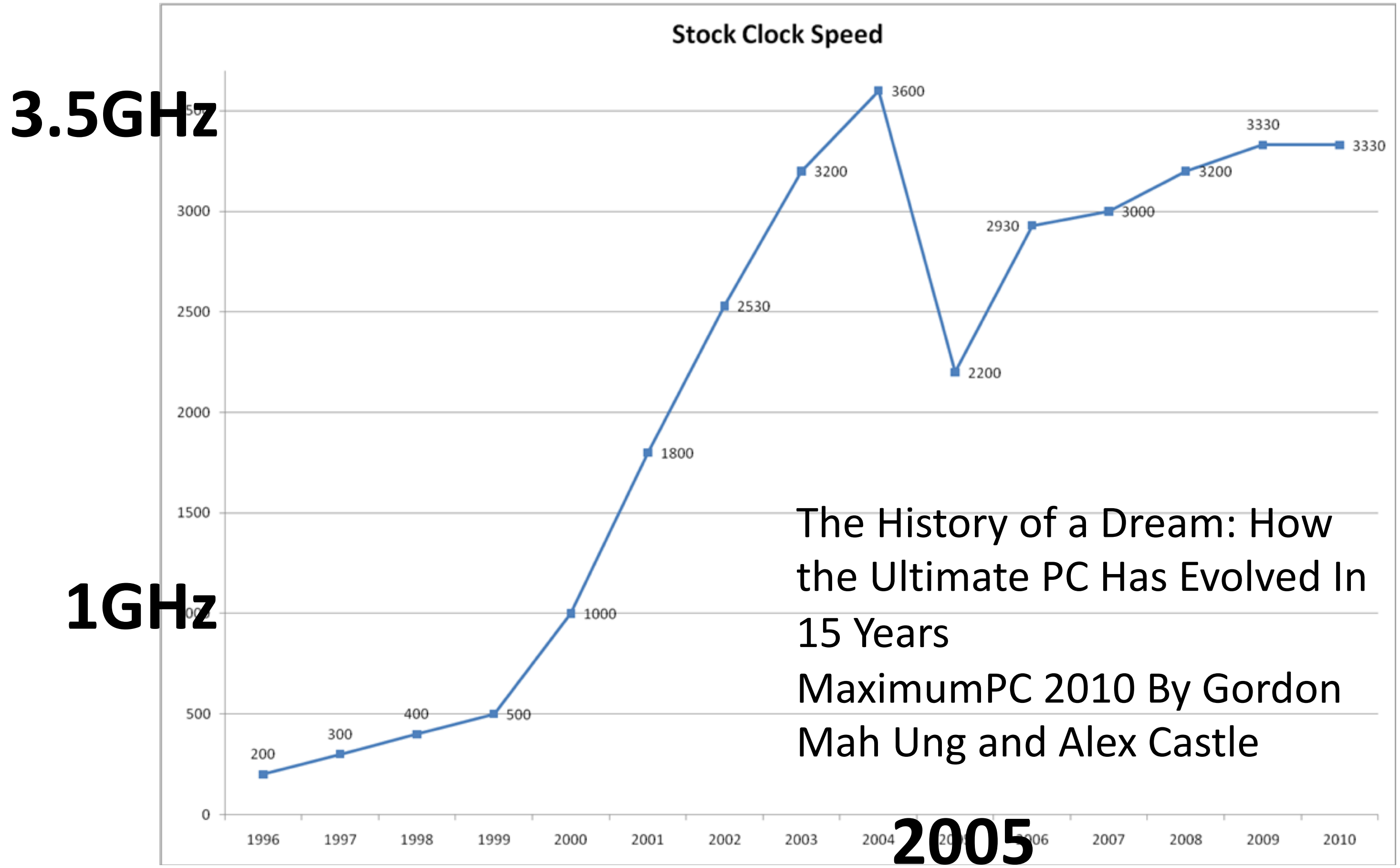
Administrative

- HW5 due Mar 4
- Term Project
  - Part 2 Posted → 10% of Final Course Grade

Defining Performance

But this one has more googlihertz!

Defining Performance



- Did Intel start selling less performant processors?

Computer Performance Metrics

- 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 many queries per minute?

- If we upgrade a machine with a new processor what do we increase?
- If we add a new machine to the lab what do we increase?

Execution Time Performance

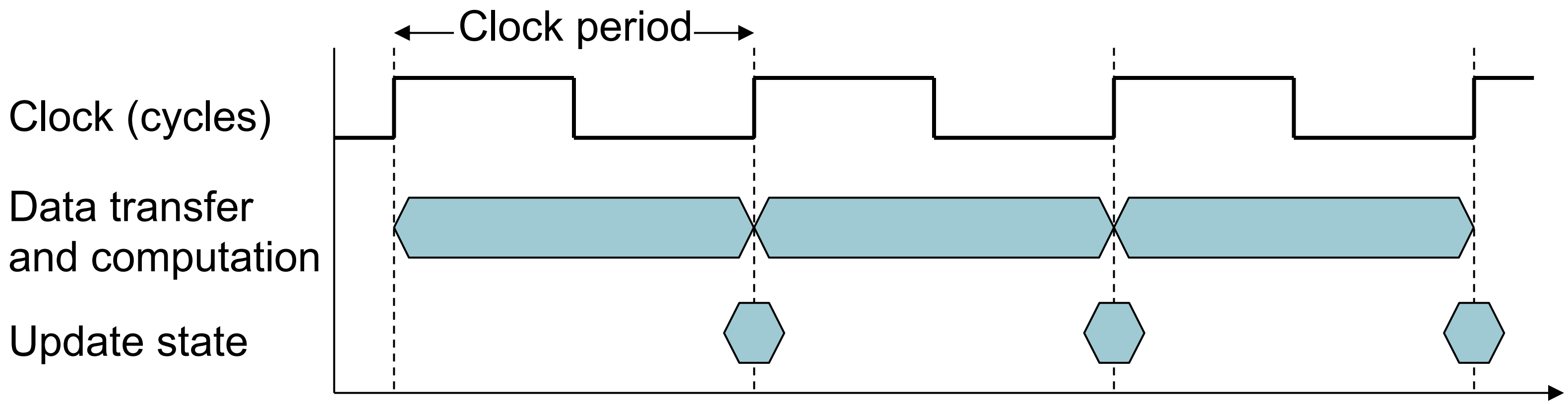
- Elapsed Time
  - Counts everything (disk and memory accesses, I/O , etc.)
  - A useful number, but often not good for processor comparison purposes
    - E.g., OS & multiprogramming time and non-determinism make it difficult to compare CPUs
- CPU time
  - Doesn't count I/O or time spent running other 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 target program(s)
  - Includes arithmetic, memory, and control instructions...

Clock Cycles

- Instead of reporting execution time in seconds, we often use cycles

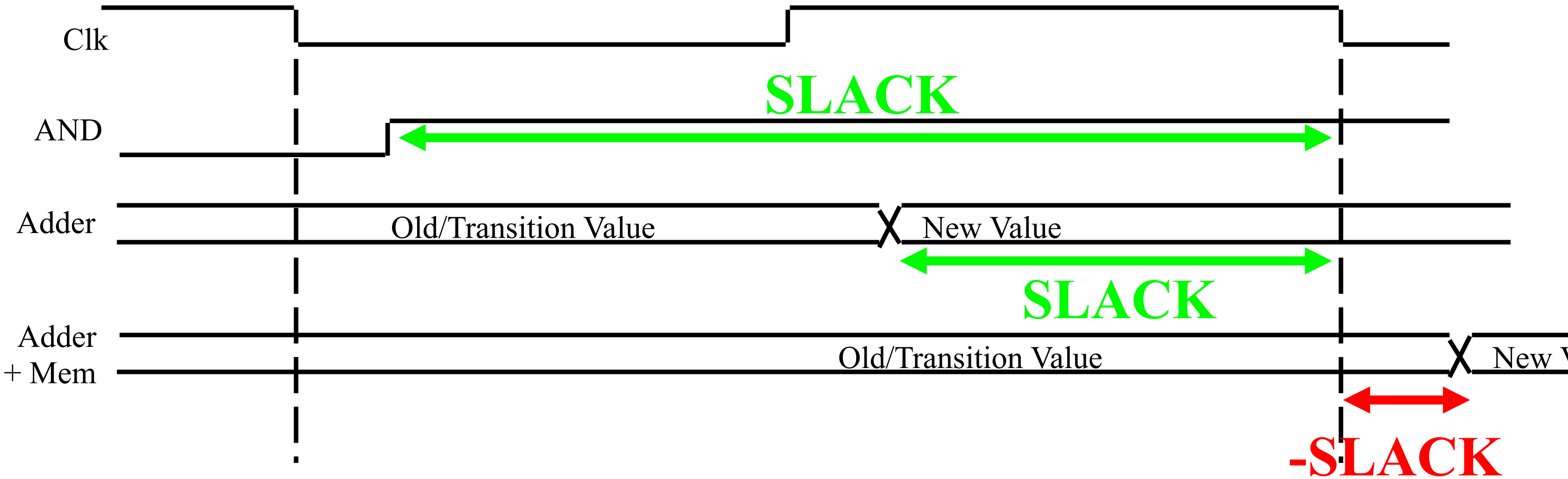
$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycles}}$$

- Clock "ticks" indicate when to start activities:



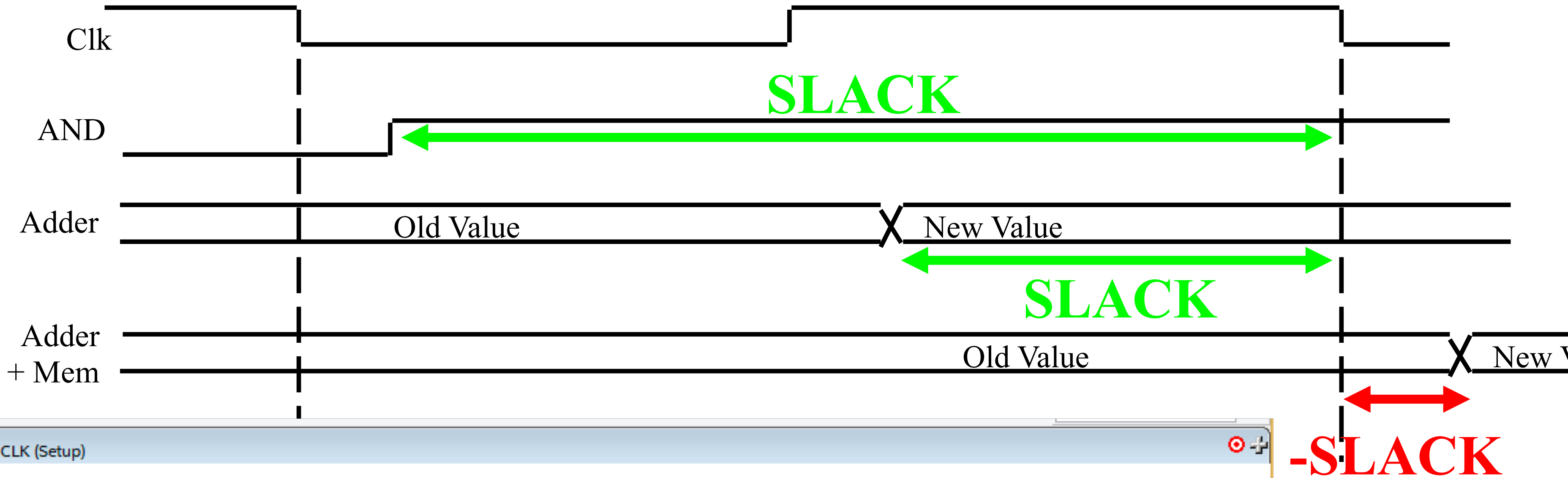
- Cycle time = time between ticks = seconds per cycle
- Clock rate (frequency) = cycles per second (1 Hz. = 1 cycle/sec)
- A 2 GHz clock has a $\frac{1}{2\times 10^9}\times 10^{12} = 500\,\text{picoseconds}\,(\text{ps})$ cycle time

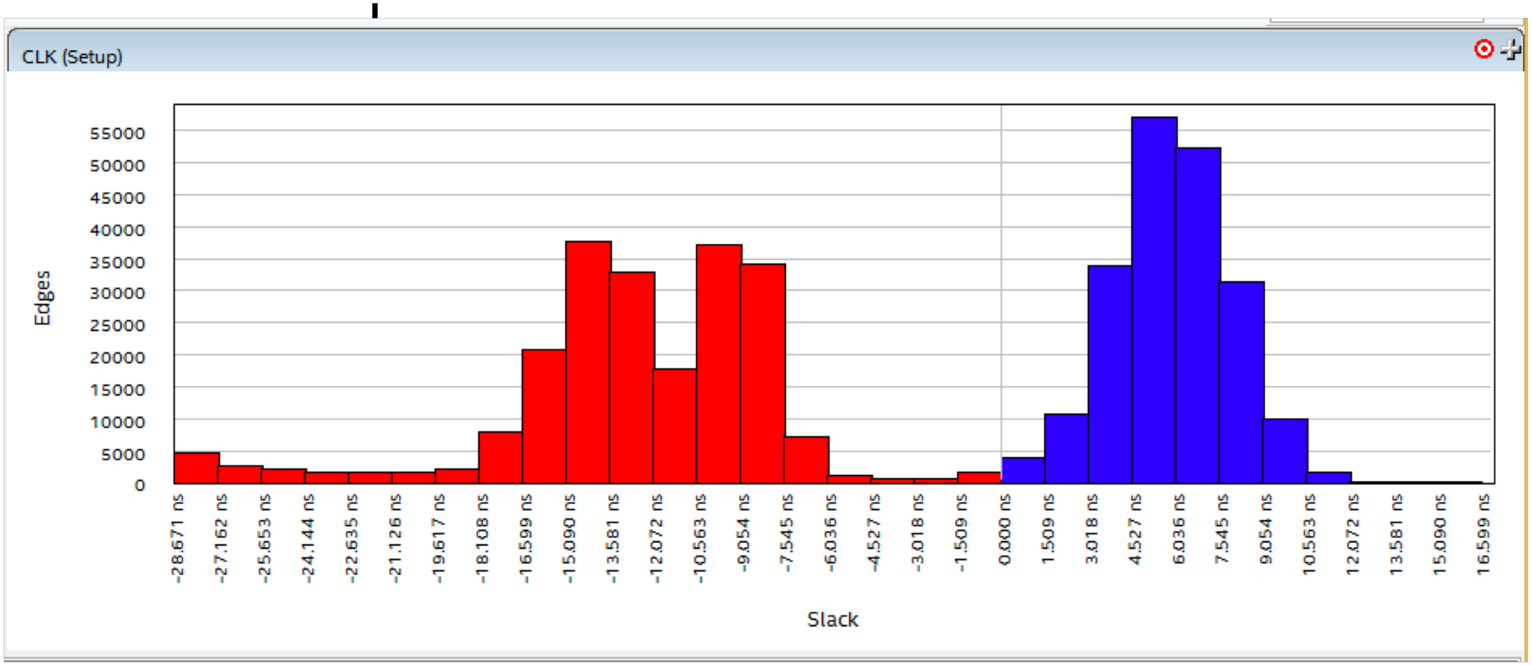
Slack



Clk
AND
SLACK
Adder
Old/Transition Value
New Value
SLACK
Adder
+ Mem
Old/Transition Value
New V
-SLACK

Slack


Clk
AND
SLACK
Adder
Old Value
New Value
SLACK
Adder + Mem
Old Value
New V
-SLACK


CLK (Setup)
Edges
55000
50000
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
-28.671 ns
-27.162 ns
-25.653 ns
-24.144 ns
-22.635 ns
-21.126 ns
-19.617 ns
-18.108 ns
-16.599 ns
-15.090 ns
-13.581 ns
-12.072 ns
-10.563 ns
-9.054 ns
-7.545 ns
-6.036 ns
-4.527 ns
-3.018 ns
-1.509 ns
0.000 ns
1.509 ns
3.018 ns
4.527 ns
6.036 ns
7.545 ns
9.054 ns
10.563 ns
12.072 ns
13.581 ns
15.090 ns
16.599 ns
Slack

Now That We Understand Clock Cycles

- A given program will require
  - Some number of instructions (machine instructions)
  - Some number of cycles
  - Some number of seconds

- We have a vocabulary that relates these quantities:
  - Cycle time or cycle period (seconds per cycle)
  - Clock rate or frequency (cycles per second)
  - CPI (cycles per instruction)
  - IPC (instructions per cycle)
  - MIPS (millions of instructions per second)

Measuring # of Clock Cycles

- Clock cycles/program is not an intuitive or easily determined value, so

$$\text{Clock Cycles} = \text{Instructions} \times \text{Clock Cycles per Instruction}$$

- Cycles Per Instruction (CPI) used often

- CPI is an **average** since the number of cycles per instruction varies from instruction to instruction
  - Average depends on instruction mix, latency of each inst. type etc.

- CPIs can be used to compare two implementations of the same ISA, but is not useful alone for comparing different ISAs
  - An x86 add is different from a MIPS add

Using CPI

- Drawing on the previous equation:

$$\boldsymbol{Execution\,Time = \#\,Instructions \times \frac{Cycles}{Instruction} \times \frac{Seconds}{Cycle}}$$

- To improve performance (i.e., reduce execution time)
  - Increase clock rate (decrease clock cycle time) OR
  - Decrease CPI OR
  - Reduce the number of instructions
- Designers balance cycle time against the number of cycles required
  - Improving one factor may make the other one worse...

Using CPI

- Drawing on the previous equation:

$$Execution\ Time = \#\ Instructions \times \frac{Cycles}{Instruction} \times \frac{Seconds}{Cycle}$$

In-class Assessment!

Access Code: Important

Note: sharing access code to those outside of classroom or using access while outside of classroom is considered cheating

- – Reduce the number of instructions
- Designers balance cycle time against the number of cycles required
  - – Improving one factor may make the other one worse...

Evaluating Performance

- Performance best determined by running a real application
  - Use programs typical of expected workload
  - Or, typical of expected class of applications
    - Compilers/editors, scientific applications, graphics, etc.
- Small benchmarks
  - Nice for architects and designers
  - Easy to standardize
  - Can be abused
- SPEC (System Performance Evaluation Cooperative)
  - Companies have agreed on a set of real program and inputs
  - Valuable indicator of performance (and compiler technology)
  - Can still be abused

CINT2006 for Intel Core i7-920

Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^{-9})	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2252	0.60	0.376	508	9770	19.2
Block-sorting compression	bzip2	2390	0.70	0.376	629	9650	15.4
GNU C compiler	gcc	794	1.20	0.376	358	8050	22.5
Combinatorial optimization	mcf	221	2.66	0.376	221	9120	41.2
Go game (AI)	go	1274	1.10	0.376	527	10490	19.9
Search gene sequence	hmmer	2616	0.60	0.376	590	9330	15.8
Chess game (AI)	sjeng	1948	0.80	0.376	586	12100	20.7
Quantum computer simulation	libquantum	659	0.44	0.376	109	20720	190.0
Video compression	h264avc	3793	0.50	0.376	713	22130	31.0
Discrete event simulation library	omnetpp	367	2.10	0.376	290	6250	21.5
Games/path finding	astar	1250	1.00	0.376	470	7020	14.9
XML parsing	xalancbmk	1045	0.70	0.376	275	6900	25.1
Geometric mean	–	–	–	–	–	–	25.7

Clock Rate != Performance

- Mobile Intel Pentium 4 Vs Intel Pentium M
  - 2.4 GHz 1.6 GHz

- Performance on Mobilemark with same memory and disk
  - Word, excel, photoshop, powerpoint, etc.
  - Pentium M takes only 15% longer

- What is the relative CPI?

Clock Rate != Performance

- Mobile Intel Pentium 4 Vs Intel Pentium M
  - 2.4 GHz 1.6 GHz

- Performance on Mobilemark with same memory and disk
  - Word, excel, photoshop, powerpoint, etc.
  - Pentium M takes only 15% longer

- What is the relative CPI?
  - ExecTime = IC • CPI/Clock rate
  - IC • CPIM/1.6 = 1.15 • IC • CPI4/2.4
  - CPI4/CPIM = 2.4/(1.15•1.6) = 1.3

CPI Varies

- Different instruction types require different numbers of cycles
- CPI is often reported for types of instructions

$$\text{Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{IC}_i)$$

- Where CPI_i is the CPI for the type of instructions and IC_i is the count of that type of instruction

Computing CPI

- To compute the overall average CPI use

$$\text{CPI} = \sum_{i=1}^{n} \left(\text{CPI}_i \times \frac{\text{Instruction Count}_i}{\text{Instruction Count}} \right)$$

Frequency of Instruction Type

Computing CPI Example

Instruction Type	CPI	Frequency	CPI * Frequency
ALU	1	50%	0.5
Branch	2	20%	0.4
Load	2	20%	0.4
Store	2	10%	0.2

- Given this machine, the CPI is the sum of CPI × Frequency
- Average CPI is 0.5 + 0.4 + 0.4 + 0.2 = 1.5
- What fraction of the time for data transfer?

CPI Question

Instruction Type	CPI	Frequency	CPI * Frequency
ALU	1	50%	0.5
Branch	2	20%	0.4
Load	2	20%	0.4
Store	2	10%	0.2

- What is the impact of the MIPS displacement based memory addressing mode?
- Assume 50% of MIPS loads and stores have a zero displacement

CPU Time versus MIPS

- Two different compilers are being tested for a 1 GHz. machine with three different classes of instructions:
  - Class A, Class B, and Class C,
  - Require 1, 2, and 3 cycles (respectively)
  - Both compilers are used to produce code for a large piece of software

- The first compiler's code uses 5 million Class A instructions, 1 million Class B instructions, and 1 million Class C instructions

- The second compiler's code uses 10 million Class A instructions, 1 million Class B instructions, and 1 million Class C instructions

- Which sequence will be faster according to MIPS?
- Which sequence will be faster according to execution time?

Speedup

- Speedup allows us to compare different CPUs or optimizations

$$\text{Speedup} = \frac{\text{CPUtimeOld}}{\text{CPUtimeNew}}$$

- Example
  - Original CPU takes 2sec to run a program
  - New CPU takes 1.5sec to run a program
  - Speedup = 1.33x or 33%

Amdahl's Law

- If an optimization improves a fraction *f* of execution time by a factor of *a*

$$Speedup = \frac{Told}{[(1-f)+f/a]\times Told} = \frac{1}{(1-f)+f/a}$$

Gene Amdahl

- This formula is known as Amdahl's Law
- Lessons from
  - If $f \rightarrow 100\%$, then speedup = *a*
  - If $a \rightarrow \infty$, the speedup = $1/(1-f)$
- Summary
  - Make the common case fast
  - Watch out for the non-optimized/optimizable component

Amdahl’s Law Example

- Suppose a program runs in 100 seconds on a machine, with multiply responsible for 80 seconds of this time. How much do we have to improve the speed of multiplication if we want the program to run 4 times faster?

In-class Assessment!

Access Code: Wafer-scale

Note: sharing access code to those outside of classroom or using access while outside of classroom is considered cheating

•

Amdahl's Law Example

- Suppose a program runs in 100 seconds on a machine, with multiply responsible for 80 seconds of this time. How much do we have to improve the speed of multiplication if we want the program to run 4 times faster?

- How about making it 5 times faster?

Performance Summary

- Performance is specific to a particular program(s)
  - Total execution time is a consistent summary of performance

- For a given architecture performance increases come from:
  - Increases in clock rate (without adverse CPI affects)
  - Improvements in processor organization that lower CPI
  - Compiler enhancements that lower CPI and/or instruction count
  - Algorithm/Language choices that affect instruction count

- Pitfall: expecting improvement in one (limited) aspect of a machine's performance to affect the total performance

Acknowledgments

- These slides contain material developed and copyright by:
  - Joe Zambreno (Iowa State)
  - David Patterson (UC Berkeley)
  - Mary Jane Irwin (Penn State)
  - Christos Kozyrakis (Stanford)
  - Onur Mutlu (Carnegie Mellon)
  - Krste Asanović (UC Berkeley)