

Reasoning about Computer Performance

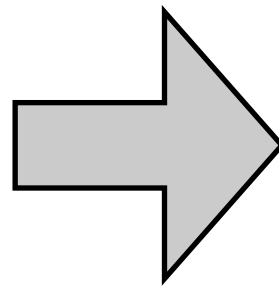
Jason Mars

What Do We Want in our Computers

- Frame rate
 - Responsiveness
 - Real-time
 - Throughput
 - Latency/Execution time
-
- Battery life
 - Low power/low temperature

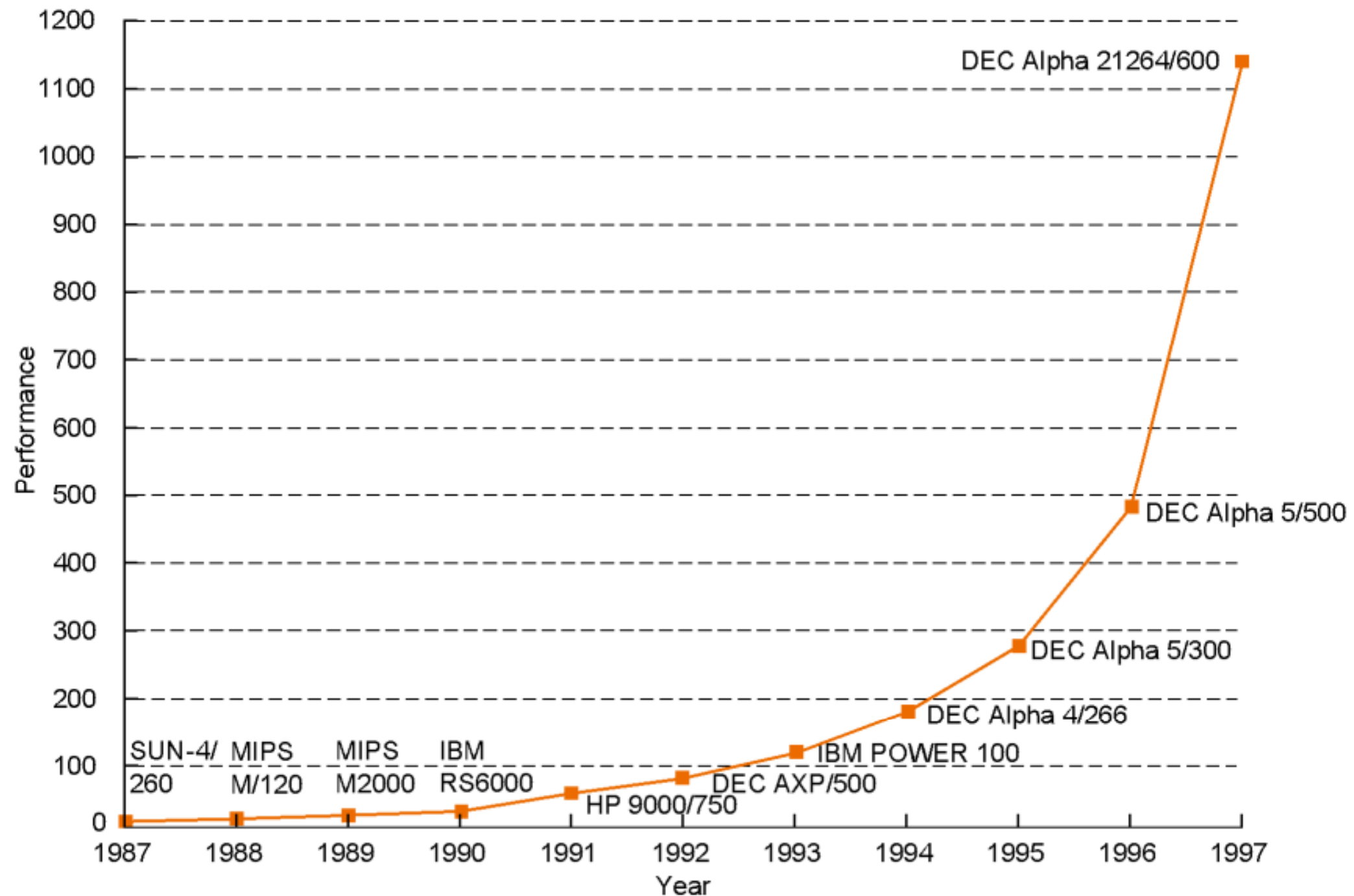
What Do We Want in our Computers

- Frame rate
- Responsiveness
- Real-time
- Throughput
- Latency/Execution time
- Battery life
- Low power/low temperature



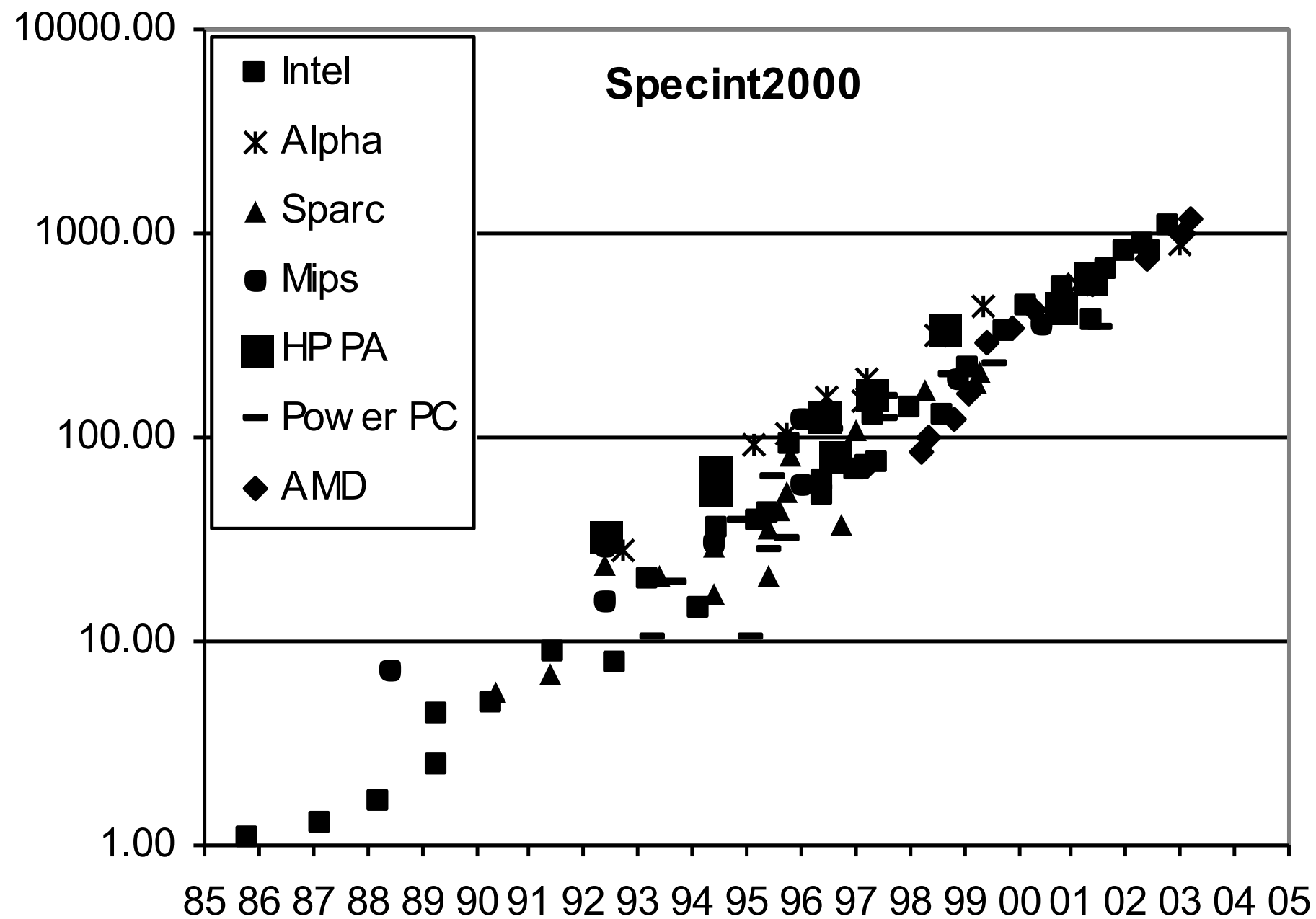
Performance!!!!

Performance Trends

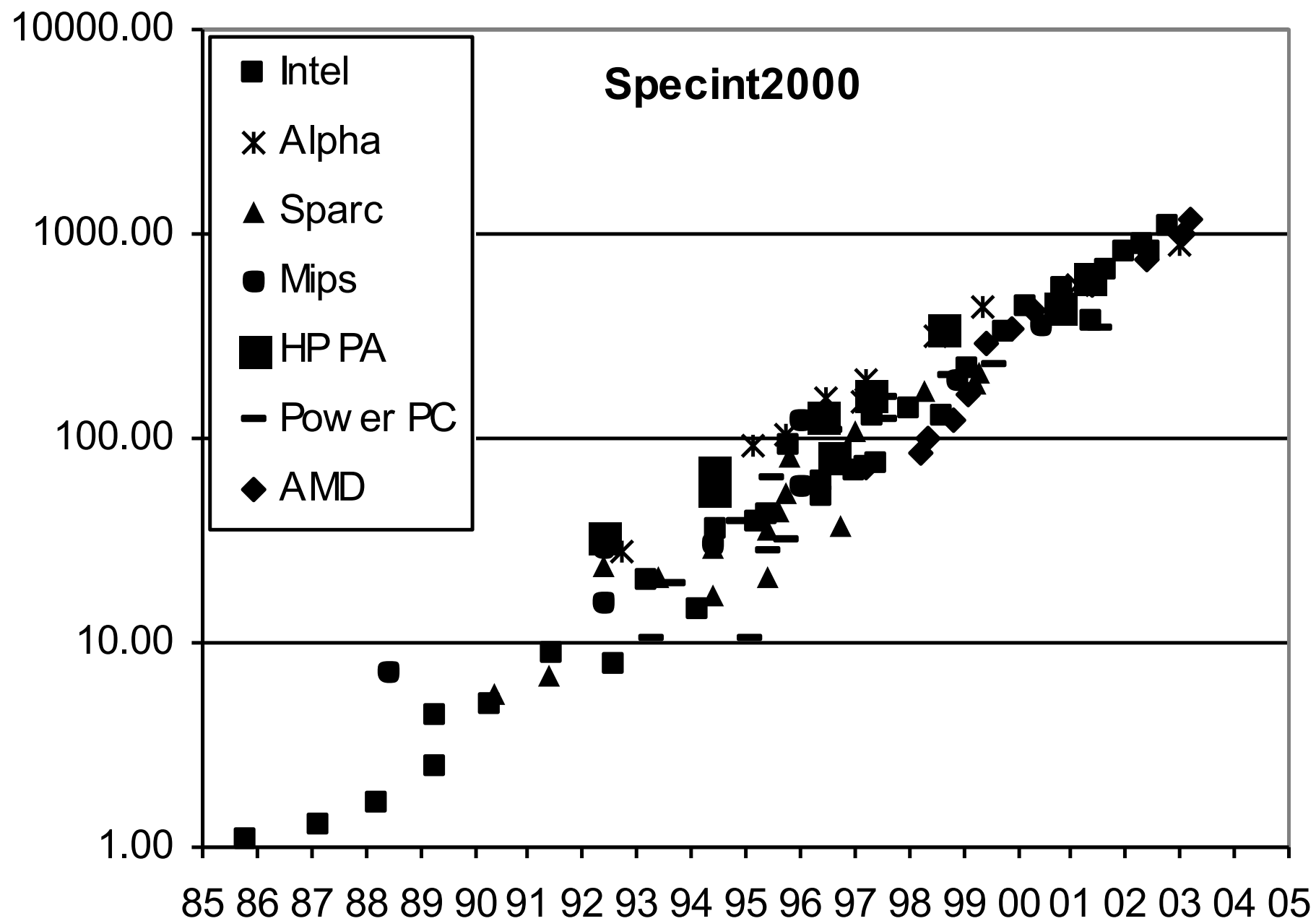


Performance grows at exponential rate

Performance Trends



Performance Trends



But *what is performance?*

What is Performance

Car	Time to Bay Area	Speed	Passengers	Throughput (pmph)
Ferrari	3.1 hours	160 mph	2	320
Greyhound	7.7 hours	65 mph	60	3900

What is Performance

Car	Time to Bay Area	Speed	Passengers	Throughput (pmph)
Ferrari	3.1 hours	160 mph	2	320
Greyhound	7.7 hours	65 mph	60	3900

- Time to do the task
 - **execution time**, response time, latency
- Tasks per day, hour, week, sec, ns. ..
 - **throughput**, bandwidth

How to Measure Execution Time?

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

How to Measure Execution Time?

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

- Wall-clock time?
- user CPU time?
- user + kernel CPU time?
- **Answer:**

How to Measure Execution Time?

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

- Wall-clock time?
- user CPU time?
- user + kernel CPU time?
- **Answer:** 103.6s

But What *is* Performance

$$\text{Performance}_X = \frac{1}{\text{Execution Time}_X} \quad , \text{ for program } X$$

But What *is* Performance

$$\text{Performance}_X = \frac{1}{\text{Execution Time}_X} \quad , \text{ for program } X$$

- only has meaning in the context of a **program** or **workload**

But What *is* Performance

$$\text{Performance}_X = \frac{1}{\text{Execution Time}_X} \quad , \text{ for program } X$$

- only has meaning in the context of a **program** or **workload**
- Not very intuitive as an absolute measure, but most of the time we're more interested in **relative performance**.

Relative Performance

- Can be confusing
 - A runs in 12 seconds
 - B runs in 20 seconds
 - $A/B = .6$, so A is 40% faster, or 1.4X faster, or B is 40% slower
 - $B/A = 1.67$, so A is 67% faster, or 1.67X faster, or B is 67% slower
- For the future scientists
 - Must use GEOMEAN for performance!
- Needs a precise definition

Defining Speedup (Relative Performance)

$$\text{Speedup (X/Y)} = \frac{\text{Performance}_X}{\text{Performance}_Y} = \frac{\text{Execution Time}_Y}{\text{Execution Time}_X} = n$$

Example

- Machine A runs program C in 9 seconds, Machine B runs the same program in 6 seconds. What is the speedup we see if we move to Machine B from Machine A?
- Machine B gets a new compiler, and can now run the program in 3 seconds. Speedup?

Example

- Machine A runs program C in 9 seconds, Machine B runs the same program in 6 seconds. What is the speedup we see if we move to Machine B from Machine A?

Speedup (X/Y) =

- Machine B gets a new compiler, and can now run the program in 3 seconds. Speedup?

Example

- Machine A runs program C in 9 seconds, Machine B runs the same program in 6 seconds. What is the speedup we see if we move to Machine B from Machine A?

X = machine B

Speedup (X/Y) =

Y = machine A

- Machine B gets a new compiler, and can now run the program in 3 seconds. Speedup?

Example

- Machine A runs program C in 9 seconds, Machine B runs the same program in 6 seconds. What is the speedup we see if we move to Machine B from Machine A?

$$\begin{array}{l} \text{X = machine B} \\ \text{Speedup (X/Y) = } \frac{\text{Execution Time}_Y}{\text{Execution Time}_X} = n \end{array} \quad \begin{array}{l} 9 \\ 6 \end{array} \quad \text{1.5x}$$

- Machine B gets a new compiler, and can now run the program in 3 seconds. Speedup?

Example

- Machine A runs program C in 9 seconds, Machine B runs the same program in 6 seconds. What is the speedup we see if we move to Machine B from Machine A?

$$\begin{array}{l} \text{X = machine B} \\ \text{Speedup (X/Y) = } \frac{\text{Execution Time}_Y}{\text{Execution Time}_X} = n \end{array} \quad \begin{array}{c} 9 \\ 6 \end{array} \quad \text{1.5x}$$

- Machine B gets a new compiler, and can now run the program in 3 seconds. Speedup?

$$\begin{array}{l} \text{X = machine B} \\ \text{Speedup (X/Y) = } \\ \text{Y = machine A} \end{array}$$

Example

- Machine A runs program C in 9 seconds, Machine B runs the same program in 6 seconds. What is the speedup we see if we move to Machine B from Machine A?

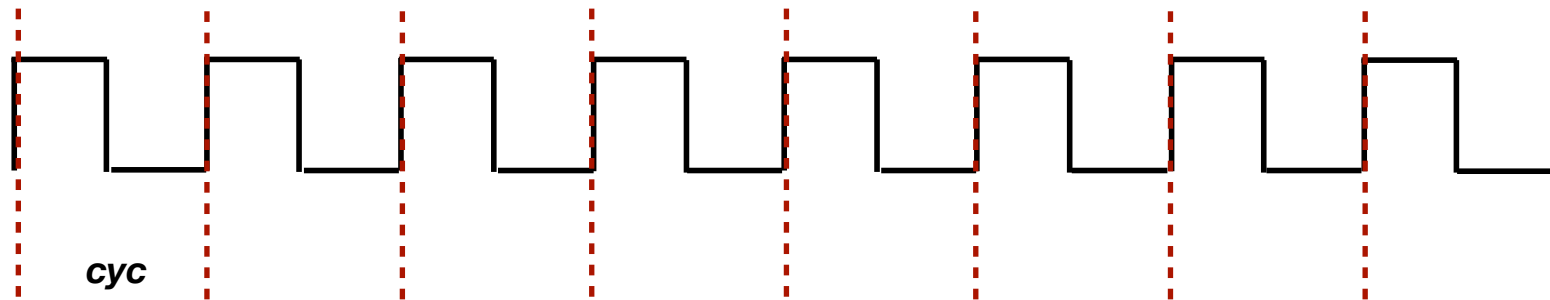
$$\begin{array}{l} \text{X = machine B} \\ \text{Speedup (X/Y)} = \frac{\text{Execution Time}_Y}{\text{Execution Time}_X} = n \end{array} \quad \begin{array}{c} 9 \\ 6 \end{array} \quad \text{1.5x}$$

- Machine B gets a new compiler, and can now run the program in 3 seconds. Speedup?

$$\begin{array}{l} \text{X = machine B} \\ \text{Speedup (X/Y)} = \frac{\text{Execution Time}_Y}{\text{Execution Time}_X} = n \end{array} \quad \begin{array}{c} 9 \\ 3 \end{array} \quad \text{3x}$$

What is Time?

- CPU Execution Time = CPU clock cycles * Clock cycle time
 - Every conventional processor has a clock with an associated clock cycle time or clock rate
 - Every program runs in an integral number of clock cycles



- Cycle Time
 - MHz = millions of cycles/second, GHz = billions of cycles/second
 - X MHz = 1000/X nanoseconds cycle time
 - Y GHz = 1/Y nanoseconds cycle time

Reasoning About Execution Time via Cycles

- Number of CPU cycles = Instructions executed * Average Clock Cycles per Instruction (CPI)
- Usually measured in seconds
- Shorter execution time means better performance

Reasoning About Execution Time via Cycles

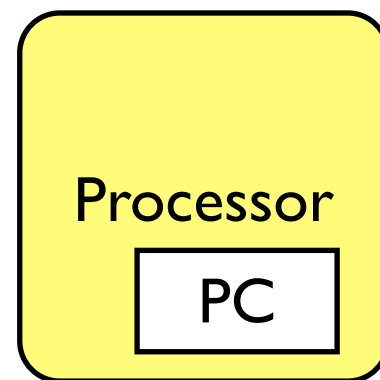
- Number of CPU cycles = Instructions executed * Average Clock Cycles per Instruction (CPI)
- Usually measured in seconds
- Shorter execution time means better performance

instruction memory

120007a30:	0f00bb27	ldah	gp,15(t12)
120007a34:	509cbd23	lda	gp,-25520(gp)
120007a38:	00005d24	ldah	t1,0(gp)
120007a3c:	0000bd24	ldah	t4,0(gp)
120007a40:	2ca422a0	ldl	t0,-23508(t1)
120007a44:	130020e4	beq	t0,120007a94
120007a48:	00003d24	ldah	t0,0(gp)
120007a4c:	2ca4e2b3	stl	zero,-23508(t1)
120007a50:	0004ff47	clr	v0
120007a54:	28a4e5b3	stl	zero,-23512(t4)
120007a58:	20a421a4	ldq	t0,-23520(t0)
120007a5c:	0e0020e4	beq	t0,120007a98
120007a60:	0204e147	mov	t0,t1
120007a64:	0304ff47	clr	t2
120007a68:	0500e0c3	br	120007a80

Reasoning About Execution Time via Cycles

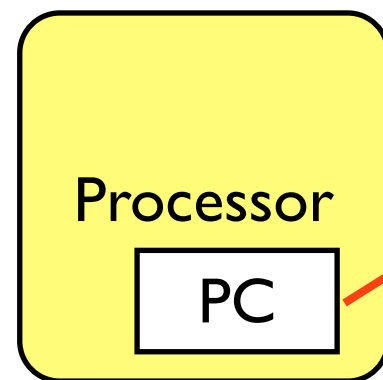
- Number of CPU cycles = Instructions executed * Average Clock Cycles per Instruction (CPI)
- Usually measured in seconds
- Shorter execution time means better performance



			instruction memory
120007a30:	0f00bb27	ldah	gp,15(t12)
120007a34:	509cbd23	lda	gp,-25520(gp)
120007a38:	00005d24	ldah	t1,0(gp)
120007a3c:	0000bd24	ldah	t4,0(gp)
120007a40:	2ca422a0	ldl	t0,-23508(t1)
120007a44:	130020e4	beq	t0,120007a94
120007a48:	00003d24	ldah	t0,0(gp)
120007a4c:	2ca4e2b3	stl	zero,-23508(t1)
120007a50:	0004ff47	clr	v0
120007a54:	28a4e5b3	stl	zero,-23512(t4)
120007a58:	20a421a4	ldq	t0,-23520(t0)
120007a5c:	0e0020e4	beq	t0,120007a98
120007a60:	0204e147	mov	t0,t1
120007a64:	0304ff47	clr	t2
120007a68:	0500e0c3	br	120007a80

Reasoning About Execution Time via Cycles

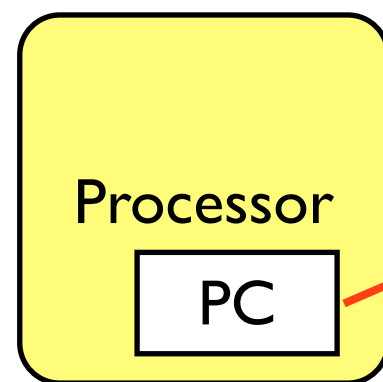
- Number of CPU cycles = Instructions executed * Average Clock Cycles per Instruction (CPI)
- Usually measured in seconds
- Shorter execution time means better performance



instruction memory			
120007a30:	0f00bb27	ldah	gp,15(t12)
120007a34:	509cbd23	lda	gp,-25520(gp)
120007a38:	00005d24	ldah	t1,0(gp)
120007a3c:	0000bd24	ldah	t4,0(gp)
120007a40:	2ca422a0	ldl	t0,-23508(t1)
120007a44:	130020e4	beq	t0,120007a94
120007a48:	00003d24	ldah	t0,0(gp)
120007a4c:	2ca4e2b3	stl	zero,-23508(t1)
120007a50:	0004ff47	clr	v0
120007a54:	28a4e5b3	stl	zero,-23512(t4)
120007a58:	20a421a4	ldq	t0,-23520(t0)
120007a5c:	0e0020e4	beq	t0,120007a98
120007a60:	0204e147	mov	t0,t1
120007a64:	0304ff47	clr	t2
120007a68:	0500e0c3	br	120007a80

Reasoning About Execution Time via Cycles

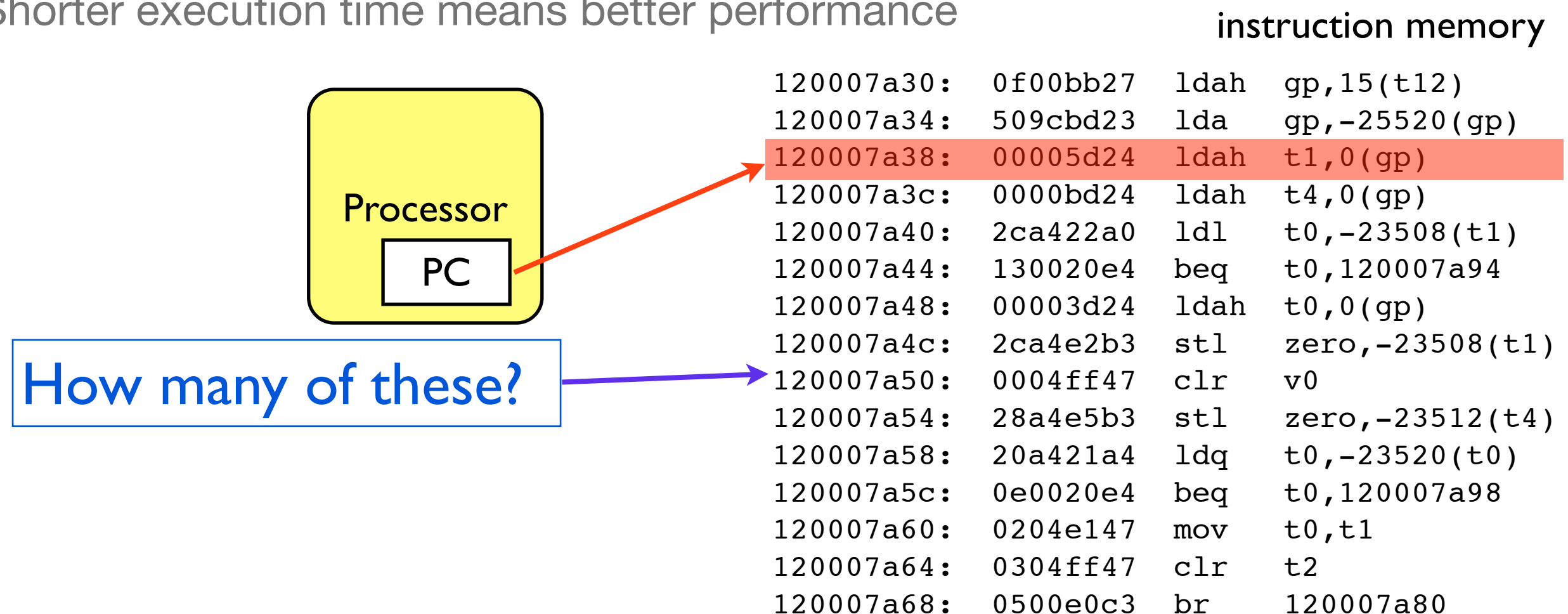
- Number of CPU cycles = Instructions executed * Average Clock Cycles per Instruction (CPI)
- Usually measured in seconds
- Shorter execution time means better performance



instruction memory			
120007a30:	0f00bb27	ldah	gp,15(t12)
120007a34:	509cbd23	lda	gp,-25520(gp)
120007a38:	00005d24	ldah	t1,0(gp)
120007a3c:	0000bd24	ldah	t4,0(gp)
120007a40:	2ca422a0	ldl	t0,-23508(t1)
120007a44:	130020e4	beq	t0,120007a94
120007a48:	00003d24	ldah	t0,0(gp)
120007a4c:	2ca4e2b3	stl	zero,-23508(t1)
120007a50:	0004ff47	clr	v0
120007a54:	28a4e5b3	stl	zero,-23512(t4)
120007a58:	20a421a4	ldq	t0,-23520(t0)
120007a5c:	0e0020e4	beq	t0,120007a98
120007a60:	0204e147	mov	t0,t1
120007a64:	0304ff47	clr	t2
120007a68:	0500e0c3	br	120007a80

Reasoning About Execution Time via Cycles

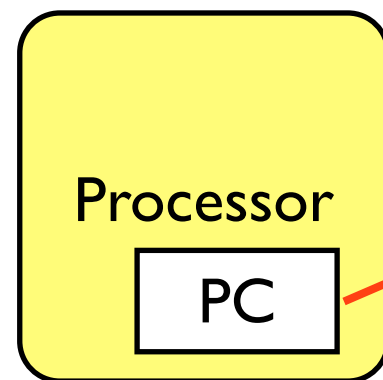
- Number of CPU cycles = Instructions executed * Average Clock Cycles per Instruction (CPI)
- Usually measured in seconds
- Shorter execution time means better performance



Reasoning About Execution Time via Cycles

- Number of CPU cycles = Instructions executed * Average Clock Cycles per Instruction (CPI)
- Usually measured in seconds
- Shorter execution time means better performance

instruction memory



How many of these?

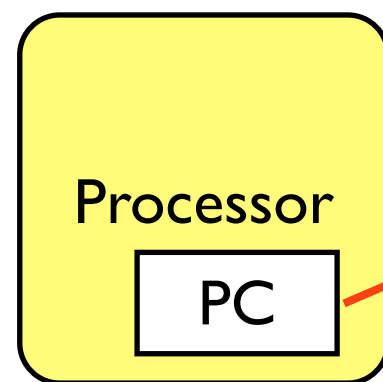
How long is it take to execution each of these?

120007a30:	0f00bb27	ldah	gp,15(t12)
120007a34:	509cbd23	lda	gp,-25520(gp)
120007a38:	00005d24	ldah	t1,0(gp)
120007a3c:	0000bd24	ldah	t4,0(gp)
120007a40:	2ca422a0	ldl	t0,-23508(t1)
120007a44:	130020e4	beq	t0,120007a94
120007a48:	00003d24	ldah	t0,0(gp)
120007a4c:	2ca4e2b3	stl	zero,-23508(t1)
120007a50:	0004ff47	clr	v0
120007a54:	28a4e5b3	stl	zero,-23512(t4)
120007a58:	20a421a4	ldq	t0,-23520(t0)
120007a5c:	0e0020e4	beq	t0,120007a98
120007a60:	0204e147	mov	t0,t1
120007a64:	0304ff47	clr	t2
120007a68:	0500e0c3	br	120007a80

Reasoning About Execution Time via Cycles

- Number of CPU cycles = Instructions executed * Average Clock Cycles per Instruction (CPI)
- Usually measured in seconds
- Shorter execution time means better performance

instruction memory



How many of these?

Instruction Count!

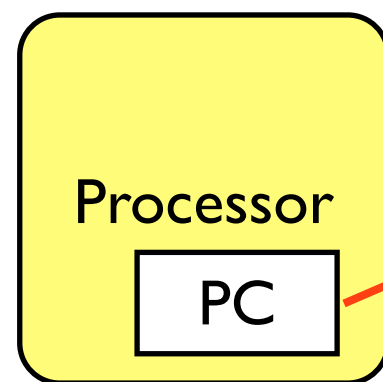
How long is it take to
execution each of these?

120007a30:	0f00bb27	ldah	gp,15(t12)
120007a34:	509cbd23	lda	gp,-25520(gp)
120007a38:	00005d24	ldah	t1,0(gp)
120007a3c:	0000bd24	ldah	t4,0(gp)
120007a40:	2ca422a0	ldl	t0,-23508(t1)
120007a44:	130020e4	beq	t0,120007a94
120007a48:	00003d24	ldah	t0,0(gp)
120007a4c:	2ca4e2b3	stl	zero,-23508(t1)
120007a50:	0004ff47	clr	v0
120007a54:	28a4e5b3	stl	zero,-23512(t4)
120007a58:	20a421a4	ldq	t0,-23520(t0)
120007a5c:	0e0020e4	beq	t0,120007a98
120007a60:	0204e147	mov	t0,t1
120007a64:	0304ff47	clr	t2
120007a68:	0500e0c3	br	120007a80

Reasoning About Execution Time via Cycles

- Number of CPU cycles = Instructions executed * Average Clock Cycles per Instruction (CPI)
- Usually measured in seconds
- Shorter execution time means better performance

instruction memory



120007a30:	0f00bb27	ldah	gp,15(t12)
120007a34:	509cbd23	lda	gp,-25520(gp)
120007a38:	00005d24	ldah	t1,0(gp)
120007a3c:	0000bd24	ldah	t4,0(gp)
120007a40:	2ca422a0	ldl	t0,-23508(t1)
120007a44:	130020e4	beq	t0,120007a94
120007a48:	00003d24	ldah	t0,0(gp)
120007a4c:	2ca4e2b3	stl	zero,-23508(t1)
120007a50:	0004ff47	clr	v0
120007a54:	28a4e5b3	stl	zero,-23512(t4)
120007a58:	20a421a4	ldq	t0,-23520(t0)
120007a5c:	0e0020e4	beq	t0,120007a98
120007a60:	0204e147	mov	t0,t1
120007a64:	0304ff47	clr	t2
120007a68:	0500e0c3	br	120007a80

How many of these?

Instruction Count!

How long is it take to
execution each of these?

Cycles per instruction * cycle time

Reasoning About Execution Time via Cycles

- Number of CPU cycles = Instructions executed * Average Clock Cycles per Instruction (CPI)

Reasoning About Execution Time via Cycles

- Number of CPU cycles = Instructions executed * Average Clock Cycles per Instruction (CPI)
 - Computer A runs program C in 3.6 billion cycles.
Program C consists of 2 billion dynamic instructions.
What is the CPI?

Reasoning About Execution Time via Cycles

- Number of CPU cycles = Instructions executed * Average Clock Cycles per Instruction (CPI)
 - Computer A runs program C in 3.6 billion cycles.
Program C consists of 2 billion dynamic instructions.
What is the CPI?
 - **1.8 cyc**

Reasoning About Execution Time via Cycles

- Number of CPU cycles = Instructions executed * Average Clock Cycles per Instruction (CPI)
 - Computer A runs program C in 3.6 billion cycles. Program C consists of 2 billion dynamic instructions. What is the CPI?
 - **1.8 cyc**
 - A computer is running a program with CPI = 2.0, and executes 24 million instructions, how long will it run?

Reasoning About Execution Time via Cycles

- Number of CPU cycles = Instructions executed * Average Clock Cycles per Instruction (CPI)
 - Computer A runs program C in 3.6 billion cycles. Program C consists of 2 billion dynamic instructions. What is the CPI?
 - **1.8 cyc**
 - A computer is running a program with CPI = 2.0, and executes 24 million instructions, how long will it run?
 - **48 mil. cyc**

Performance Equation

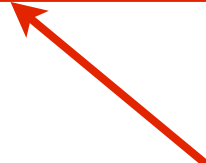
Execution Time =



How many instruction
executed?



How long is it to execute
each instruction



Performance Equation

Execution Time = $\frac{\text{Instructions}}{\text{Program}}$

How many instruction executed?

How long is it to execute each instruction

The diagram illustrates the Performance Equation: Execution Time = Instructions / Program. The fraction is enclosed in a red box. A red arrow points from the text 'How many instruction executed?' to the 'Instructions' part of the fraction. Another red arrow points from the text 'How long is it to execute each instruction' to the 'Program' part of the fraction. To the right of the fraction is a large, empty red rectangular box.

Performance Equation

$$\text{Execution Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}}$$

How many instruction executed?

How long is it to execute each instruction

Performance Equation

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

How many instruction executed?

How long is it to execute each instruction

Performance Equation

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

How many instruction executed?

How long is it to execute each instruction

- $ET = IC * CPI * CT$

Performance Equation

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

How many instruction executed?

How long is it to execute each instruction

- $ET = IC * CPI * CT$
- IC (Instruction Count)

Performance Equation

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

How many instruction executed?

How long is it to execute each instruction

- $ET = IC * CPI * CT$
- IC (Instruction Count)
- CPI (Cycles Per Instruction)

Performance Equation

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

How many instruction executed?

How long is it to execute each instruction

- $ET = IC * CPI * CT$
- IC (Instruction Count)
- CPI (Cycles Per Instruction)
- CT (Seconds Per Cycle)

Performance Equation

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

How many instruction executed?

How long is it to execute each instruction

- $ET = IC * CPI * CT$
- IC (Instruction Count)
- CPI (Cycles Per Instruction)
- CT (Seconds Per Cycle)
 - 1 Hz = 1 second per cycle; 1 GHz = 1 ns per cycle

Computer Performance...

Performance Equation

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

How many instruction executed?

How long is it to execute each instruction

- $ET = IC * CPI * CT$
- IC (Instruction Count)
- CPI (Cycles Per Instruction)
- CT (Seconds Per Cycle)
 - 1 Hz = 1 second per cycle; 1 GHz = 1 ns per cycle

Speedup

$$\text{Speedup (X/Y)} = \frac{\text{Performance}_X}{\text{Performance}_Y} = \frac{\text{Execution Time}_Y}{\text{Execution Time}_X} = n$$

Speedup

$$\text{Speedup (X/Y)} = \frac{\text{Performance}_X}{\text{Performance}_Y} = \frac{\text{Execution Time}_Y}{\text{Execution Time}_X} = n$$

$$\text{Speedup} = \frac{\text{Execution time}_{\text{baseline}}}{\text{Execution time}_{\text{improved system}}}$$

Speedup

$$\text{Speedup (X/Y)} = \frac{\text{Performance}_X}{\text{Performance}_Y} = \frac{\text{Execution Time}_Y}{\text{Execution Time}_X} = n$$

$$\text{Speedup} = \frac{\text{Execution time}_{\text{baseline}}}{\text{Execution time}_{\text{improved system}}}$$

- Speedup when moving to the faster system ($>1x$)
 - Execution time of **slower** system is in the **numerator**
- Speedup when moving to the slower system ($<1x$)
 - Execution time of **faster** system is in the **numerator**

Performance Example

- Assume that we have an application composed with a total of **500,000** instructions, the average CPI of the application is **2** cycle. If the processor runs at **1GHz**, how long is it take to execute the application?

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

Performance Example

- Assume that we have an application composed with a total of **500,000** instructions, the average CPI of the application is **2** cycle. If the processor runs at **1GHz**, how long is it take to execute the application?

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

500,000 *

Performance Example

- Assume that we have an application composed with a total of **500,000** instructions, the average CPI of the application is **2** cycle. If the processor runs at **1GHz**, how long is it take to execute the application?

$$\text{Execution Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$
$$\frac{500,000 *}{2 *}$$

Performance Example

- Assume that we have an application composed with a total of **500,000** instructions, the average CPI of the application is **2** cycle. If the processor runs at **1GHz**, how long is it take to execute the application?

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

$$\begin{aligned} & 500,000 * \\ & 2 * \\ & 1 \text{ ns} = \end{aligned}$$

Performance Example

- Assume that we have an application composed with a total of **500,000** instructions, the average CPI of the application is **2** cycle. If the processor runs at **1GHz**, how long is it take to execute the application?

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

$$\begin{aligned} & 500,000 * \\ & 2 * \\ & 1 \text{ ns} = \\ & 1,000,000 \text{ ns} \end{aligned}$$

Slightly More Interesting Example...

- Assume that we have an application composed with a total of **500,000** instructions, in which **20%** of them are the load/store instructions with an average CPI of **6** cycles, and the rest of the instructions are integer instructions with average CPI of **1** cycle.

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

Slightly More Interesting Example...

- Assume that we have an application composed with a total of **500,000** instructions, in which **20%** of them are the load/store instructions with an average CPI of **6** cycles, and the rest of the instructions are integer instructions with average CPI of **1** cycle.

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

500,000 *

Slightly More Interesting Example...

- Assume that we have an application composed with a total of **500,000** instructions, in which **20%** of them are the load/store instructions with an average CPI of **6** cycles, and the rest of the instructions are integer instructions with average CPI of **1** cycle.

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

$$500,000 * (.2 * 6 + .8 * 1) *$$

Slightly More Interesting Example...

- Assume that we have an application composed with a total of **500,000** instructions, in which **20%** of them are the load/store instructions with an average CPI of **6** cycles, and the rest of the instructions are integer instructions with average CPI of **1** cycle.

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

$$\begin{aligned} & 500,000 * \\ & (.2 * 6 + .8 * 1) * \\ & 1 \text{ ns} = \end{aligned}$$

Slightly More Interesting Example...

- Assume that we have an application composed with a total of **500,000** instructions, in which **20%** of them are the load/store instructions with an average CPI of **6** cycles, and the rest of the instructions are integer instructions with average CPI of **1** cycle.

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

$$\begin{aligned} & 500,000 * \\ & (.2 * 6 + .8 * 1) * \\ & 1 \text{ ns} = \\ & 1,000,000 \text{ ns} \end{aligned}$$

Speedup example

- If we double the clock rate to be **2GHz** without improve the memory latency, the average CPI for load/store instruction will also be doubled to **12** cycles. What's the performance improvement after this change?

Speedup example

- If we double the clock rate to be **2GHz** without improve the memory latency, the average CPI for load/store instruction will also be doubled to **12** cycles. What's the performance improvement after this change?

500,000 *

Speedup example

- If we double the clock rate to be **2GHz** without improve the memory latency, the average CPI for load/store instruction will also be doubled to **12** cycles. What's the performance improvement after this change?

$$500,000 * (.2 * 12 + .8 * 1) *$$

Speedup example

- If we double the clock rate to be **2GHz** without improve the memory latency, the average CPI for load/store instruction will also be doubled to **12** cycles. What's the performance improvement after this change?

$$\frac{500,000 * (.2 * 12 + .8 * 1) * 0.5 \text{ ns}}{0.5 \text{ ns}} =$$

Speedup example

- If we double the clock rate to be **2GHz** without improve the memory latency, the average CPI for load/store instruction will also be doubled to **12** cycles. What's the performance improvement after this change?

$$\begin{aligned} & 500,000 * \\ & (.2 * 12 + .8 * 1) * \\ & 0.5 \text{ ns} = \\ & 800,000 \text{ ns} \end{aligned}$$

Speedup example

- If we double the clock rate to be **2GHz** without improve the memory latency, the average CPI for load/store instruction will also be doubled to **12** cycles. What's the performance improvement after this change?

$$\begin{aligned} & 500,000 * \\ & (.2 * 12 + .8 * 1) * \\ & 0.5 \text{ ns} = \\ & 800,000 \text{ ns} \end{aligned}$$

$$\text{Speedup} = \frac{\text{Execution time}_{\text{baseline}}}{\text{Execution time}_{\text{improved system}}}$$

Speedup example

- If we double the clock rate to be **2GHz** without improve the memory latency, the average CPI for load/store instruction will also be doubled to **12** cycles. What's the performance improvement after this change?

$$\begin{aligned} & 500,000 * \\ & (.2 * 12 + .8 * 1) * \\ & 0.5 \text{ ns} = \\ & 800,000 \text{ ns} \end{aligned}$$

new machine = 800,000 ns

$$\text{Speedup} = \frac{\text{Execution time}_{\text{baseline}}}{\text{Execution time}_{\text{improved system}}}$$

Speedup example

- If we double the clock rate to be **2GHz** without improve the memory latency, the average CPI for load/store instruction will also be doubled to **12** cycles. What's the performance improvement after this change?

$$\begin{aligned} & 500,000 * \\ & (.2 * 12 + .8 * 1) * \\ & 0.5 \text{ ns} = \\ & 800,000 \text{ ns} \end{aligned}$$

$$\text{Speedup} = \frac{\text{Execution time}_{\text{baseline}}}{\text{Execution time}_{\text{improved system}}}$$

new machine = 800,000 ns

old machine = 1,000,000 ns

Speedup example

- If we double the clock rate to be **2GHz** without improve the memory latency, the average CPI for load/store instruction will also be doubled to **12** cycles. What's the performance improvement after this change?

$$\begin{aligned} & 500,000 * \\ & (.2 * 12 + .8 * 1) * \\ & 0.5 \text{ ns} = \\ & 800,000 \text{ ns} \end{aligned}$$

$$\text{Speedup} = \frac{\text{Execution time}_{\text{baseline}}}{\text{Execution time}_{\text{improved system}}}$$

old machine = 1,000,000 ns
new machine = 800,000 ns

Speedup example

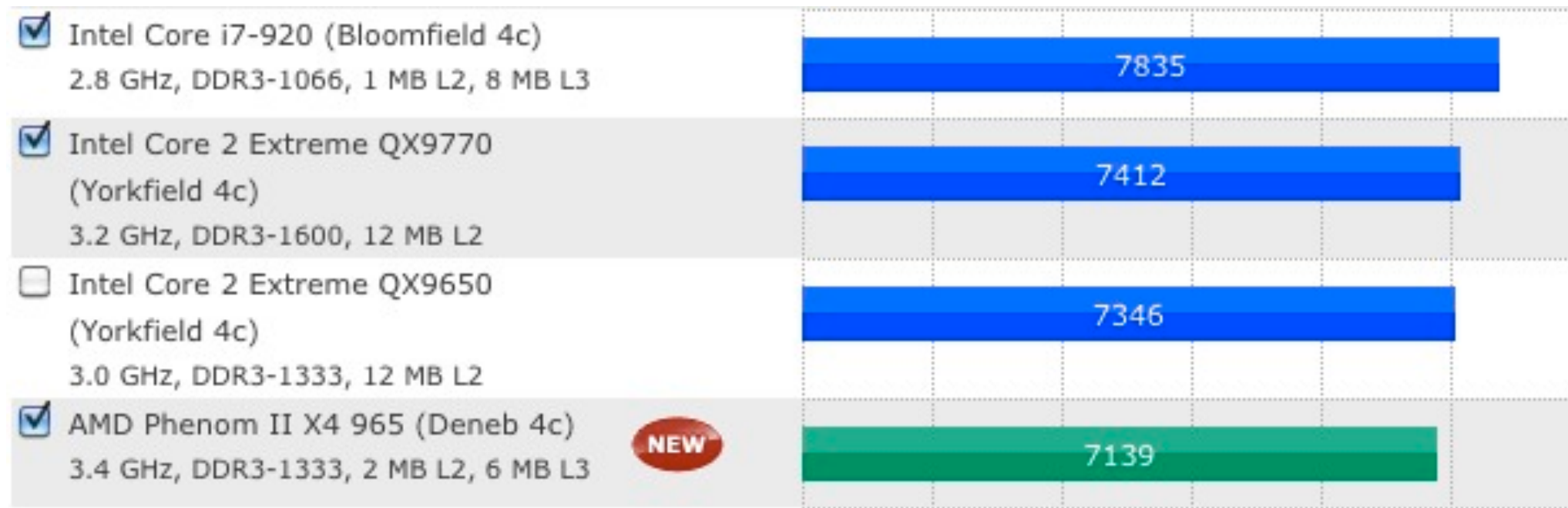
- If we double the clock rate to be **2GHz** without improve the memory latency, the average CPI for load/store instruction will also be doubled to **12** cycles. What's the performance improvement after this change?

$$\begin{aligned} & 500,000 * \\ & (.2 * 12 + .8 * 1) * \\ & 0.5 \text{ ns} = \\ & 800,000 \text{ ns} \end{aligned}$$

$$\text{Speedup} = \frac{\text{Execution time}_{\text{baseline}}}{\text{Execution time}_{\text{improved system}}} = 1.25x$$

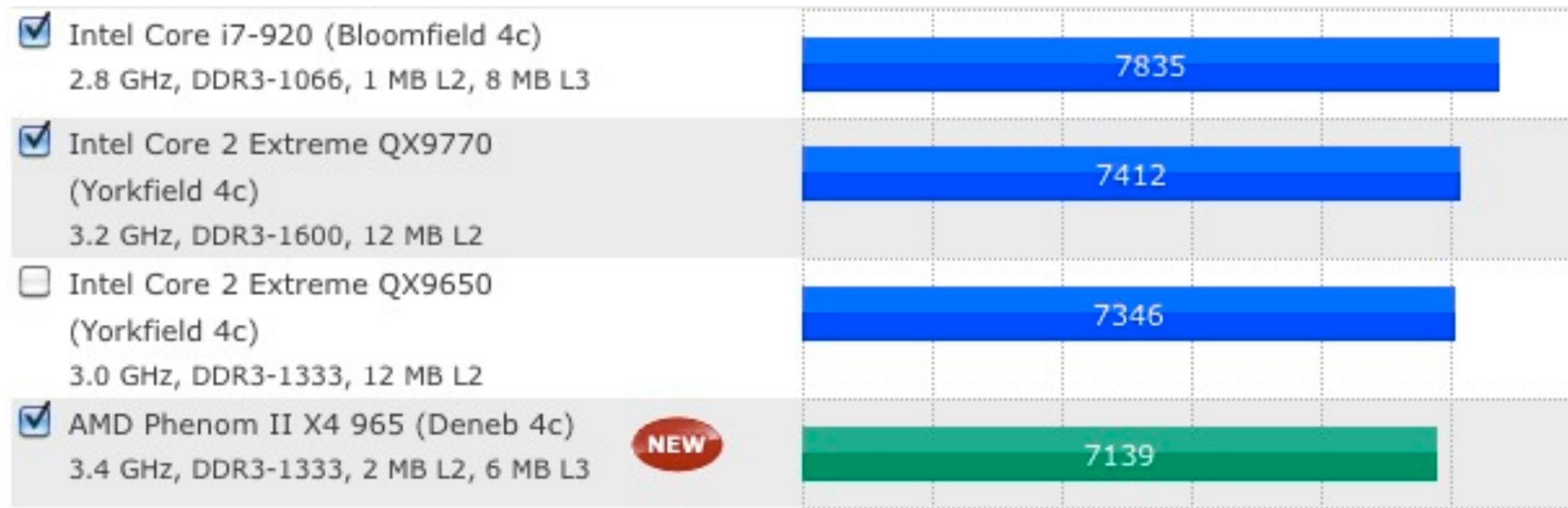
old machine = 1,000,000 ns
new machine = 800,000 ns

Reasoning About Performance



- Why does an Intel Core i7 @ 2.8 GHz usually perform better than an Intel Core 2 Extreme @ 3.2 GHz or AMD Phenom II X4 @ **3.4GHz**?

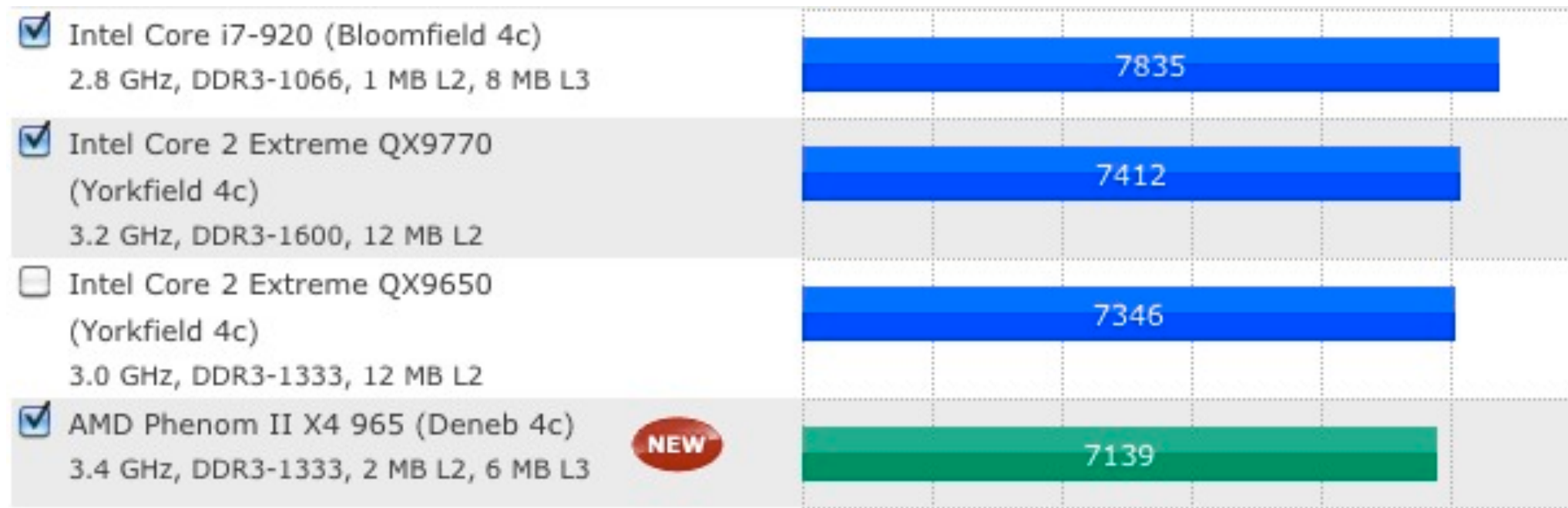
Reasoning About Performance



- Why does an Intel Core i7 @ 2.8 GHz usually perform better than an Intel Core 2 Extreme @ 3.2 GHz or AMD Phenom II X4 @ **3.4GHz**?

$$ET = IC \times CPI \times \text{Cycle Time}$$

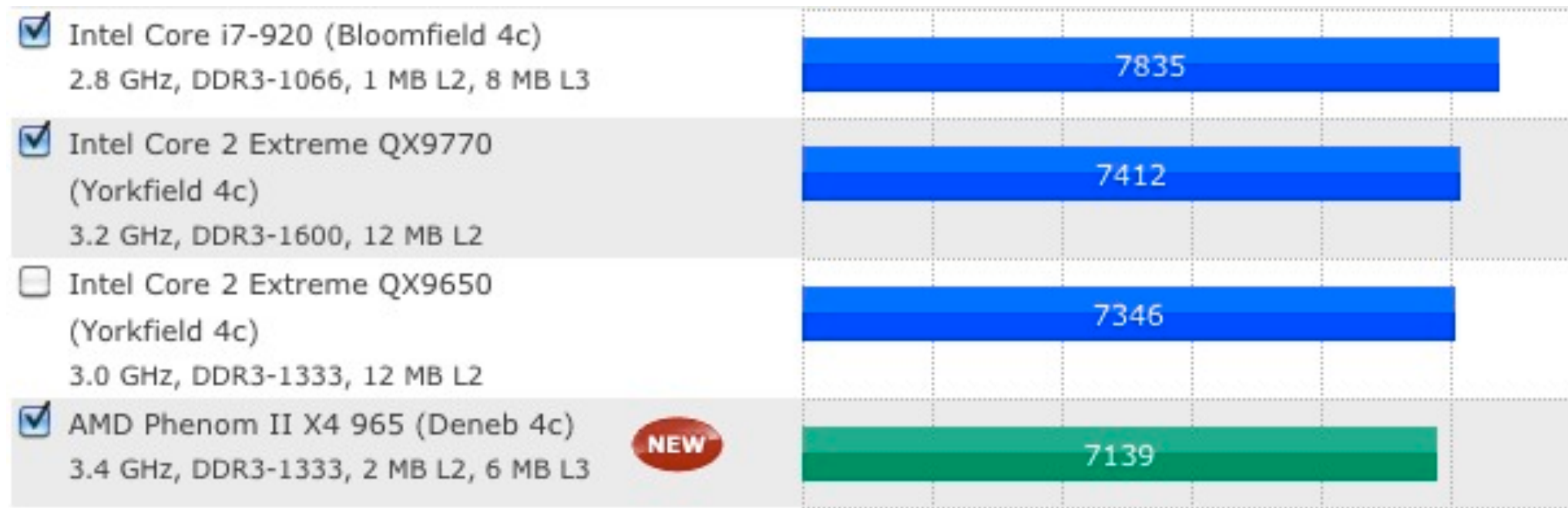
Reasoning About Performance



- Why does an Intel Core i7 @ 2.8 GHz usually perform better than an Intel Core 2 Extreme @ 3.2 GHz or AMD Phenom II X4 @ **3.4GHz**?

$$ET = IC \times CPI \times \text{Cycle Time}$$

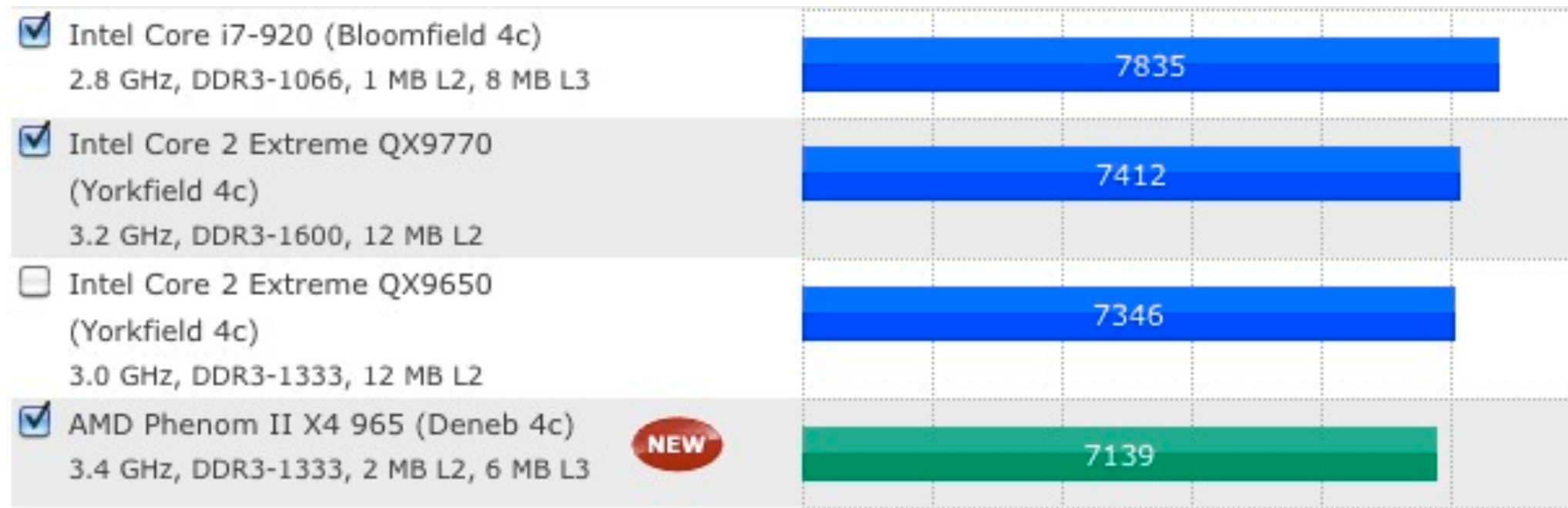
Reasoning About Performance



- Why does an Intel Core i7 @ 2.8 GHz usually perform better than an Intel Core 2 Extreme @ 3.2 GHz or AMD Phenom II X4 @ **3.4GHz**?

$$ET = \cancel{IC} \times \text{CPI} \times \text{Cycle Time}$$

Reasoning About Performance



- Why does an Intel Core i7 @ 2.8 GHz usually perform better than an Intel Core 2 Extreme @ 3.2 GHz or AMD Phenom II X4 @ **3.4GHz**?

$$ET = \cancel{IC} \times \text{CPI} \times \cancel{\text{CycleTime}}$$

What Affects Performance

	Number of instructions	CPI	Clock Cycle Time
Same machine different programs			
same programs, different machines, same ISA			
Same programs, different machines different ISA			

What Affects Performance

	Number of instructions	CPI	Clock Cycle Time
Same machine different programs			
same programs, different machines, same ISA			
Same programs, different machines different ISA			

What Affects Performance

	Number of instructions	CPI	Clock Cycle Time
Same machine different programs	✓	✓	
same programs, different machines, same ISA			
Same programs, different machines different ISA			

What Affects Performance

	Number of instructions	CPI	Clock Cycle Time
Same machine different programs	✓	✓	✗
same programs, different machines, same ISA			
Same programs, different machines different ISA			

What Affects Performance

	Number of instructions	CPI	Clock Cycle Time
Same machine different programs	✓	✓	✗
same programs, different machines, same ISA	✗		
Same programs, different machines different ISA			

What Affects Performance

	Number of instructions	CPI	Clock Cycle Time
Same machine different programs	✓	✓	✗
same programs, different machines, same ISA	✗	✓	
Same programs, different machines different ISA			

What Affects Performance

	Number of instructions	CPI	Clock Cycle Time
Same machine different programs	✓	✓	✗
same programs, different machines, same ISA	✗	✓	✓
Same programs, different machines different ISA			

What Affects Performance

	Number of instructions	CPI	Clock Cycle Time
Same machine different programs	✓	✓	✗
same programs, different machines, same ISA	✗	✓	✓
Same programs, different machines different ISA		✓	

What Affects Performance

	Number of instructions	CPI	Clock Cycle Time
Same machine different programs	✓	✓	✗
same programs, different machines, same ISA	✗	✓	✓
Same programs, different machines different ISA		✓	✓

What Affects Performance

	Number of instructions	CPI	Clock Cycle Time
Same machine different programs	✓	✓	✗
same programs, different machines, same ISA	✗	✓	✓
Same programs, different machines different ISA	✓	✓	✓

What Affects Performance?

What Affects Performance?

- Compiler

What Affects Performance?

- Compiler
 - IC, CPI

What Affects Performance?

- Compiler
 - IC, CPI
- Instruction Set Architecture

What Affects Performance?

- Compiler
 - IC, CPI
- Instruction Set Architecture
 - IC, CPI, CT

What Affects Performance?

- Compiler
 - IC, CPI
- Instruction Set Architecture
 - IC, CPI, CT
- Programmer

What Affects Performance?

- Compiler
 - IC, CPI
- Instruction Set Architecture
 - IC, CPI, CT
- Programmer
 - IC, CPI

What Affects Performance?

- Compiler
 - IC, CPI
- Instruction Set Architecture
 - IC, CPI, CT
- Programmer
 - IC, CPI
- Application

What Affects Performance?

- Compiler
 - IC, CPI
- Instruction Set Architecture
 - IC, CPI, CT
- Programmer
 - IC, CPI
- Application
 - IC

What Affects Performance?

- Compiler
 - IC, CPI
- Instruction Set Architecture
 - IC, CPI, CT
- Programmer
 - IC, CPI
- Application
 - IC
- Processor architecture

What Affects Performance?

- Compiler
 - IC, CPI
- Instruction Set Architecture
 - IC, CPI, CT
- Programmer
 - IC, CPI
- Application
 - IC
- Processor architecture
 - CPI, CT

What Affects Performance?

- Compiler
 - IC, CPI
- Instruction Set Architecture
 - IC, CPI, CT
- Programmer
 - IC, CPI
- Application
 - IC
- Processor architecture
 - CPI, CT
- Process technology

What Affects Performance?

- Compiler
 - IC, CPI
- Instruction Set Architecture
 - IC, CPI, CT
- Programmer
 - IC, CPI
- Application
 - IC
- Processor architecture
 - CPI, CT
- Process technology
 - CT

What Affects Performance?

- Compiler
 - IC, CPI
- Instruction Set Architecture
 - IC, CPI, CT
- Programmer
 - IC, CPI
- Application
 - IC
- Processor architecture
 - CPI, CT
- Process technology
 - CT
- Programming languages

What Affects Performance?

- Compiler
 - IC, CPI
- Instruction Set Architecture
 - IC, CPI, CT
- Programmer
 - IC, CPI
- Application
 - IC
- Processor architecture
 - CPI, CT
- Process technology
 - CT
- Programming languages
 - IC, CPI

The Practice of Measuring Performance

The Practice of Measuring Performance

- Peak throughput measures (simple programs)? (GFLOPS)

The Practice of Measuring Performance

- Peak throughput measures (simple programs)? (GFLOPS)
- Synthetic benchmarks (whetstone, dhrystone,...)?

The Practice of Measuring Performance

- Peak throughput measures (simple programs)? (GFLOPS)
- Synthetic benchmarks (whetstone, dhrystone,...)?
- **Real applications**

The Practice of Measuring Performance

- Peak throughput measures (simple programs)? (GFLOPS)
- Synthetic benchmarks (whetstone, dhrystone,...)?
- **Real applications**
- SPEC (best of both worlds, but with problems of their own)
 - *System Performance Evaluation Cooperative*
 - Provides a common set of real applications along with strict guidelines for how to run them.
 - provides a relatively unbiased means to compare machines

Danger in Benchmark Specific Measures



*Measures **compiler** as much as **architecture**!*



Amdahl's Law

Amdahl's Law

$$\text{Execution time after improvement} = \frac{\text{Execution Time Affected}}{\text{Amount of Improvement}} + \text{Execution Time Unaffected}$$

- The impact of a performance improvement is limited by the percent of execution time affected by the improvement

Amdahl's Law

$$\text{Speedup} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$



- Not only application to CS, but can be used anywhere!

Amdahl's Law: The Super MPEG Decoder

- Assume that we have a game spending 25% of its time doing MPEG decoding. If we add a hardware MPEG decoder that can speed up the MPEG decoding by 10x. How much does the hardware MPEG decoder help?

Amdahl's Law: The Super MPEG Decoder

- Assume that we have a game spending 25% of its time doing MPEG decoding. If we add a hardware MPEG decoder that can speed up the MPEG decoding by 10x. How much does the hardware MPEG decoder help?

$$\text{Speedup} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

Amdahl's Law: The Super MPEG Decoder

- Assume that we have a game spending 25% of it's time doing MPEG decoding. If we add a hardware MPEG decoder that can speed up the MPEG decoding by 10x. How much does the hardware MPEG decoder help?

$$\text{Speedup} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

$$\text{Speedup} =$$

Amdahl's Law: The Super MPEG Decoder

- Assume that we have a game spending 25% of its time doing MPEG decoding. If we add a hardware MPEG decoder that can speed up the MPEG decoding by 10x. How much does the hardware MPEG decoder help?

$$\text{Speedup} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

$$\text{Speedup} = \frac{1}{1 - 0.25 + \frac{0.25}{10}}$$

Amdahl's Law: The Super MPEG Decoder

- Assume that we have a game spending 25% of its time doing MPEG decoding. If we add a hardware MPEG decoder that can speed up the MPEG decoding by 10x. How much does the hardware MPEG decoder help?

$$\text{Speedup} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

$$\text{Speedup} = \frac{1}{\quad}$$

Amdahl's Law: The Super MPEG Decoder

- Assume that we have a game spending 25% of its time doing MPEG decoding. If we add a hardware MPEG decoder that can speed up the MPEG decoding by 10x. How much does the hardware MPEG decoder help?

$$\text{Speedup} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

$$\text{Speedup} = \frac{1}{(1 - 0.25)}$$

Amdahl's Law: The Super MPEG Decoder

- Assume that we have a game spending 25% of its time doing MPEG decoding. If we add a hardware MPEG decoder that can speed up the MPEG decoding by 10x. How much does the hardware MPEG decoder help?

$$\text{Speedup} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

$$\text{Speedup} = \frac{1}{(1 - 0.25) + \frac{0.25}{10}}$$

Amdahl's Law: The Super MPEG Decoder

- Assume that we have a game spending 25% of its time doing MPEG decoding. If we add a hardware MPEG decoder that can speed up the MPEG decoding by 10x. How much does the hardware MPEG decoder help?

$$\text{Speedup} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

$$\text{Speedup} = \frac{1}{(1 - 0.25) + \frac{0.25}{10}}$$

Amdahl's Law: The Super MPEG Decoder

- Assume that we have a game spending 25% of its time doing MPEG decoding. If we add a hardware MPEG decoder that can speed up the MPEG decoding by 10x. How much does the hardware MPEG decoder help?

$$\text{Speedup} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

$$\text{Speedup} = \frac{1}{(1 - 0.25) + \frac{0.25}{10}}$$

Amdahl's Law: The Super MPEG Decoder

- Assume that we have a game spending 25% of its time doing MPEG decoding. If we add a hardware MPEG decoder that can speed up the MPEG decoding by 10x. How much does the hardware MPEG decoder help?

$$\text{Speedup} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

$$\text{Speedup} = \frac{1}{(1 - 0.25) + \frac{0.25}{10}}$$

Amdahl's Law: The Super MPEG Decoder

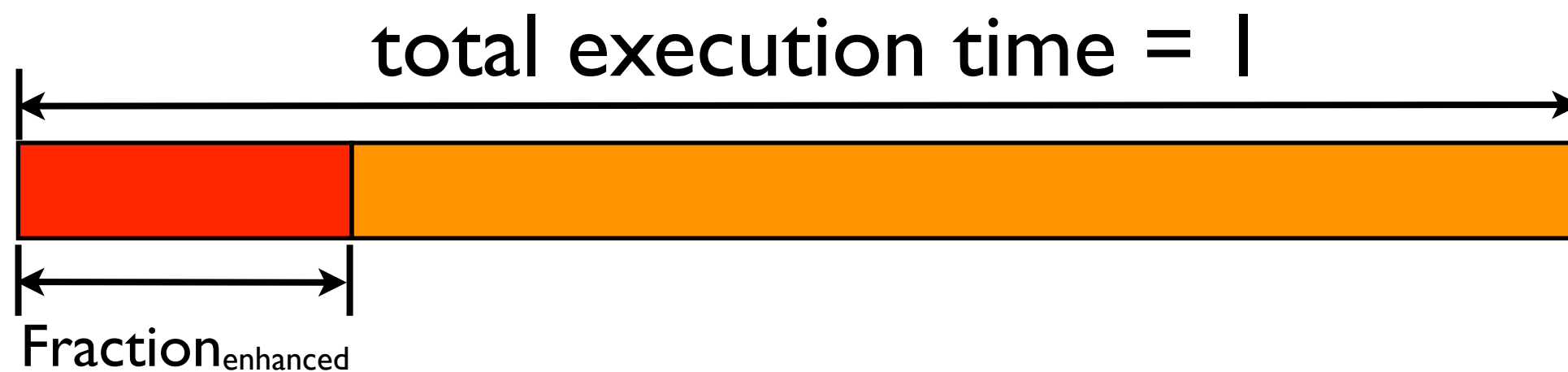
- Assume that we have a game spending 25% of its time doing MPEG decoding. If we add a hardware MPEG decoder that can speed up the MPEG decoding by 10x. How much does the hardware MPEG decoder help?

$$\text{Speedup} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

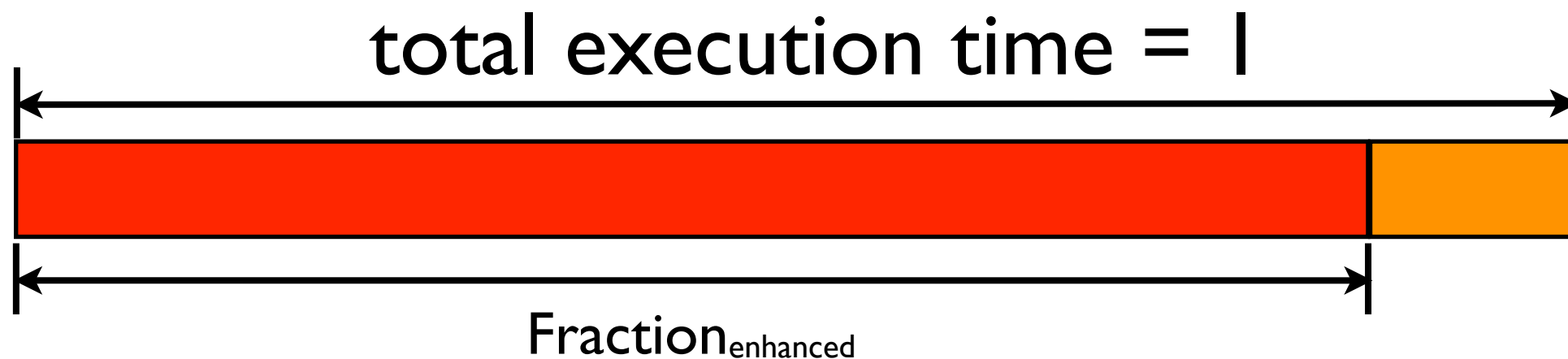
$$\text{Speedup} = \frac{1}{(1 - 0.25) + \frac{0.25}{10}} = 1.29$$

Amdahl's Law

- Make the common case fast!!!



VS



Key Points

Key Points

- Be careful how you specify performance

Key Points

- Be careful how you specify performance
- Execution time = instructions * CPI * cycle time

Key Points

- Be careful how you specify performance
- Execution time = instructions * CPI * cycle time
- Use real applications

Key Points

- Be careful how you specify performance
- Execution time = instructions * CPI * cycle time
- Use real applications
- Use standards, if possible

Key Points

- Be careful how you specify performance
- Execution time = instructions * CPI * cycle time
- Use real applications
- Use standards, if possible
- Make the common case fast