NWEN 242

2. Understand performance

The performance concepts

- Throughput: total work done in a given time
- Response time: time between start and completion of a task

- Applications usually have varied demand on throughput and response time

- Example: do the following changes to a computer system increase throughput or decrease response time?
  - 1. Replace the processor in a computer with a faster one?
  - 2. Adding more processors to a computer system?

Performance vs. execution time

- To maximize performance, we want to minimize execution time

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

- For two computers X and Y, if the performance of X is greater than that of Y, then

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

$$\frac{1}{\text{Execution time}_X} > \frac{1}{\text{Execution time}_Y}$$

$$\text{Execution time}_Y > \text{Execution time}_X$$

- If X is n-times faster than Y, then

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

Quick question

- If computer A runs a program in 10 seconds and computer B runs the same program in 15 seconds, how much faster is A than B?
  - A. 1.0
  - B. 2.5
  - C. 1.5
  - D. 0.5

Quick exercise

- If computer A runs a program in 10 seconds and computer B runs the same program in 15 seconds, how much faster is A than B?

- Answer:

$$\frac{\text{Performance}_A}{\text{Performance}_B} = \frac{\text{Execution time}_B}{\text{Execution time}_A} = n$$

  - Thus the performance ratio is

$$\frac{15}{10} = 1.5$$

  - So A is 1.5 times faster than B, or B is 1.5 times slower than A.

$$\frac{\text{Performance}_A}{1.5} = \text{Performance}_B$$

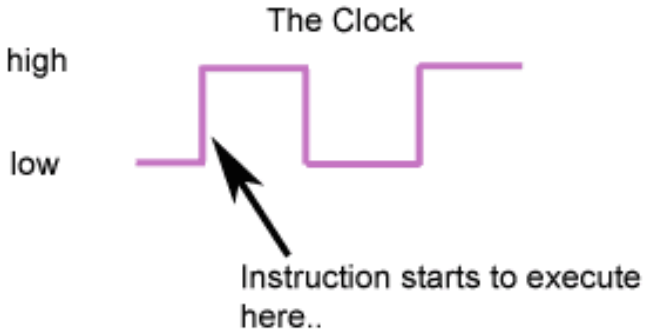
Measuring performance

- Time is the measure of computer performance
  - The computer that performs the same amount of work in the least time is the fastest.

- Different time concepts
  - Clock time / response time / elapsed time
    - The total time to complete a task, including disk accesses, memory accesses, I/O operations, operating system overhead, etc.
  - CPU time: the time the CPU spends on a task
    - Does not include time spent waiting for I/O or running other programs.
    - User CPU time: CPU time spent in the program
    - System CPU time: CPU time spent in the operating system performing tasks on behalf of the program

Other methods to measure performance

- Clock cycles: how fast the hardware can perform basic functions.
  - Almost all computers are constructed using a clock that determines when events take place in hardware.



- Clock period: the time for a complete clock cycle.
- Clock rate: the inverse of clock period.

0.1 s ⇒ 10 Hz

CPU performance

- CPU execution time

$$\text{CPU execution time for a program} = \text{CPU clock cycles for a program} \times \text{Clock cycle time}$$

  - Since clock period (cycle time) and clock rate are inverses

$$\text{CPU execution time for a program} = \frac{\text{CPU clock cycles for a program}}{\text{Clock rate}}$$

- How can we improve performance based on the above formula?

  - A. Reducing the number of clock cycles required for a program
  - B. Increasing the clock cycle time.
  - C. Increasing the clock rate
  - D. All of the above.

Example: improve performance

- An important program runs in 10 seconds on computer A
  - Computer A has a 2GHz clock.
- We are trying develop a computer B which will run the program in 6 seconds.
  - Computer B requires 1.2 times as many clock cycles as computer A for the program (because of increased clock rate)
- Question: what clock rate should computer B have in order to run the program in 6 seconds?

- The number of clock cycles required for A

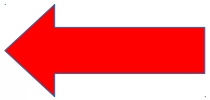
$$\text{CPU time}_A = \frac{\text{CPU clock cycles}_A}{\text{Clock rate}_A}$$

$$10 \text{ seconds} = \frac{\text{CPU clock cycles}_A}{2 \times 10^9 \frac{\text{cycles}}{\text{second}}}$$

$$\text{CPU clock cycles}_A = 10 \text{ seconds} \times 2 \times 10^9 \frac{\text{cycles}}{\text{second}} = 20 \times 10^9 \text{ cycles}$$

- CPU time for B then is

$$\text{CPU time}_B = \frac{1.2 \times \text{CPU clock cycles}_A}{\text{Clock rate}_B}$$



$$6 \text{ seconds} = \frac{1.2 \times 20 \times 10^9 \text{ cycles}}{\text{Clock rate}_B}$$

$$\text{Clock rate}_B = \frac{1.2 \times 20 \times 10^9 \text{ cycles}}{6 \text{ seconds}} = \frac{0.2 \times 20 \times 10^9 \text{ cycles}}{\text{second}} = \frac{4 \times 10^9 \text{ cycles}}{\text{second}} = 4 \text{ GHz}$$

Instruction performance

- Average number of clock cycles per instruction (CPI):

$$\text{CPU clock cycles} = \text{Instructions for a program} \times \text{Average clock cycles per instruction}$$

- Average clock cycles per instruction (CPI) provides a way of comparing two different implementations of the same instruction set architecture.

Instruction set architecture: an abstract interface between the hardware and software that encompasses all the information necessary to write a machine language program.
Instructions, registers, memory access, I/O, etc.

Example question

- Suppose we have two implementations of the same instruction set architecture.
  - unit ps = picosecond = 10^{-12} seconds = 0.000,000,000,001 seconds
  - Computer A has a clock cycle time of 250ps and a CPI of 2.0 for some program.
  - Computer B has a clock cycle time of 500ps and a CPI of 1.2 for the same program.
- Question: which computer is faster for this program and by how much?
  - A. Computer A is faster
  - B. Computer B is faster

- Each computer executes the same number (I) of instructions for the program.

  CPU clock cycles = I x CPI

- Calculate the CPU time for each computer:

$$\text{CPU time}_A = \text{CPU clock cycles}_A \times \text{Clock cycle time}$$

$$= I \times 2.0 \times 250 \text{ ps} = 500 \times I \text{ ps}$$

$$\text{CPU time}_B = I \times 1.2 \times 500 \text{ ps} = 600 \times I \text{ ps}$$

- Computer A is fast by

$$\frac{\text{CPU performance}_A}{\text{CPU performance}_B} = \frac{\text{Execution time}_B}{\text{Execution time}_A} = \frac{600 \times I \text{ ps}}{500 \times I \text{ ps}} = 1.2$$

Example: comparing code segments

- A compiler designer is trying to decide between two code sequences for a particular computer

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

- Details of the two code sequences

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

- Question: which code sequence will be faster?

Answer

- Sequence 1: 2+1+2=5 instructions
- Sequence 2: 4+1+1=6 instructions

- Calculate clock cycles

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

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

- Calculate CPI

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

$$\text{CPI}_1 = \frac{\text{CPU clock cycles}_1}{\text{Instruction count}_1} = \frac{10}{5} = 2.0$$

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

Summary of performance analysis

- General formula

$$\text{CPU Time} = \frac{\text{seconds}}{\text{program}} = \frac{\text{instructions}}{\text{program}} \times \frac{\text{clock cycles}}{\text{instructions}} \times \frac{\text{seconds}}{\text{clock cycle}}$$

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

Understanding program performance

Hardware or software component	Affects what?	How?
Algorithm	Instruction count, possibly CPI	The algorithm determines the number of source program instructions executed and hence the number of processor instructions executed. The algorithm may also affect the CPI, by favoring slower or faster instructions. For example, if the algorithm uses more floating-point operations, it will tend to have a higher CPI.
Programming language	Instruction count, CPI	The programming language certainly affects the instruction count, since statements in the language are translated to processor instructions, which determine instruction count. The language may also affect the CPI because of its features; for example, a language with heavy support for data abstraction (e.g.. Java) will require indirect calls, which will use higher CPI instructions.
Compiler	Instruction count, CPI	The efficiency of the compiler affects both the instruction count and average cycles per instruction, since the compiler determines the translation of the source language instructions into computer instructions. The compiler's role can be very complex and affect the CPI in complex ways.
Instruction set architecture	Instruction count, clock rate, CPI	The instruction set architecture affects all three aspects of CPU performance, since it affects the instructions needed for a function, the cost in cycles of each instruction, and the overall clock rate of the processor.

Check yourself

- A given application written in Java runs 15 seconds on a desktop processor. A new Java compiler is released that requires only 0.6 as many instructions as the old compiler. Unfortunately it increases the CPI by 1.1. How fast can we expect the application to run using this new compiler?

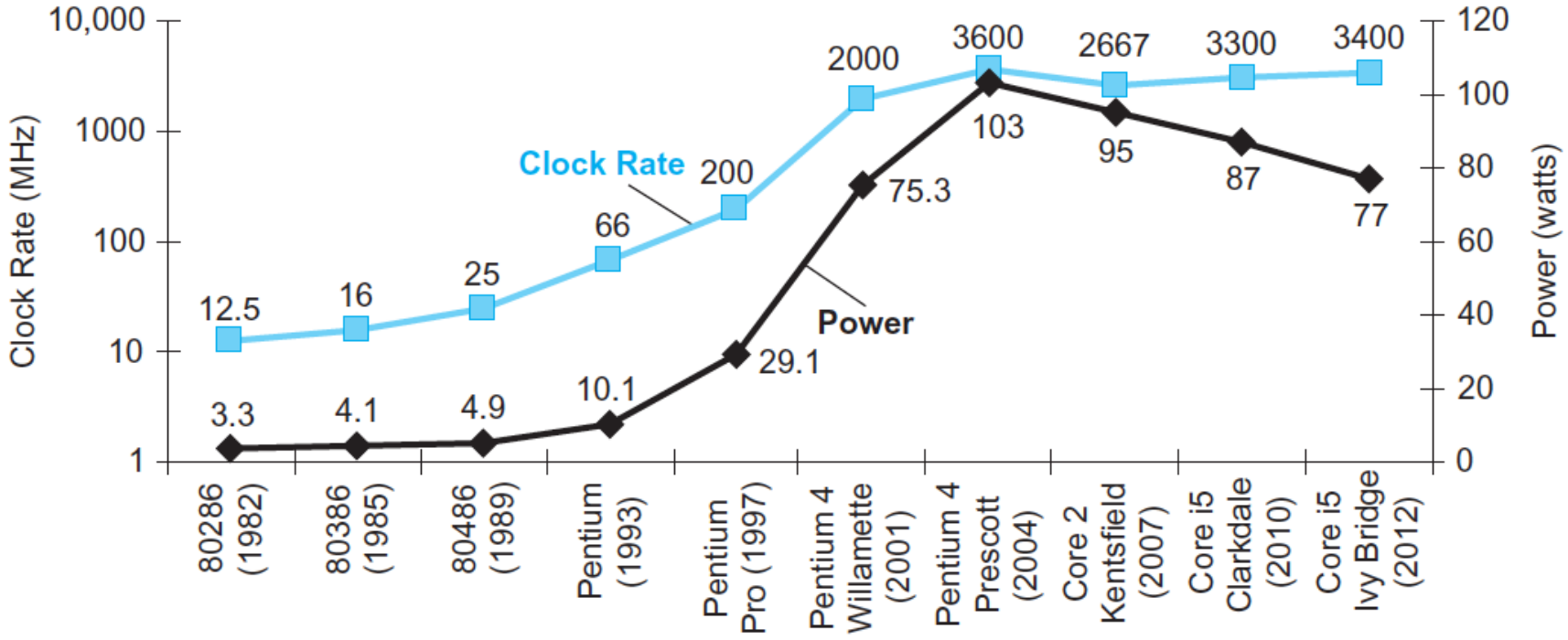
Check yourself

- A given application written in Java runs 15 seconds on a desktop processor. A new Java compiler is released that requires only 0.6 as many instructions as the old compiler. Unfortunately it increases the CPI by 1.1. How fast can we expect the application to run using this new compiler?

$$15 \times 0.6 \times 1.1 = 9.9 \text{ seconds}$$

The power wall

- Clock rate and power for Intel x86 microprocessors over eight generations and 30 years.



Quick question

- The energy consumption of a processor

$$Power \propto 1/2 \times Capacitive\ load \times Voltage^2 \times Frequency\ switched$$

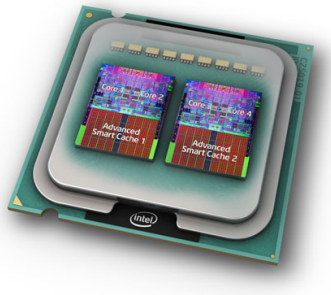
- Which approach is the most effective in reducing power consumption?
  - A. Reduce the size of each transistor
  - B. Reduce operating voltage
  - C. Reduce clock rate
  - D. Design simple processors

Problem of lowering the voltage

- Further lowering the voltage appears to make the transistors too leaky.
- About 40% of the power consumption is due to leakage.
- Solution:
  - Attach large devices to increase cooling
  - Turn off parts of the chip that are not used in a given clock cycle

Switch from uniprocessors to multiprocessors

- All desktop and server companies are shipping microprocessors with multiple processors per chip
  - Multi-core processor
    - Dual core processor, quad core processor, etc.
  - Improve throughput rather than response time



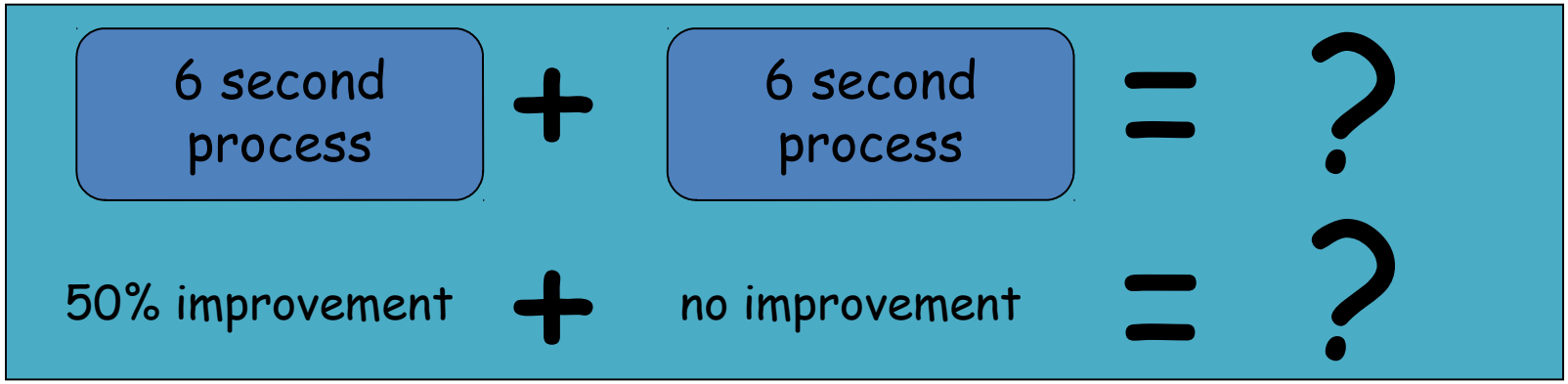
- In the past, program performance can be doubled every 18 months without having to change a line of code.
- Today, to further improve performance, program needs to be explicitly designed to run on parallel processors.

Amdahl's law

- The performance enhancement possible with a given improvement is limited by the amount that the improved feature is used.

$$\text{Execution time after improvement} = \frac{\text{Execution time affected by improvement}}{\text{Amount of improvement}} + \text{Execution time unaffected}$$

  - A chain is only as strong as its weakest link.



Example

- Suppose that program runs in 100 seconds on a computer.
- Multiply operation responsible for 80 seconds of the total time.

- Question: how much should the multiplication operation be improved if the program needs to be run five times faster?

$$\text{Execution time after improvement} = \frac{80 \text{ seconds}}{n} + (100 - 80 \text{ seconds})$$

- Since the performance should be five times faster

$$20 \text{ seconds} = \frac{80 \text{ seconds}}{n} + 20 \text{ seconds}$$

$$0 = \frac{80 \text{ seconds}}{n}$$

Millions instructions per second

- A measurement of program execution speed based on the number of millions of instructions.

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

  - Faster computer means bigger MIPS.

- Problems with using MIPS for comparing computers
  - Different instruction set
  - MIPS varies between programs on the same computer

$$\text{MIPS} = \frac{\text{Instruction count}}{\frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}} \times 10^6} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}$$

  - MIPS can vary independently from performance.