

CSC 411

Computer Organization (Spring 2022)
Lecture 4: Performance

Prof. Marco Alvarez, University of Rhode Island

Quick notes

- Getting access to a Linux machine
 - download your favorite OS and install
 - standalone or side-by-side with Windows
 - built-in Unix subsystem for Windows 10
 - <https://docs.microsoft.com/en-us/windows/wsl/install-win10>
 - use a virtual machine

SI prefixes

Factor	Name	Symbol	Factor	Name	Symbol
10^{24}	yotta	Y	10^{-1}	deci	d
10^{21}	zetta	Z	10^{-2}	centi	c
10^{18}	exa	E	10^{-3}	milli	m
10^{15}	peta	P	10^{-6}	micro	μ
10^{12}	tera	T	10^{-9}	nano	n
10^9	giga	G	10^{-12}	pico	p
10^6	mega	M	10^{-15}	femto	f
10^3	kilo	k	10^{-18}	atto	a
10^2	hecto	h	10^{-21}	zepto	z
10^1	deka	da	10^{-24}	yocto	y

“The 20 SI prefixes used to form decimal multiples and submultiples of SI units.”

<https://physics.nist.gov/cuu/Units/prefixes.html>

Prefixes for binary multiples

Factor	Name	Symbol	Origin	Derivation
2^{10}	kibi	Ki	kilobinary: $(2^{10})^1$	kilo: $(10^3)^1$
2^{20}	mebi	Mi	megabinary: $(2^{10})^2$	mega: $(10^3)^2$
2^{30}	gibi	Gi	gigabinary: $(2^{10})^3$	giga: $(10^3)^3$
2^{40}	tebi	Ti	terabinary: $(2^{10})^4$	tera: $(10^3)^4$
2^{50}	pebi	Pi	petabinary: $(2^{10})^5$	peta: $(10^3)^5$
2^{60}	exbi	Ei	exabinary: $(2^{10})^6$	exa: $(10^3)^6$

“In December 1998 the International Electrotechnical Commission (IEC), the leading international organization for worldwide standardization in electrotechnology, approved as an IEC International Standard names and symbols for prefixes for binary multiples for use in the fields of data processing and data transmission.”

<https://physics.nist.gov/cuu/Units/binary.html>

Examples

Examples and comparisons with SI prefixes

one **kibibit** 1 Kibit = 2^{10} bit = **1024 bit**

one **kilobit** 1 kbit = 10^3 bit = **1000 bit**

one **byte** 1 B = 2^3 bit = **8 bit**

one **mebibyte** 1 MiB = 2^{20} B = **1 048 576 B**

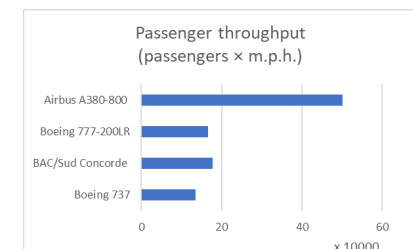
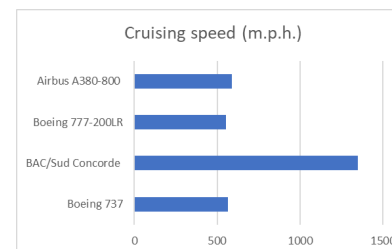
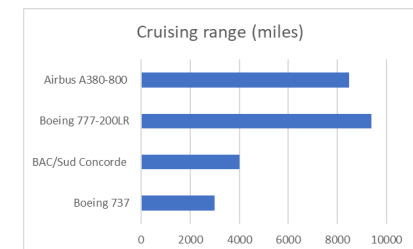
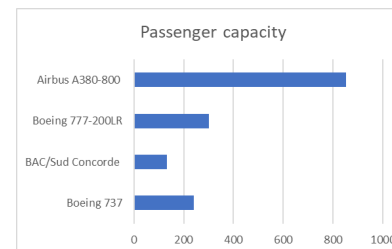
one **megabyte** 1 MB = 10^6 B = **1 000 000 B**

one **gibibyte** 1 GiB = 2^{30} B = **1 073 741 824 B**

one **gigabyte** 1 GB = 10^9 B = **1 000 000 000 B**

Performance

Think about performance ...



Understanding performance

- Algorithm
 - determines number of operations executed
- Programming language, compiler, architecture
 - determines number of machine instructions executed per operation
- Processor and memory system
 - determines how fast instructions are executed
- I/O system (including OS)
 - determines how fast I/O operations are executed

Response time and throughput

- Response time
 - how long it takes to do a task
- Throughput
 - total work done per unit time
 - e.g., tasks/transactions/... per hour
- We'll focus on **response time** for now...

Quizz

- In the following scenarios, are the changes (A) decreasing response time, (B) increasing throughput, or (C) both?
 - replacing the processor in a computer with a faster version
 - adding additional processors to a system that uses multiple processors for separate tasks

Performance and speedup

- Performance

$$\text{performance}_x = \frac{1}{\text{execution time}_x}$$

- Relative performance (speedup):

- “X is **n** times faster than Y”

$$\frac{\text{performance}_x}{\text{performance}_y} = \frac{\text{execution time}_y}{\text{execution time}_x} = n$$

- Example: time taken to run a program

- 10s on A, 15s on B
- $n = 15s / 10s = 1.5$
- A is **1.5** times faster than B

Measuring execution time

- **Elapsed time** (wall clock time, response time)
 - total response time, including all aspects: processing, I/O, OS overhead, idle time
- **CPU execution time** (CPU time)
 - discounts I/O time, other jobs' shares
 - comprises user **CPU time** and **system CPU time** (difficult to separate accurately)
- **Different applications are affected by different performance aspects**
 - one must have a clear definition of what performance metric to use to improve system/program performance

```
$ time ls
```

```
$ time host uri
```

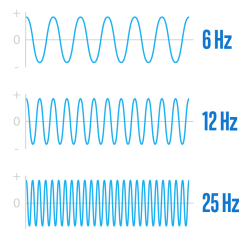
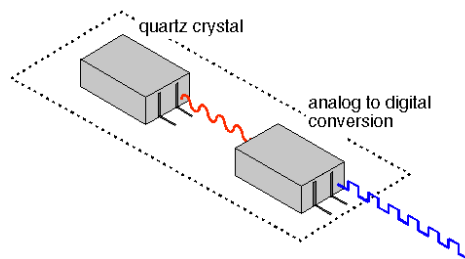
```
$ time (seq 1000 | wc -l)
```

```
$ time (seq 10000000 | wc -l)
```

The time command may show different outputs if run on a Linux or Mac systems, in general it shows “elapsed time”, “CPU time”, and “system time”

CPU clocking

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

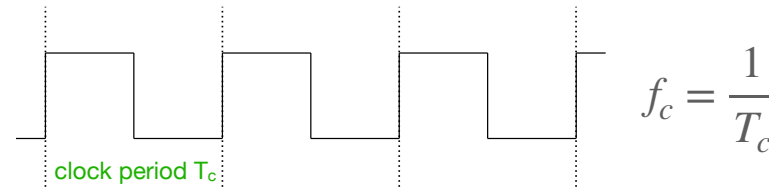


“A clock cycle is a single electronic pulse of a CPU. During each cycle, a CPU can perform a basic operation. Most CPU processes require multiple clock cycles.”

https://www.engineersgarage.com/how_to/how-clock-in-computer-works/
<https://www.intel.com/content/www/us/en/gaming/resources/cpu-clock-speed.html>

CPU clocking

- **Clock period:** duration of a clock cycle
 - e.g., 250ps = 0.25ns = 250×10⁻¹²s
- **Clock frequency (rate):** cycles per second
 - e.g., 4.0GHz = 4000MHz = 4.0×10⁹Hz



$$f_c = \frac{1}{T_c}$$

CPU time

- CPU execution time

$$\begin{aligned}\text{CPU time} &= \text{CPU clock cycles} \times \text{clock cycle time} \\ &= \frac{\text{CPU clock cycles}}{\text{clock rate}}\end{aligned}$$

- Performance improved by:
 - Reducing number of clock cycles
 - Increasing clock rate
 - Hardware designer must often trade off clock rate against cycle count

Example

- Computer A: 2GHz clock, 10s CPU time
- Designing Computer B aiming for 6s CPU time
 - can do faster clock, but causes 1.2 times the number of clock cycles required by the program
- What clock rate should the designer target?

$$\text{clock cycles}_A = \text{CPU time}_A \times \text{clock rate}_A = 10s \times 2GHz = 20 \times 10^9$$

$$\text{clock rate}_B = \frac{\text{clock cycles}_B}{\text{CPU time}_B} = \frac{1.2 \times \text{clock cycles}_A}{6s}$$

$$\text{clock rate}_B = \frac{1.2 \times 20 \times 10^9}{6s} = \frac{24 \times 10^9}{6s} = 4GHz$$

Instruction count and CPI

- Instruction Count for a program
 - previous formula does not consider the number of instructions
 - low level instructions are determined by the program, ISA and compiler
- Average cycles per instruction (CPI)
 - average of clock cycles each instruction takes to execute, over all the instructions executed by the program

$$\text{CPU clock cycles} = \text{instruction count} \times \text{CPI}$$

$$\text{CPU time} = \text{instruction count} \times \text{CPI} \times \text{clock cycle time}$$

$$= \frac{\text{instruction count} \times \text{CPI}}{\text{clock rate}}$$

Example

- Computer A: Cycle Time = 250ps, CPI = 2.0
- Computer B: Cycle Time = 500ps, CPI = 1.2
- Same ISA, which computer is **faster**, and **by how much**?

$$\begin{aligned}\text{CPU time}_A &= ic \times \text{CPI}_A \times \text{cycle time}_A \\ &= ic \times 2.0 \times 250ps = ic \times 500ps\end{aligned}$$

$$\begin{aligned}\text{CPU time}_B &= ic \times \text{CPI}_B \times \text{cycle time}_B \\ &= ic \times 1.2 \times 500ps = ic \times 600ps\end{aligned}$$

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

CPI in more detail

- If different instruction classes take different numbers of cycles

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

- Calculating the (average) CPI using a weighted average

$$\text{CPI} = \frac{\text{CPU clock cycles}}{\text{instruction count}} = \sum_{i=1}^n \left(\text{CPI}_i \times \frac{\text{instruction count}_i}{\text{instruction count}} \right)$$

CPI example

- Alternative compiled code sequences using instructions in classes A, B, C. Which code sequence executes the most instructions? Which sequence is faster? What is the CPI for each sequence?

Class	A	B	C
CPI for class	1	2	3
IC in sequence 1	2	1	2
IC in sequence 2	4	1	1

- Sequence 1

- clock cycles = $2 \times 1 + 1 \times 2 + 2 \times 3 = 10$
- average CPI = $10/5 = 2.0$

- Sequence 2

- clock cycles = $4 \times 1 + 1 \times 2 + 1 \times 3 = 9$
- average CPI = $9/6 = 1.5$

Quizz

- Consider 3 different processors executing the same instructions
 - P1 has a clock cycle time of 0.33ns and a CPI of 1.5
 - P2 has a clock cycle time of 0.40ns and a CPI of 1
 - P3 has a clock cycle time of 0.25ns and a CPI of 2.2
- Questions
 - which has the highest clock rate? what is it?
 - which is the fastest computer? which is the slowest?

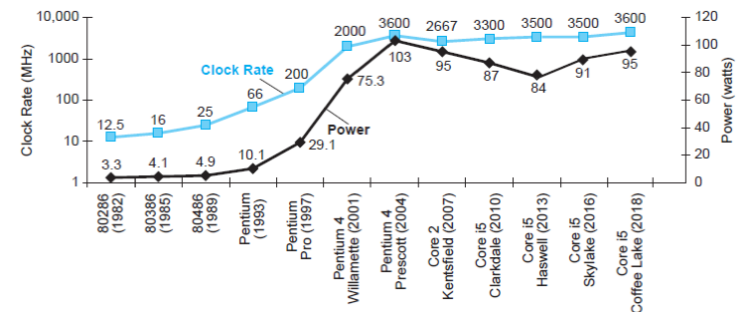
Performance summary

- Performance depends on:
 - **algorithm**: affects IC, possibly CPI
 - **programming language**: affects IC, CPI
 - **compiler**: affects IC, CPI
 - **ISA**: affects IC, CPI, T_c

The Power Wall

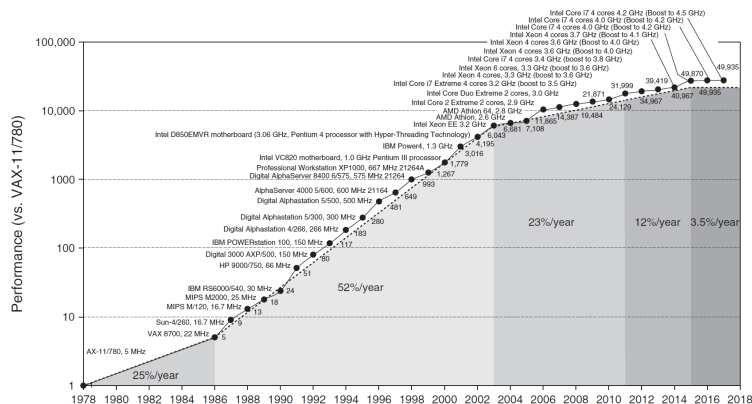
Power trends

- ▶ Clock rate and power are correlated
- $\text{Power} \propto 1/2 \times \text{capacitive load} \times \text{voltage}^2 \times \text{frequency}$
- can't reduce voltage further, can't remove more heat




Uniprocessor performance (response time)

- ▶ Multicore microprocessors
- requires **parallel programming**



Benchmarking Processors



Standard Performance Evaluation Corporation

[Home](#)
[Benchmarks](#)
[Tools](#)
[Results](#)
[Contact](#)
[Site Map](#)
[Search](#)
[Help](#)

Benchmarks

- Cloud
- CPU
- Graphics/Workstations
- ACCEL/MPI/OMP
- Java Client/Server
- Mail Servers
- Storage
- Power
- Virtualization
- Web Servers
- Results Search
- Submitting Results
 - Cloud/CPU/Java/Power
 - Storage/Virtualization
 - ACCEL/MPI/OMP
 - SPECcapc/SPECviewer/SPECwpcc

The Standard Performance Evaluation Corporation (SPEC) is a non-profit corporation formed to establish, maintain and endorse standardized benchmarks and tools to evaluate performance and energy efficiency for the newest generation of computing systems. SPEC develops benchmark suites and also reviews and publishes submitted results from our [member organizations](#) and other benchmark licensees.

What's New:

12/15/2020: SPEC has released an updated storage benchmark, [SPECstorage Solution 2020](#). This benchmark includes new workloads for artificial intelligence (AI) and genomics, expanded custom workload capabilities, massively better scaling, and a statistical visualization mechanism for displaying benchmark results. More details are available in ["What's New in SPECstorage Solution 2020"](#).

11/20/2020: SPEC has released SPEC CPU 2017 v1.1.5, with improvements to the sysinfo utility, better compatibility with GCC version 10, and various minor bug fixes. More details are available in ["Changes in v1.1.5"](#). All users are encouraged to update their copy by using the [runcpu update option](#).

SPEC CPU 2017 Basics

Q6. What does SPEC CPU 2017 measure?

SPEC CPU 2017 focuses on compute intensive performance, which means these benchmarks emphasize the performance of:

- **Processor** - The CPU chip(s).
- **Memory** - The memory hierarchy, including caches and main memory.
- **Compilers** - C, C++, and Fortran compilers, including optimizers.

SPEC CPU 2017 intentionally depends on all three of the above - not just the processor.

SPEC CPU 2017 is not intended to stress other computer components such as networking, graphics, Java libraries, or the I/O system. Note that there are [other](#) SPEC benchmarks that focus on those areas.

Q8. What does SPEC provide?

SPEC CPU 2017 is distributed as an ISO image that contains:

- Source code for the benchmarks
- Data sets
- A tool set for compiling, running, validating and reporting on the benchmarks
- Pre-compiled tools for a variety of operating systems
- Source code for the SPEC CPU 2017 tools, for systems not covered by the pre-compiled tools
- Documentation
- Run and reporting rules

The documentation is also available at [www.spec.org/cpu2017/Docs/index.html](#), including the [Unix](#) and [Windows](#) installation guides.

Q12. What is a CPU 2017 "suite"?

A *suite* is a set of benchmarks that are run as a group to produce one of the overall metrics.

SPEC CPU 2017 includes four suites that focus on different types of compute intensive performance:

Short Tag	Suite	Contents	Metrics	How many copies? What do Higher Scores Mean?
intspeed	SPECspeed@2017 Integer	10 integer benchmarks	SPECspeed2017_int_base SPECspeed2017_int_peak	SPECspeed suites always run one copy of each benchmark.
fpspeed	SPECspeed@2017 Floating Point	10 floating point benchmarks	SPECspeed2017_fp_base SPECspeed2017_fp_peak	Higher scores indicate that less time is needed.
intrate	SPECrate@2017 Integer	10 integer benchmarks	SPECrate2017_int_base SPECrate2017_int_peak	SPECrate suites run multiple concurrent copies of each benchmark. The tester selects how many.
fprate	SPECrate@2017 Floating Point	13 floating point benchmarks	SPECrate2017_fp_base SPECrate2017_fp_peak	Higher scores indicate more <i>throughput</i> (work per unit of time).

Q15. What are "SPECspeed" and "SPECrate" metrics?

There are many ways to measure computer performance. Among the most common are:

- Time - For example, seconds to complete a workload.
- Throughput - Work completed per unit of time, for example, jobs per hour.

SPECspeed is a time-based metric; SPECrate is a throughput metric.

- The **base** metrics (such as SPECspeed2017_int_base) require that all modules of a given language in a suite must be compiled using the same flags, in the same order. All reported results must include the **base** metric.
- The optional **peak** metrics (such as SPECspeed2017_int_peak) allow greater flexibility. Different compiler options may be used for each benchmark, and feedback-directed optimization is allowed.

SPEC CPU2017 Results

These results have been submitted to SPEC; see [the disclaimer](#) before studying any results.

Available Results

Browse

The following are sets of available results since the announcement of the benchmark in June 2017.

- [All CPU2017 Results](#)

Results from all publication quarters, broken out by reported metric:

- Speed:
[[SPECspeed 2017 Integer](#), [SPECspeed 2017 Floating Point](#)]
- Throughput:
[[SPECrate 2017 Integer](#), [SPECrate 2017 Floating Point](#)]

<https://www.spec.org/cpu2017/results/>

SPEC CPU benchmark

- Summaries as **geometric mean** of performance ratios

$$\sqrt[n]{\prod_{i=1}^n \text{execution time ratio}_i}$$

- Always summarize relative performance with the **geometric mean**
 - not affected by the baseline performance