

Performance of Computer Systems

Presentation C

slides by Gojko Babić

[Studying Assignment: Chapter 4](#)

Performance in General

- "X is n times faster than Y" means:

$$n = \frac{\text{Performance (X)}}{\text{Performance (Y)}}$$

- Speed of Concorde vs. Boeing 747
- Throughput of Boeing 747 vs. Concorde
- Cost is also an important parameter in the equation, which is why Concorde was put to pasture!
- **The Bottom Line:**
Performance and Cost or Cost and Performance?

g. babic

Presentation C

2

Basic Performance Metrics

- **Response time:** the time between the start and the completion of a task (in time units)
- **Throughput:** the total amount of tasks done in a given time period (in number of tasks per unit of time)
- **Example:** Car assembly factory:
 - 4 hours to produce a car (response time),
 - 6 cars per an hour produced (throughput)

In general, there is no relationship between those two metrics,
– throughput of the car assembly factory may increase to 18 cars per an hour without changing time to produce one car.
– How?

g. babic

Presentation C

3

Computer Performance: Introduction

- The computer user is interested in response time (or execution time) – the time between the start and completion of a given task (program).
- The manager of a data processing center is interested in throughput – the total amount of work done in given time.
- The computer user wants response time to decrease, while the manager wants throughput increased.
- Main factors influencing performance of computer system are:
 - processor and memory,
 - input/output controllers and peripherals,
 - compilers, and
 - operating system.

g. babic

Presentation C

4

CPU Time or CPU Execution Time

CPU time (or CPU Execution time) is the time between the start and the end of execution of a given program. This time accounts for the time CPU is computing the given program, including operating system routines executed on the program's behalf, and it does not include the time waiting for I/O and running other programs.

- CPU time is a true measure of processor/memory performance.
- Performance of processor/memory = $1 / \text{CPU_time}$

g. babic

Presentation C

5

Analysis of CPU Time

CPU time depends on the program which is executed, including:

- the number of instructions executed,
- types of instructions executed and their frequency of usage.

Computers are constructed in such way that events in hardware are synchronized using a **clock**.

Clock rate is given in Hz (=1/sec).

A clock rate defines durations of discrete time intervals called **clock cycle times** or **clock cycle periods**:

$$\text{clock_cycle_time} = 1/\text{clock_rate (in sec)}$$

Thus, when we refer to different instruction types (from performance point of view), we are referring to instructions with different number of clock cycles required (needed) to execute.

g. babic

Presentation C

6

CPU Time Equation

- $CPU_time = Clock_cycles_for_a_program \cdot Clock_cycle_time$
 $= Clock_cycles_for_a_program / Clock_rate$
 Clock cycles for a program is a total number of clock cycles needed to execute all instructions of a given program.
- $CPU_time = Instruction_count \cdot CPI / Clock_rate$
 Instruction count is a number of instructions executed, sometimes referred as the instruction path length.
 CPI – the average number of clock cycles per instruction (for a given execution of a given program) is an important parameter given as:
 $CPI = Clock_cycles_for_a_program / Instructions_count$

g. babic

Presentation C

7

Calculating Components of CPU time

- For an existing processor it is easy to obtain the CPU time (i.e. the execution time) by measurement, and the clock rate is known. But, it is difficult to figure out the instruction count or CPI.
 Newer processors, MIPS processor is such an example, include counters for instructions executed and for clock cycles. Those can be helpful to programmers trying to understand and tune the performance of an application.
- Also, different simulation techniques and queuing theory could be used to obtain values for components of the execution (CPU) time.

g. babic

Presentation C

8

Analysis of CPU Performance Equation

- $CPU_time = Instruction_count \cdot CPI / Clock_rate$
- How to improve (i.e. decrease) CPU time:
 - Clock rate: hardware technology & organization,
 - CPI: organization, ISA and compiler technology,
 - Instruction count: ISA & compiler technology.

Many potential performance improvement techniques primarily improve one component with small or predictable impact on the other two.

g. babic

Presentation C

9

Calculating CPI

The table below indicates frequency of all instruction types executed in a "typical" program and, from the reference manual, we are provided with a number of cycles per instruction for each type.

Instruction Type	Frequency	Cycles
ALU instruction	50%	4
Load instruction	30%	5
Store instruction	5%	4
Branch instruction	15%	2

$$CPI = 0.5 \cdot 4 + 0.3 \cdot 5 + 0.05 \cdot 4 + 0.15 \cdot 2 = 4 \text{ cycles/instruction}$$

g. babic

Presentation C

10

CPU Time: Example 1

Consider an implementation of MIPS ISA with 500 MHz clock and

- each ALU instruction takes 3 clock cycles,
- each branch/jump instruction takes 2 clock cycles,
- each sw instruction takes 4 clock cycles,
- each lw instruction takes 5 clock cycles.

Also, consider a program that during its execution executes:

- x=200 million ALU instructions
- y=55 million branch/jump instructions
- z=25 million sw instructions
- w=20 million lw instructions

Find CPU time. Assume sequentially executing CPU.

g. babic

Presentation C

11

CPU Time: Example 1 (continued)

- a. Approach 1:
 $Clock_cycles_for_a_program = (x \cdot 3 + y \cdot 2 + z \cdot 4 + w \cdot 5)$
 $= 910 \times 10^6 \text{ clock cycles}$
 $CPU_time = Clock_cycles_for_a_program / Clock_rate$
 $= 910 \times 10^6 / 500 \times 10^6 = 1.82 \text{ sec}$
- b. Approach 2:
 $CPI = Clock_cycles_for_a_program / Instructions_count$
 $CPI = (x \cdot 3 + y \cdot 2 + z \cdot 4 + w \cdot 5) / (x + y + z + w)$
 $= 3.03 \text{ clock cycles/instruction}$
 $CPU_time = Instruction_count \times CPI / Clock_rate$
 $= (x + y + z + w) \times 3.03 / 500 \times 10^6$
 $= 300 \times 10^6 \times 3.03 / 500 \times 10^6$
 $= 1.82 \text{ sec}$

g. babic

Presentation C

12

CPU Time: Example 2

Consider another implementation of MIPS ISA with 1 GHz clock and

- each ALU instruction takes 4 clock cycles,
- each branch/jump instruction takes 3 clock cycles,
- each sw instruction takes 5 clock cycles,
- each lw instruction takes 6 clock cycles.

Also, consider the same program as in Example 1.

Find CPI and CPU time. Assume sequentially executing CPU.

$$\text{CPI} = (x \times 4 + y \times 3 + z \times 5 + w \times 6) / (x + y + z + w)$$

$$= 4.03 \text{ clock cycles/instruction}$$

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

$$= (x+y+z+w) \times 4.03 / 1000 \times 10^6$$

$$= 300 \times 10^6 \times 4.03 / 1000 \times 10^6$$

$$= 1.21 \text{ sec}$$

g. babic

Presentation C

13

Calculating CPI

$$\text{CPI} = (x \times 3 + y \times 2 + z \times 4 + w \times 5) / (x + y + z + w)$$

The calculation may not be necessary correct (and usually it isn't) since the numbers of cycles per instruction given don't account for pipeline effects and other advanced design techniques.

g. babic

Presentation C

14

Phases in MIPS Instruction Execution

- We can divide the execution of an instruction into the following five stages:
 - **IF:** Instruction fetch
 - **ID:** Instruction decode and register fetch
 - **EX:** Execution, effective address or branch calculation
 - **MEM:** Memory access (for lw and sw instructions only)
 - **WB:** Register write back (for ALU and lw instructions)

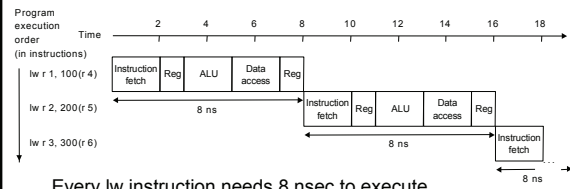
g. babic

Presentation C

15

Sequential Execution of 3 LW Instructions

- Assumed are the following delays: Memory access = 2 nsec, ALU operation = 2 nsec, Register file access = 1 nsec;



Every lw instruction needs 8 nsec to execute.

In this course, we shall design a processor that executes instructions sequentially, i.e. as illustrated here.

g. babic

Presentation C

16

Pipelining: Its Natural!

- Modern processors use advanced hardware design techniques, such as pipelining and out of order execution.

- Dave has four loads of clothes to wash, dry, and fold



- Washer takes 30 minutes



- Dryer takes 40 minutes



- "Folder" takes 20 minutes

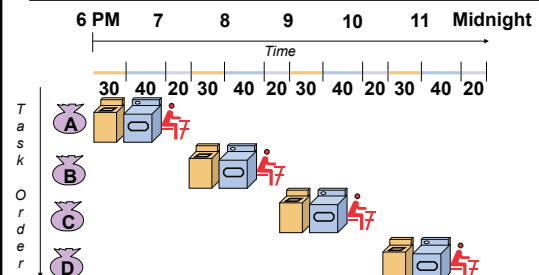


g. babic

Presentation C

17

Sequential Laundry



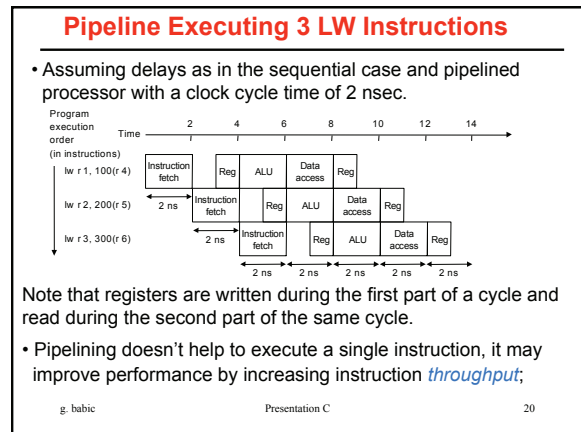
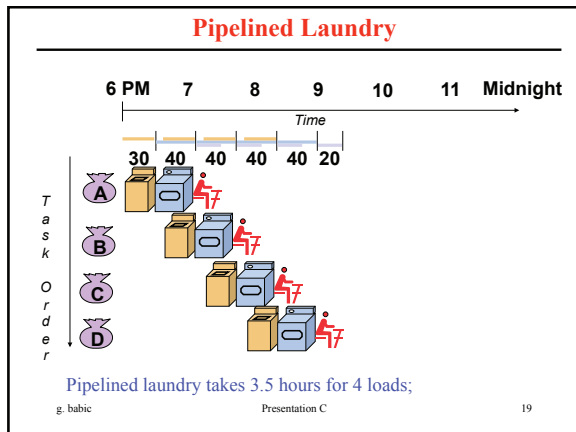
Sequential laundry takes 6 hours for 4 loads;

If Dave learned pipelining, how long would laundry take?

g. babic

Presentation C

18



Quantitative Performance Measures

- The original performance measure was *time to perform an individual instruction*, e.g. add.
- Next performance measure was the *average instruction time*, obtained from the relative frequency of instructions in some typical instruction mix and times to execute each instruction. Since instruction sets were similar, this was a more accurate comparison.
- One alternative to execution time as the metric was *MIPS – Million Instructions Per Second*. For a given program MIPS rating is simple:

$$\text{MIPS rating} = \frac{\text{Instruction_count}}{\text{CPU_time} \cdot 10^6} = \frac{\text{Clock_rate}}{\text{CPI} \cdot 10^6}$$

The problems with MIPS rating as a performance measure:

- difficult to compare computers with different instruction sets,
- MIPS varies between programs on the same computer,
- MIPS can vary inversely with performance!

g. babic Presentation C 21

Quantitative Performance Measures (continued)

- Another popular, misleading and essentially useless measure was *peak MIPS*. That is a MIPS obtained using an instruction mix that minimizes the CPI, even if that instruction mix is totally impractical. Computer manufacturers still occasionally announce products using peak MIPS as a metric, often neglecting to include the work "peak".
- Another popular alternative to execution time was *Million Floating Point Operations Per Second – MFLOPS*:

$$\text{MFLOPS} = \frac{\text{Number_of_floating_point_operations_in_a_program}}{\text{Execution_time} \cdot 10^6}$$

Because it is based on operations in the program rather than on instructions, MFLOPS has a stronger claim than MIPS to being a fair comparison between different machines. MFLOPS are not applicable outside floating-point performance.

g. babic Presentation C 22

Benchmark Suites

Collections of "representative" programs used to measure the performance of processors.

Benchmarks could be:

- real programs;
- modified (or scripted) applications;
- kernels – small, key pieces from real programs;
- synthetic benchmarks – not real programs, but codes try to match the average frequency of operations and operands of a large set of programs.

Examples: Whetstone and Dhrystone benchmarks;

- SPEC (Standard Performance Evaluation Corporation)** was founded in late 1980s to try to improve the state of benchmarking and make more valid base for comparison of desk top and server computers.

g. babic Presentation C 23

SPEC Benchmark Suites

- The SPEC benchmarks are real programs, modified for portability and to minimize the role of I/O in overall benchmark performance. Example: Optimizer GNU C compiler.
- First in 1989, **SPEC89** was introduced with 4 integer programs and 6 floating point programs, providing a single "SPECmarks".
- SPEC92** had 5 integer programs and 14 floating point programs, and provided SPECint92 and SPECfp92.
- SPEC95** provided SPECint_base95, SPECfp_base95.
- SPEC CPU2000** has 12 integer benchmarks and 14 floating point benchmarks, and provides CINT2000 and CFP2000.

g. babic Presentation C 24

Summarizing Performance

- The arithmetic mean of the execution times is given as:

$$\frac{1}{n} \cdot \sum_{i=1}^n \text{Time}_i$$

where Time_i is the execution time for the i th program of a total of n in the workload (benchmark).

- The weighted arithmetic mean of execution times is given as:

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

where Weight_i is the frequency of the i th program in the workload.

- The geometric mean of execution times is given as:

$$\sqrt[n]{\prod_{i=1}^n \text{Time}_i} \quad \text{where} \quad \prod_{i=1}^n X_i = X_1 \cdot X_2 \cdot X_3 \cdot \dots \cdot X_n$$

g. babic

Presentation C

25

Summarizing SPEC CPU2000 Performance

SPEC CPU2000 summarizes performance using a geometric mean of ratios, with larger numbers indicating higher performance.

CINT2000 is indicator of integer performance and it is given as:

$$\text{CINT2000} = k_1 \times \sqrt[12]{\prod_{i=1}^{12} \text{CPU_time_base}_i / \text{CPU_time}_i}$$

where k_1 is a coefficient and CPU_time_i is the CPU time for the i th integer program of a total of 12 programs in the workload.

Similarly for floating point performance, CFP2000 is given as:

$$\text{CFP2000} = k_2 \times \sqrt[14]{\prod_{i=1}^{14} \text{FPExec_time_base}_i / \text{FPExec_time}_i}$$

g. babic

Presentation C

26

Performance Example (part 1/5)

Note: This example is equivalent to Exercises 4.35, 4.36 and 4.37 in the textbook.

- We are interested in two implementations of two similar but still different ISA, one with and one without special real number instructions.
- Both machines have 1000MHz clock.
- Machine With Floating Point Hardware - MFP implements real number operations directly with the following characteristics:
 - real number multiply instruction requires 6 clock cycles
 - real number add instruction requires 4 clock cycles
 - real number divide instruction requires 20 clock cycles
 Any other instruction (including integer instructions) requires 2 clock cycles

g. babic

Presentation C

27

Performance Example (part 2/5)

- Machine with No Floating Point Hardware - MNFP does not support real number instructions, but all its instructions are identical to non-real number instructions of MFP. Each MNFP instruction (including integer instructions) takes 2 clock cycles. Thus, MNFP is identical to MFP without real number instructions.
- Any real number operation (in a program) has to be emulated by an appropriate software subroutine (i.e. compiler has to insert an appropriate sequence of integer instructions for each real number operation). The number of integer instructions needed to implement each real number operations is as follows:
 - real number multiply needs 30 integer instructions
 - real number add needs 20 integer instructions
 - real number divide needs 50 integer instructions

g. babic

Presentation C

28

Performance Example (part 3/5)

Consider Program P with the following mix of operations:

- real number multiply 10%
- real number add 15%
- real number divide 5%
- other instructions 70%

- a. Find MIPS rating for both machines.

$$\text{CPI}_{\text{MFP}} = 0.1 \times 6 + 0.15 \times 4 + 0.05 \times 20 + 0.7 \times 2 = 3.6 \text{ clocks/instr}$$

$$\text{CPI}_{\text{MNFP}} = 2$$

$$\text{MIPS}_{\text{MFP}} \text{ rating} = \frac{\text{clock rate}}{\text{CPI} \times 10^6} = \frac{1000 \times 10^6}{3.6 \times 10^6} = 277.777$$

$$\text{MIPS}_{\text{MNFP}} \text{ rating} = 500$$

According to MIPS rating, MNFP is better than MFP !?

g. babic

Presentation C

29

Performance Example (part 4/5)

- b. If Program P on MFP needs 300,000,000 instructions, find time to execute this program on each machine.

	MFP Number of instructions	MNFP Number of instructions
real mul	30 × 10 ⁶	900 × 10 ⁶
real add	45 × 10 ⁶	900 × 10 ⁶
real div	15 × 10 ⁶	750 × 10 ⁶
others	210 × 10 ⁶	210 × 10 ⁶
Totals	300 × 10 ⁶	2760 × 10 ⁶

$$\text{CPU_time}_{\text{MFP}} = 300 \times 10^6 \times 3.6 / 1000 \times 10^6 = 1.08 \text{ sec}$$

$$\text{CPU_time}_{\text{MNFP}} = 2760 \times 10^6 \times 2 / 1000 \times 10^6 = 5.52 \text{ sec}$$

g. babic

Presentation C

30

Performance Example (part 5/5)

c. Calculate MFLOPS for both computers.

$$\text{MFLOPS} = \frac{\text{Number_of_floating_point_operations_in_a_program}}{\text{Execution_time} \cdot 10^6}$$

$$\text{MFLOPS}_{\text{MFP}} = 90 \times 10^6 / 1.08 \times 10^6 = 83.3$$

$$\text{MFLOPS}_{\text{MNFP}} = 90 \times 10^6 / 5.52 \times 10^6 = 16.3$$