What is the Performance?

Plane	DC to Paris	Speed	Passengers	passengers X mph
Boeing 747	6.5 hours	610 mph	470	286,700
Concorde	3 hours	1350 mph	132	178,200

Which of the planes has better performance

- The plane with the highest speed is **Concorde**
- The plane with the largest capacity is **Boeing 747**

Performance Example

- Time of Concorde vs. Boeing 747?
  - Concord is 1350 mph / 610 mph = 2.2 times faster

- Throughput of Concorde vs. Boeing 747 ?
  - Boeing is 286,700 pmph / 178,200 pmph = 1.6 times faster

- Boeing is 1.6 times faster in terms of throughput
- Concord is 2.2 times faster in terms of flying time

- When discussing processor performance, we will focus primarily on execution time for a single job - why?

Definitions of Time

- Time can be defined in different ways, depending on what we are measuring:
  - Response time : The time between the start and completion of a task. It includes time spent executing on the CPU, accessing disk and memory, waiting for I/O and other processes, and operating system overhead. This is also referred to as execution time.
  - Throughput :The total amount of work done in a given time.
  - CPU execution time : Total time a CPU spends computing on a given task (excludes time for I/O or running other programs). This is also referred to as simply CPU time.

Performance Definition

- For some program running on machine X,

  Performance = 1 / Execution $time_X$
- "X is n times faster than Y"

  $Performance_X$ / $Performance_Y$ = n

Problem:

- machine A runs a program in 20 seconds
- machine B runs the same program in 25 seconds
- how many times faster is machine A?

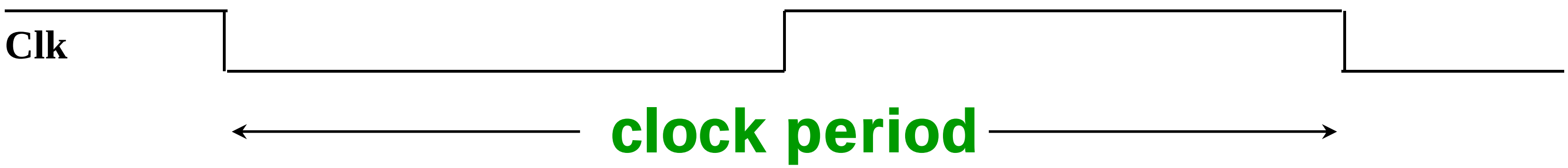
$$\frac{25}{20} = 1.25$$

Basic Measurement Metrics

- Comparing Machines
  - Metrics
    - Execution time
    - Throughput
    - CPU time
    - MIPS – millions of instructions per second
    - MFLOPS – millions of floating point operations per second
- Comparing Machines Using Sets of Programs
  - Arithmetic mean, weighted arithmetic mean
  - Benchmarks

Computer Clock

- A computer clock runs at a constant rate and determines when events take placed in hardware.

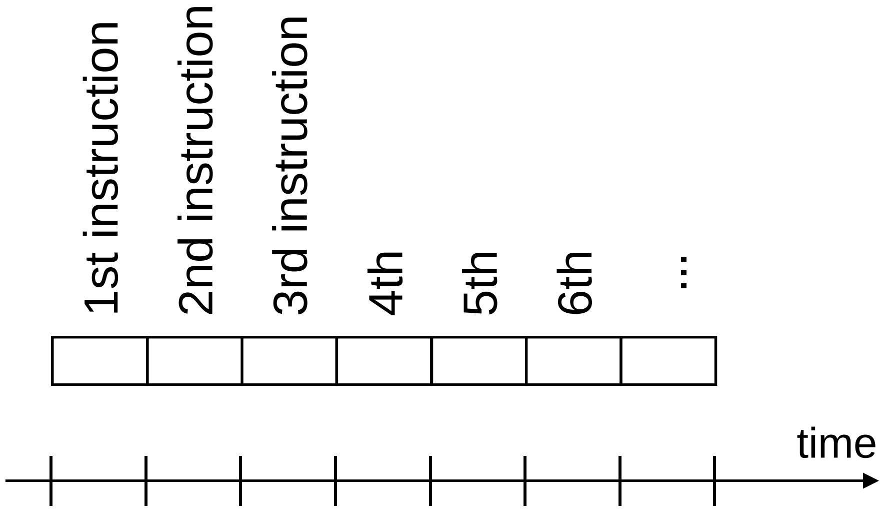


- The clock cycle time is the amount of time for one **clock period** to elapse (e.g. 5 ns).
- The clock rate is the inverse of the clock cycle time.
- For example, if a computer has a clock cycle time of 5 ns, the clock rate is:

$$\frac{1}{5 \times 10^{-9} \text{ sec}} = 200 \text{ MHz}$$

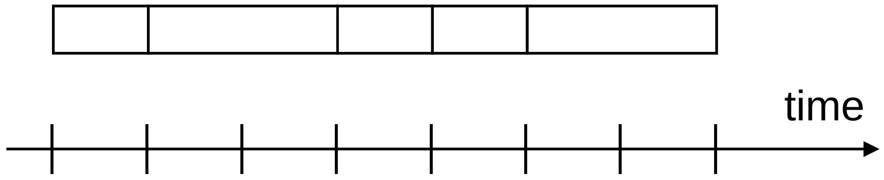
How Many Cycles are Required for a Program?

- Could assume that # of cycles = # of instructions



- This assumption is incorrect, different instructions take different amounts of time on different machines.

Different Numbers of Cycles for Different Instructions



- Division takes more time than addition
- Floating point operations take longer than integer ones
- Accessing memory takes more time than accessing registers

Now That We Understand Cycles

- A given program will require
  - some number of instructions (machine instructions)
  - some number of clock cycles
  - some number of seconds
- We have a vocabulary that relates these quantities:
  - clock cycle time (seconds per cycle)
  - clock rate (cycles per second)
  - CPI (cycles per instruction)
    - *a floating point intensive application might have a higher CPI*

Computing CPU Time

- The time to execute a given program can be computed as

  CPU time = CPU clock cycles x clock cycle time

- Since clock cycle time and clock rate are reciprocals

  CPU time = CPU clock cycles / clock rate

- The number of CPU clock cycles can be determined by

  CPU clock cycles = (instructions/program) x (clock cycles/instruction)

  = Instruction count x CPI

  which gives

  CPU time = Instruction count x CPI x clock cycle time

  CPU time = Instruction count x CPI / clock rate

- The units for CPU time are

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

Which factors are affected by each of the following?

	instr. Count	CPI	clock rate
Program	X		
Compiler	X	X	
Instr. Set Arch.	X	X	
Organization		X	X
Technology			X

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

CPU Time Example

- **Example 1:**
  - CPU clock rate is 1 MHz
  - Program takes 45 million cycles to execute
  - What's the CPU time?

45,000,000 * (1 / 1,000,000) = 45 seconds

- **Example 2:**
  - CPU clock rate is 500 MHz
  - Program takes 45 million cycles to execute
  - What's the CPU time

45,000,000 * (1 / 500,000,000) = 0.09 seconds

CPI Example

- **Example:** Let assume that a benchmark has 100 instructions:

  25 instructions are loads/stores (each take 2 cycles)

  50 instructions are adds (each takes 1 cycle)

  25 instructions are square root (each takes 50 cycles)

What is the CPI for this benchmark?

$$CPI = ((0.25 * 2) + (0.50 * 1) + (0.25 * 50)) = 13.5$$

Computing CPI

- The CPI is the average number of cycles per instruction.
- If for each instruction type, we know its frequency and number of cycles need to execute it, we can compute the overall CPI as follows:

$$CPI = \sum CPI \times F$$

- For example

Op	F	CPI	CPI x F	% Time
ALU	50%	1	.5	23%
Load	20%	5	1.0	45%
Store	10%	3	.3	14%
Branch	20%	2	.4	18%
Total	100%		2.2	100%

Performance

- Performance is determined by execution time
- Do you think any of the variables is sufficient enough to determine computer performance?
  - # of cycles to execute program?
  - # of instructions in program?
  - # of cycles per second?
  - average # of cycles per instruction?
  - average # of instructions per second

- It is not true to think that one of the variables is indicative of performance.

CPI Example

- Suppose we have two implementations of the same instruction set architecture (ISA).

  For some program,

  Machine A has a clock cycle time of **10 ns**. and a CPI of **2.0**
  Machine B has a clock cycle time of **20 ns**. and a CPI of **1.2**

- Which machine is faster for this program, and by how much?

Assume that # of instructions in the program is 1,000,000,000.

CPU $\text{Time}_A = 10^9 * 2.0 * 10 * 10^{-9} = 20$ seconds

CPU $\text{Time}_B = 10^9 * 1.2 * 20 * 10^{-9} = 24$ seconds

Machine A is faster

$$\frac{24}{20} = 1.2 \text{ times}$$

Number of Instruction Example

- A compiler designer is trying to decide between two code sequences for a particular machine. Based on the hardware implementation, there are three different classes of instructions: Class A, Class B, and Class C, and they require one, two, and three cycles (respectively).

The first code sequence has 5 instructions: 2 of A, 1 of B, and 2 of C

The second sequence has 6 instructions: 4 of A, 1 of B, and 1 of C.

- Which sequence will be faster? How much?
- What is the CPI for each sequence?

# of cycles for first code = (2 * 1) + (1 * 2) + (2 * 3) = 10 cycles

# of cycles for second code = (4 * 1) + (1 * 2) + (1 * 3) = 9 cycles

10 / 9 = 1.11 times

CPI for first code = 10 / 5 = 2

CPI for second code = 9 / 6 = 1.5

Problems with Arithmetic Mean

- Applications do not have the same probability of being run
- For example, two machines timed on two benchmarks:

	Machine A	**Machine B**
Program 1	2 seconds (%20)	6 seconds (20%)
Program 2	12 seconds (%80)	10 seconds (%80)

Average execution time$_A$ = (2 + 12) / 2 = 7 seconds

Average execution time$_B$ = (6 + 10) / 2 = 8 seconds

Weighted average execution time$_A$ = 2*0.2 + 12*0.8 = 10 seconds

Weighted average execution time$_B$ = 6*0.2 + 10*0.8 = 9.2 seconds

Poor Performance Metrics

- Marketing metrics for computer performance included MIPS and MFLOPS
- MIPS : millions of instructions per second
  - MIPS = instruction count / (execution time x 10^6)
  - For example, a program that executes 3 million instructions in 2 seconds has a MIPS rating of 1.5
  - Advantage : Easy to understand and measure
  - Disadvantages : May not reflect actual performance, since simple instructions do better.
- MFLOPS : millions of floating point operations per second
  - MFLOPS = floating point operations / (execution time x 10^6)
  - For example, a program that executes 4 million fp. instructions in 5 seconds has a MFLOPS rating of 0.8
  - Advantage : Easy to understand and measure
  - Disadvantages : Same as MIPS, only measures floating point

MIPS Example

- Two different compilers are being tested for a 500 MHz. machine with three different classes of instructions: Class A, Class B, and Class C, which require one, two, and three cycles (respectively). Both compilers are used to produce code for a large piece of software.

  The first compiler's code uses 5 billions Class A instructions, 1 billion Class B instructions, and 1 billion Class C instructions.

  The second compiler's code uses 10 billions Class A instructions, 1 billion Class B instructions, and 1 billion Class C instructions.

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

MIPS Example (Con't)

	Instruction counts (in billions) for each instruction class		
Code from	A	B	C
Compiler 1	5	1	1
Compiler 2	10	1	1

CPU Clock cycles$_1$ = $(5 \times 1 + 1 \times 2 + 1 \times 3) \times 10^9 = 10 \times 10^9$

CPU Clock cycles$_2$ = $(10 \times 1 + 1 \times 2 + 1 \times 3) \times 10^9 = 15 \times 10^9$

CPU time$_1$ = $10 \times 10^9 / 500 \times 10^6 = 20$ seconds

CPU time$_2$ = $15 \times 10^9 / 500 \times 10^6 = 30$ seconds

MIPS$_1$ = $(5 + 1 + 1) \times 10^9 / 20 \times 10^6 = 350$

MIPS$_2$ = $(10 + 1 + 1) \times 10^9 / 30 \times 10^6 = 400$

Performance Summary

- The two main measure of performance are
  - execution time : time to do the task
  - throughput : number of tasks completed per unit time
- Performance and execution time are reciprocals. Increasing performance, decreases execution time.
- The time to execute a given program can be computed as:

  CPU time = Instruction count x CPI x clock cycle time

  CPU time = Instruction count x CPI / clock rate
- These factors are affected by compiler technology, the instruction set architecture, the machine organization, and the underlying technology.
- When trying to improve performance, look at what occurs frequently => make the common case fast.

Computer Benchmarks

- A benchmark is a program or set of programs used to evaluate computer performance.
- Benchmarks allow us to make performance comparisons based on execution times
- Benchmarks should
  - Be representative of the type of applications run on the computer
  - Not be overly dependent on one or two features of a computer
- Benchmarks can vary greatly in terms of their complexity and their usefulness.

SPEC: System Perf. Evaluation Cooperative

- The SPEC Benchmarks are the most widely used benchmarks for reporting workstation and PC performance.
  - First Round SPEC CPU89
    - 10 programs yielding a single number
  - Second Round SPEC CPU92
    - SPEC CINT92 (6 integer programs) and SPEC CFP92 (14 floating point programs)
    - Compiler flags can be set differently for different programs
  - Third Round SPEC CPU95
    - New set of programs: SPEC CINT95 (8 integer programs) and SPEC CFP95 (10 floating point)
    - Single compiler flag setting for all programs
  - Fourth Round SPEC CPU2000
    - New set of programs: SPEC CINT2000 (12 integer programs) and SPEC CFP2000 (14 floating point)
    - Single compiler flag setting for all programs
  - Value reported is the SPEC ratio
    - CPU time of reference machine / CPU time of measured machine

Examples of SPEC95 Benchmarks

- SPEC ratios are shown for the Pentium and the Pentium Pro (Pentium+) processors

Clock Rate	Pentium SPECint	Pentium+ SPECint	Pentium SPECfp	Pentium+ SPECfp
100 MHz	3.2	N/A	2.6	N/A
150 MHZ	4.3	6.0	3.0	5.1
200 MHZ	5.5	8.0	3.8	6.8

- What can we learn from this information?

1. SPECint shows Pentium+ is 1.4 to 1.45 times faster than Pentium
   SPECfp shows Pentium+ is 1.7 to 1.8 times faster than Pentium
2. Clock rate of the Pentium doubles from 100 MHz to 200 MHz, the SPECint performance improves by only 1.7 and SPECfp performance improves by only 1.46

Example

- Assume that a program runs in 100 seconds on a machine, with multiply operations responsible for 80 seconds. How much do I have to improve the speed of multiplication if I want my program to run 2 times faster.

Execution time after improvement =

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

$$50 \text{ seconds} = \frac{80 \text{ seconds}}{n} + (100 - 80 \text{ seconds})$$

$$n = \frac{80 \text{ seconds}}{30 \text{ seconds}} = 2.67$$

Amdahl's Law

- Speedup due to an enhancement is defined as:

$$\text{Speedup} = \frac{\text{ExTime old}}{\text{ExTime new}} = \frac{\text{Performance new}}{\text{Performance old}}$$

- Suppose that an enhancement accelerates a fraction
- $\textbf{Fraction}_{\textbf{enhanced}}$ of the task by a factor $\textbf{Speedup}_{\textbf{enhanced}}$

$$\text{ExTime}_{new} = \text{ExTime}_{old} \times \left[(1 - \text{Fraction}_{enhanced}) + \frac{\text{Fraction}_{enhanced}}{\text{Speedup}_{enhanced}} \right]$$

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

Example of Amdahl's Law

- Floating point instructions are improved to run twice as fast, but only 10% of the time was spent on these instructions originally. How much faster is the new machine?

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

$$\text{Speedup} = \frac{1}{(1 - 0.1) + 0.1/2} = 1.053$$

- The new machine is 1.053 times as fast, or 5.3% faster.
- How much faster would the new machine be if floating point instructions become 100 times faster?

$$\text{Speedup} = \frac{1}{(1 - 0.1) + 0.1/100} = 1.109$$

Estimating Perf. Improvements

- Assume a processor currently requires 10 seconds to execute a program and processor performance improves by 50 percent per year.
- By what factor does processor performance improve in 5 years?

  $(1 + 0.5)^5 = 7.59$

- How long will it take a processor to execute the program after 5 years?

  $ExTime_{new} = 10/7.59 = 1.32$ seconds

Performance Example

- Computers M1 and M2 are two implementations of the same instruction set.
- M1 has a clock rate of 50 MHz and M2 has a clock rate of 100 MHz.
- M1 has a CPI of 2.8 and M2 has a CPI of 3.2 for a given program.
- How many times faster is M2 than M1 for this program?

$$\frac{\mathbf{ExTime_{M1}}}{\mathbf{ExTime_{M2}}} = \frac{\mathbf{IC_{M1} \times CPI_{M1} / Clock\ Rate_{M1}}}{\mathbf{IC_{M2} \times CPI_{M2} / Clock\ Rate_{M2}}} = \frac{\mathbf{2.8/50}}{\mathbf{3.2/100}} = \mathbf{1.75}$$

- What would the clock rate of M1 have to be for them to have the same execution time?

$2.8 / \text{Clock Rate}_{M1} = 3.2 / 100 \Longrightarrow \text{Clock Rate}_{M1} = 87.5\ \text{MHz}$

Summary of Performance Evaluation

- Good benchmarks, such as the SPEC benchmarks, can provide an accurate method for evaluating and comparing computer performance.
- MIPS and MFLOPS are easy to use, but inaccurate indicators of performance.
- Amdahl’s law provides an efficient method for determining speedup due to an enhancement.
- Make the common case fast!

Summary

- Computer Architecture includes the design of the Instruction Set Architecture (programmer's view) and the Machine Organization (logic designer’s view).
- Levels of abstraction, which consist of an interface and an implementation are useful to manage designs.
- Processor performance increases rapidly, but the speeds of memory and I/0 have not kept pace.
- Computer systems are comprised on datapath, memory, input devices, output devices, and control.