# Performance Definitions

- Performance is in units of things-per-second
  - bigger is better
- If we are primarily concerned with response time
  - $$\text{performance}(x) = \frac{1}{\text{execution\_time}(x)}$$

- " X is n times faster than Y" means

$$n = \frac{\text{performance}(X)}{\text{performance}(Y)} = \frac{\text{execution\_time}(Y)}{\text{execution\_time}(X)}$$

- When is throughput more important than execution time?
- When is execution time more important than throughput?

# Peformance Examples

- Time of Concorde vs. Boeing 747?
  - Concord is 1350 mph / 610 mph = 2.2 times faster
    = 6.5 hours / 3 hours

- Throughput of Concorde vs. Boeing 747 ?
  - Concord is 178,200 pmph / 286,700 pmph = 0.62 "times faster"
  - Boeing is 286,700 pmph / 178,200 pmph = 1.6 "times faster"

- Boeing is 1.6 times ("60%") faster in terms of throughput
- Concord is 2.2 times ("120%") faster in terms of flying time

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

# Understanding Performance

- How are the following likely to effect response time and throughput?
  - Increasing the clock speed of a given processor.
  - Increasing the number of jobs in a system (e.g., having a single computer service multiple users).
  - Increasing the number of processors in a sytem that uses multiple processors (e.g., a network of ATM machines).
- If a Pentium III runs a program in 8 seconds and a PowerPC runs the same program in 10 seconds, how many times faster is the Pentium?

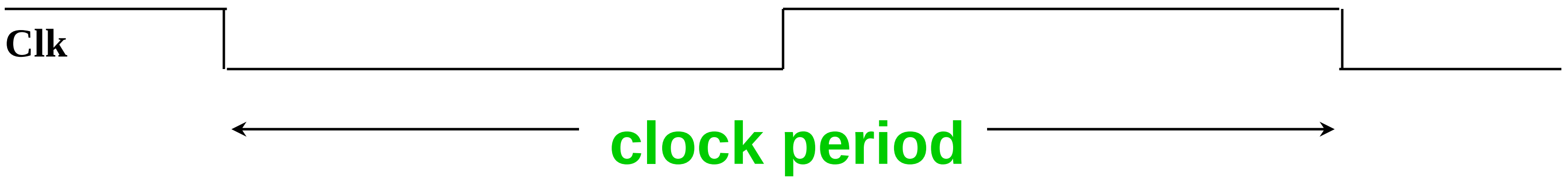
  n = 10 / 8 = 1.25 times faster (or 25% faster)
- Why might someone chose to buy the PowerPC in this case?

# Definitions of Time

- Time can be defined in different ways, depending on what we are measuring:
  - Response time : Total time to complete a task, including time spent executing on the CPU, accessing disk and memory, waiting for I/O and other processes, and operating system overhead.
  - 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.
  - User CPU time : Total time CPU spends in the program
  - System CPU execution time : Total time operating systems spends executing tasks for the program.
- For example, a program may have a system CPU time of 22 sec., a user CPU time of 90 sec., a CPU execution time of 112 sec., and a response time of 162 sec..

# Computer Clocks

- 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:**

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

# 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 this are

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

# Example of Computing CPU time

- If a computer has a clock rate of 50 MHz, how long does it take to execute a program with 1,000 instructions, if the CPI for the program is 3.5?
- Using the equation

  CPU time = Instruction count x CPI / clock rate

  gives

  CPU time = 1000 x 3.5 / (50 x $10^6$) = 70 µsec

- If a computer's clock rate increases from 200 MHz to 250 MHz and the other factors remain the same, how many times faster will the computer be?

$$\frac{\text{CPU time old}}{\text{CPU time new}} = \frac{\text{clock rate new}}{\text{clock rate old}} = \frac{250\text{ MHz}}{200\text{ MHZ}} = 1.25$$

- What simplifying assumptions did we make?

# Our Goal

- Minimize time which is the product, NOT isolated terms
- Common error to miss terms while devising optimizations
  - E.g. ISA change to decrease instruction count
  - BUT leads to CPU organization which makes clock slower
- Bottom line: terms are inter-related

# RISC vs. CISC

$$T_{exec} = N * CPI * cycle\_time$$

- RISCs (Reduced Instruction Set Computers) and CISCs (Complex Instruction Set Computers):
  - RISC CPUs try to reduce $T_{exec}$ by:
    - reducing CPI
    - reducing HW complexity so that we can use faster clocks (shorter cycle times)
  - CISC CPUs try to reduce $T_{exec}$ by:
    - reducing N (number of instructions executed)

# RISC ISA

$T_{exec} = N * CPI * cycle\_time$

1. Load/Store Architecture
2. Instruction semantics are at low level
3. Small number of addressing modes
4. Uniform instruction format

1 & 2 : Fast operation

2 & 3 & 4 : Fetching and decoding of instructions is very quick

<u>Advantages:</u>

- HW is simple
- Logic delays are smaller => we can use faster clock

<u>Disadvantages:</u>

- Because of $2^{nd}$ item => N goes up

Design Challenge:

- Growth in N should be compensated by the reduction in CPI and cycle time

# CISC ISA

$$T_{exec} = N * CPI * cycle\_time$$

1. Many “operate” instructions with at least one memory operand
2. Instruction semantics are at high level
3. Many addressing modes
4. Non-uniform instruction format

1 & 2 : Slow operation

2 & 3 & 4 : decoding of instructions is very slow

Advantages:

- N is smaller

Disadvantages:

- Complex HW
- Larger logic delays => faster clock is not possible
- Higher CPI

Design Challenge:

- Growth in CPI and cycle time should be compensated by the reduction in N

# 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:

$$CPI = \sum_{i=1} CPI_i \times F_i$$

- For example

| Op | $F_i$ | $CPI_i$ | $CPI_i \times F_i$ | % 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 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.

# Types of Benchmarks

| Pros | | Cons |
|---|---|---|
| • representative | Actual Target Workload | • very specific<br>• non-portable<br>• difficult to run, or measure<br>• hard to identify cause |
| • portable<br>• widely used<br>• improvements useful in reality | Full Application Benchmarks (e.g., SPEC benchmarks) | •less representative |
| • easy to run, early in design cycle | Small “Kernel” Benchmarks | • does not measure memory system |
| • identify peak capability and potential bottlenecks | Microbenchmarks | • “peak” may be a long way from application performance |

# SPEC: System Performance Evaluation Cooperative

- The SPEC Bencmarks 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

# Other SPEC Benchmarks

- JVM98:
  - Measures performance of Java Virtual Machines
- SFS97:
  - Measures performance of network file server (NFS) protocols
- Web99:
  - Measures performance of World Wide Web applications
- HPC96:
  - Measures performance of large, industrial applications
- APC, MEDIA, OPC
  - Meausre performance of graphics applications
- For more information about the SPEC benchmarks see: http://www.spec.org.

# 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 | 4.4 | 6.0 | 3.0 | 5.1 |
| 200 MHZ | 5.5 | 8.0 | 3.8 | 6.8 |

° **What can we learn from this information?**

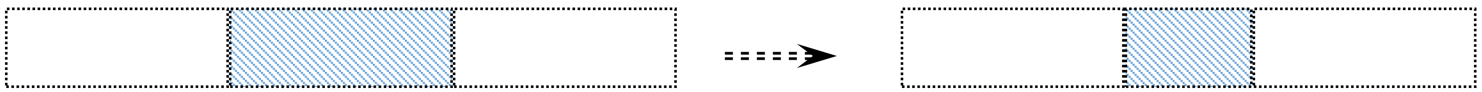
# Poor Performance Metrics

- 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 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

# 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 $\text{Fraction}_{\text{enhanced}}$ of the task by a factor $\text{Speedup}_{\text{enhanced}}$,**

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

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

# 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?

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

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

- **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 Perfomance 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

- What assumptions are made in the above problem?

# 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 75 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?

$$\frac{\mathbf{ExTime_{M1}}}{\mathbf{ExTime_{M2}}} = \frac{\mathbf{IC_{M1} \times CPI_{M1} / Clock\ Rate_{M1}}}{\mathbf{IC_{M2} \times CPI_{M2} / Clock\ Rate_{M2}}} = \frac{\mathbf{2.8/50}}{\mathbf{3.2/75}} = \mathbf{1.31}$$

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

# Summary of Performance Evaluation

- <u>Good</u> 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!