CH3: Assessing and Understanding Performance

- This chapter discusses how to measure, report, and summarize performance and describes the major factors that determine the performance of a computer.

Performance Metrics

- Trying to choose among different computers, performance is almost always an important attribute.
  - Accurately measuring and comparing different computers is critical to purchasers, and therefore to designers.
- Purchasing perspective (i.e., user's perspective):
  - given a collection of machines, which has the
    - best performance ?
    - least cost ?
    - best cost/performance?
- Design perspective (i.e., designer's perspective):
  - faced with design options, which has the
    - best performance improvement ?
    - least cost ?
    - best cost/performance?
- Both require
  - basis for comparison – **benchmark**
  - metric for evaluation – **performance measure**
- So we need to understand the cost and performance implications of different architectural choices.
  - For example:
    - Why is some hardware better than others for different programs?
    - What factors of system performance are hardware related? (e.g., do we need a new machine, or a new operating system?)
    - How does the machine's instruction set affect performance?

Which of these airplanes has the best performance?

Airplane	**Passengers**	**Range (mi)**	**Speed (mph)**	**Passenger throughput**
Boeing 737-100	101	630	598	228,750
Boeing 747	470	4150	610	286,700
BAC/Sud Concorde	132	4000	1350	178,200
Douglas DC-8-50	146	8720	544	79,429

- How much faster is the Concorde compared to the 747?
- How much bigger is the 747 than the Douglas DC-8?

Computer Performance: Basic Metrics

- 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.
- Example: Car assembly factory:
  - 4 hours to produce a car (Response Time)
  - 6 cars per an hour produced (Throughput)
- If we upgrade a machine with a new processor what do we increase?
- If we add a new machine to the lab what do we increase?

Execution Time

- Elapsed Time (Response Time)
  - counts everything (disk and memory accesses, I/O , etc.)
  - a useful number, but often not good for comparison purposes
- CPU time
  - doesn't count I/O or time spent running other programs
  - can be broken up into system time, and user time
- Sometimes, instead of using response time, we use ***CPU time*** to measure performance. ***CPU time*** can also be divided into ***user CPU time*** (program) and ***system CPU time*** (OS).
- Our focus: user CPU time
  - time spent executing the lines of code that are "in" our program.
  - CPU time is a true measure of processor/memory performance.
  - Performance of processor/memory = 1 / CPU_time

Definition of Performance

- For some program running on machine X,

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

- "X is n times faster than Y"

  $\text{Speedup} = \text{Performance}_X / \text{Performance}_Y = \text{Execution time}_Y / \text{Execution time}_X = n$

- A machine is said to be faster or has better performance running this program if the total execution time is shorter.

- Thus the inverse of the total measured program execution time is a possible performance measure or metric

- Example:
  - For a given program:

    Execution time on machine A: $\text{Execution}_A = 1$ second

    Execution time on machine B: $\text{Execution}_B = 10$ seconds

    $\text{Performance}_A / \text{Performance}_B = \text{Execution Time}_B / \text{Execution Time}_A = 10/1 = 10$
  - The performance of machine A is 10 times faster than machine B when running this program.

- Problem:
  - machine A runs a program in 20 seconds
  - machine B runs the same program in 25 seconds

Analysis of CPU Time

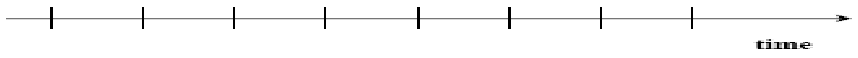
- CPU time depends on the program which is executed, including:
  - a number of instructions executed,
  - types of instructions executed and their frequency of usage.
- Computers are constructed is such way that events in hardware are synchronized using a clock.
- Clock rate is given in Hz (=1/sec).
- A clock rate defines durations of discrete time intervals called clock cycle times or clock cycle periods:
  - clock_cycle_time = 1/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.

Clock Cycles

- Instead of reporting execution time in seconds, we often use cycles

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

- Clock "ticks" indicate when to start activities (one abstraction):

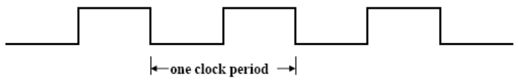


- cycle time = time between ticks = seconds per cycle
- clock rate (frequency) = cycles per second (1 Hz = 1 cycle/sec)

A 4 GHz clock has a $\frac{1}{4\times10^{9}}\times10^{12} = 250 \text{ picoseconds (ps)}$ cycle time

- Clock rate (MHz, GHz) is inverse of clock cycle time (clock period)

  CC = 1 / CR



10 nsec clock cycle => 100 MHz clock rate
500 psec clock cycle => 2 GHz clock rate

How to Improve Performance

- So, to improve performance (everything else being equal) you can either (increase or decrease?)
  _________ the # of required cycles for a program, or
  _________ the clock cycle time or, said another way,
  _________ the clock rate.

Example

- Our favorite program runs in 10 seconds on computer A, which has a 4 GHz. clock. We are trying to help a computer designer build a new machine B, that will run this program in 6 seconds The designer can use new (or perhaps more expensive) technology to substantially increase the clock rate, but has informed us that this increase will affect the rest of the CPU design, causing machine B to require 1.2 times as many clock cycles as machine A for the same program.
  - What clock rate should we tell the designer to target?"

- you can easily work this out from basic principles
  - A program runs in 10s on computer A, at 4GHz
  - How to build a computer B to run this program in 6s
  - The designer has determined that if the clock rate will be increased, it will cause computer B to require 1.2 times more clock cycles than A
  - What clock rate should be used in computer B?

$$\text{CPU time}_A = \frac{\text{CPU clock cycles}_A}{\text{Clock rate}_A} \qquad 10\text{s} = \frac{\text{CPU clock cycles}_A}{4 \times 10^9 \ \frac{\text{cycles}}{\text{seconds}}}$$

$$\text{CPU clock cycles}_A = 40 \times 10^9 \text{ cycles}$$

$$\text{CPU time}_B = \frac{1.2 \times \text{CPU clock cycles}_A}{\text{Clock rate}_B} \qquad \text{Clock rate}_B = 8 \text{ GH}$$

Measuring Time using Clock Cycles

- CPU execution time for program
  = Clock Cycles for a program x Clock Cycle Time
- One way to define clock cycles:
  Clock Cycles for program =
  Instructions for a program (called “Instruction Count”) x Average Clock cycles Per Instruction (called “CPI”)

  - Instruction_count is the number of instructions executed
- CPI – the average number of clock cycles per instructions is an important parameter
  CPI = Clock_cycles_for_a_program / Instruction_count

- Substituting for clock cycles:

  CPU execution time for program= (Instruction Count x CPI) x Clock Cycle Time
  = Instruction Count x CPI x Clock Cycle Time

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

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

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}}$$

- CPI tells us something about the Instruction Set Architecture, the Implementation of that architecture, and the program measured

- **Example:** A Program is running on a specific machine with the following parameters:

  Total instruction count: 10,000,000 instructions
  Average CPI for the program: 2.5 cycles/instruction.
  CPU clock rate: 200 MHz.

  - What is the execution time for this program:

    CPU time = Instruction count x CPI x Clock cycle
    = 10,000,000 x 2.5 x 1 / clock rate
    = 10,000,000 x 2.5 x 5×10^{-9}
    = .125 seconds

- **Performance Comparison: Example**
  - Using the same program with these changes:

    A new compiler used: New instruction count 9,500,000
    New CPI: 3.0
    Faster CPU implementation: New clock rate = 300 MHZ

  - What is the speedup with the changes?

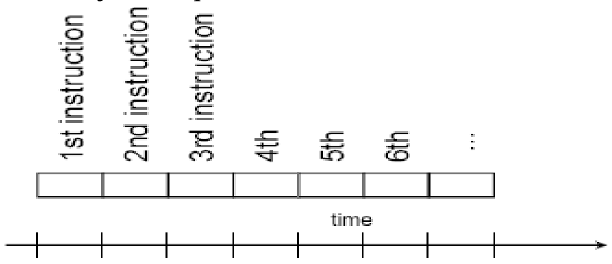
    Speedup = (10,000,000 x 2.5 x 5×10^{-9}) / (9,500,000 x 3 x 3.33×10^{-9})
    = .125 / .095 = 1.32
    or 32 % faster after the changes.

How Calculate the 3 Components?

- Clock Cycle Time: in specification of computer (Clock Rate in advertisements)
- Instruction Count:
  - Use simulator to count instructions
  - Hardware counter in special register (most CPUs)
- CPI= Clock_cycles_for_a_program/Instruction_count
  - Calculate: Clock_cycles_for_a_program = Execution Time / Clock cycle time

How many cycles are required for a program?

- Could assume that number of cycles equals number of instructions



This assumption is incorrect, different instructions take different amounts of time on different machines. Why?
Hint: remember that these are machine instructions, not lines of C code

Different numbers of cycles for different instructions

- Multiplication takes more time than addition
- Floating point operations take longer than integer ones
- Accessing memory takes more time than accessing registers
- Important point: changing the cycle time often changes the number of cycles required for various instructions (more later)

Now that we understand cycles

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

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 250 ps and a CPI of 2.0
      Machine B has a clock cycle time of 500 ps and a CPI of 1.2
      What machine is faster for this program, and by how much?

CPU clock cyclesA = I x 2.0
CPU clock cyclesB = I x 1.2
CPU timeA = CPU clock cyclesA x CPU clock timeA = I x 2.0 x 250ps=Ix500ps
CPU timeB = CPU clock cyclesB x CPU clock timeB = I x 1.2 x 500ps=Ix600ps

$$\frac{\text{CPU time}_B}{\text{CPU time}_A} = 1.2$$

Instruction Frequency & CPI

- Given a program with n types or classes of instructions with the following characteristics:

  C_i = Count of instructions of type$_i$ executed
  CPI_i = Average cycles per instruction of type$_i$
  F_i = Frequency or fraction of instruction type$_i$ executed
  = C_i/ total executed instruction count = C_i/ I

  Then:

$$CPI = \frac{Clock\_Cycles\_for\_a\_Program}{IC}$$

$$CPI = \frac{\sum_{i=1}^{n} C_i * CPI_i}{IC} = \sum_{i=1}^{n} \frac{C_i}{IC} * CPI_i$$

$$CPI = \sum_{i=1}^{n} (CPI_i \times F_i)$$

- Fraction of total execution time for instructions of type i = $\frac{CPI_i \times F_i}{CPI}$

Example1:

- 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?

CPU clock cycles1 = Σ(CPIi x Ci) = (2x1)+(1x2)+(2x3) = 10 cycles
CPU clock cycles2 = Σ(CPIi x Ci) = (4x1)+(1x2)+(1x3) = 9 cycles
Speedup = 10/9 = 1.11
CPI1 = 10/5 = 2
CPI2 = 9/6 = 1.5

- The above example shows the danger of using only one factor (IC) to assess performance.
- When comparing two computers, you must look at all three factors: clock rate, number of instructions, and CPI should be compared

Example 2_A

- 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.

  - Approach 1:

    Clock cycles for a program = $(x\times3 + y\times2 + z\times4 + w\times5)$ = =910×10^{6} clock cycles

    CPU_time = Clock cycles for a program / Clock rate = =$910 \times 10^{6} / 500 \times 10^{6}$ = 1.82 sec

  - Approach 2:

    CPI = Clock cycles for a program / Instructions count

    CPI = $(x\times3 + y\times2 + z\times4 + w\times5)/ (x + y + z + w)$ = 3.03 clock cycles/ instruction

    CPU time = Instruction count × 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 sec

Example 2_B

- 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 2_A. Find CPI and CPU time.

  CPI = $(x\times4 + y\times3 + z\times5 + w\times6)/ (x + y + z + w)$ = 4.03 clock cycles/ instruction

  CPU time = Instruction count $\times$ CPI / 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 sec

Example3: Attempting to Calculate 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*4 + 0.3*5 + 0.05*4 + 0.15*2 = 4 cycles/instruction

Example 4

Op	Freq	CPI_i	Freq x CPI_i
ALU	50%	1	.
Load	20%	5	
Store	10%	3	
Branch	20%	2	
			Σ =

- How much faster would the machine be if a better data cache reduced the average load time to 2 cycles?
- How does this compare with using branch prediction to shave a cycle off the branch time?
- What if two ALU instructions could be executed at once?

% Time $\frac{CPI_i \times F_i}{CPI}$

23% = .5/2.2

45% = 1/2.2

14% = .3/2.2

18% = .4/2.2

Op	Freq	CPI_i	Freq x CPI_i			
ALU	50%	1	.5	.5	.5	.25
Load	20%	5	1.0	.4	1.0	1.0
Store	10%	3	.3	.3	.3	.3
Branch	20%	2	.4	.4	.2	.4
			Σ = 2.2	1.6	2.0	1.95

- How much faster would the machine be if a better data cache reduced the average load time to 2 cycles?
  CPU time new = 1.6 x IC x CC so 2.2/1.6 = 1.375 means 37.5% faster

- How does this compare with using branch prediction to shave a cycle off the branch time?
  CPU time new = 2.0 x IC x CC so 2.2/2.0 = 1.1 means 10% faster

- What if two ALU instructions could be executed at once?
  CPU time new = 1.95 x IC x CC so 2.2/1.95 = 1.128 means 12.8% faster

MIPS (Million Instructions Per Second) Rating

- Note: (MIPS Processor stands for: Microprocessor without Interlocking Pipeline Stages)

- One alternative to time is the metric MIPS (Million Instructions per Second)
- For a specific program running on a specific CPU the MIPS rating is a measure of how many millions of instructions are executed per second
  - The required time to execute one instruction = CPI * CT
  - The # of executed instructions in one second = 1 / (CPI * CT)
  - MIPS = 1 / (CPI * CT * 10^6) = CR / (CPI * 10^6) = IC / (Execution time * 10^6)

- There are three problems with using MIPS as a measure for comparing computers.
  - MIPS does not take into account the capabilities of instructions. We cannot compare computers with different instruction sets using MIPS, since instruction counts will certainly differ.
  - MIPS varies among programs on the same computer; thus a computer cannot have a single MIPS rating for all programs (Program-dependent).
  - MIPS can vary inversely with performance
    - A higher MIPS rating in some cases may not mean higher performance or better execution time. i.e., due to compiler design variations
- Faster execution time usually means faster MIPS rating.

MIPS example1

- Two different compilers are being tested for a 4 GHz. 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 million Class A instructions, 1 million Class B instructions, and 1 million Class C instructions.
  - The second compiler's code uses 10 million Class A instructions, 1 million Class B instructions and 1 million Class C instructions

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

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

CPU clock cycles1 = Σ(CPIi x Ci) = ((5x1) + (1x2) + (1x3))x10^6 = 10x10^6
CPU clock cycles2 = Σ (CPIi x Ci) = ((10x1) + (1x2) + (1x3))x10^6 = 15x10^6
Execution time1 = 2.5 ms
Execution time2 = 3.75 ms

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

MIPS1 = 7 * 10^6 / (2.5 * 10^{-3} * 10^6) = 2800
MIPS2 = 12 * 10^6 / (3.75 * 10^{-3} * 10^6) = 3200

- The code from compiler 2 has a higher MIPS rating, but the code from compiler 1 runs faster!

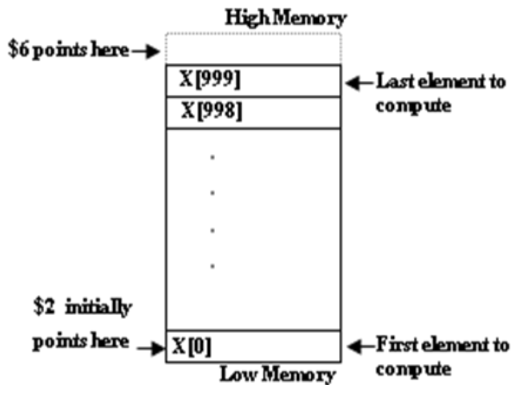
MIPS example2

- Consider the following performance measurements for a program:

Measurement	Computer A	Computer B
Instruction count	10 billion	8 billion
Clock rate	4 GHz	4 GHz
CPI	1.0	1.1

- Which computer has the higher MIPS rating?
- Which computer is faster?

MIPS Loop Performance Example



For the loop:

```
                for (i=0; i<1000; i=i+1){
                          x[i] = x[i] + s; }
```

MIPS assembly code is given by:

```
       lw    $3,  8($1)         # load s in $3
       addi  $6,  $2,  4000     # $6 = address of last element + 4
loop:  lw    $4,  0($2)         # load x[i] in $4
       add   $5,  $4, $3        # $5 has x[i] + s
       sw    $5,  0($2)         # store computed x[i]
       addi  $2,  $2,  4        # increment $2 to point to next x[ ] element
       bne   $6,  $2,  loop     # last loop iteration reached?
```

The MIPS code is executed on a specific CPU that runs at 500 MHz (clock cycle = 2ns = $2x10^{-9}$ seconds) with following instruction type CPIs:

Instruction type	**CPI**
ALU	4
Load	5
Store	7
Branch	3

For this MIPS code running on this CPU finds:

1- Fraction of total instructions executed for each instruction type

2- Total number of CPU cycles

3- Average CPI

4- Fraction of total execution time for each instructions type

5- Execution time

6- MIPS rating

Performance Example

- 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

- 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

- 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.

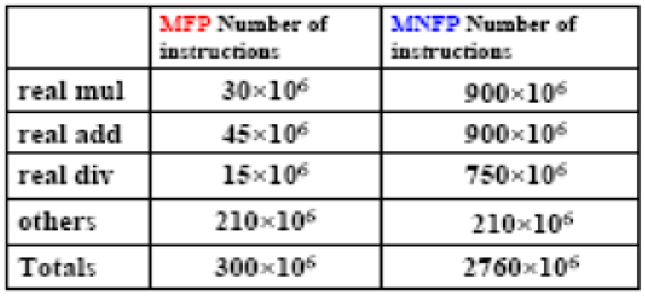
$$CPI_{MFP} = 0.1\times6 + 0.15\times4 + 0.05\times20 + 0.7\times2$$
$$= 3.6 \text{ clocks/instr}$$
$$CPI_{MNFP} = 2$$

$$MIPS_{MFP}\text{rating} = \frac{\text{clock rate}}{\text{CPI} * 10^6} = 270.3$$

$$MIPS_{MNFP}\text{rating} = 500$$

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

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

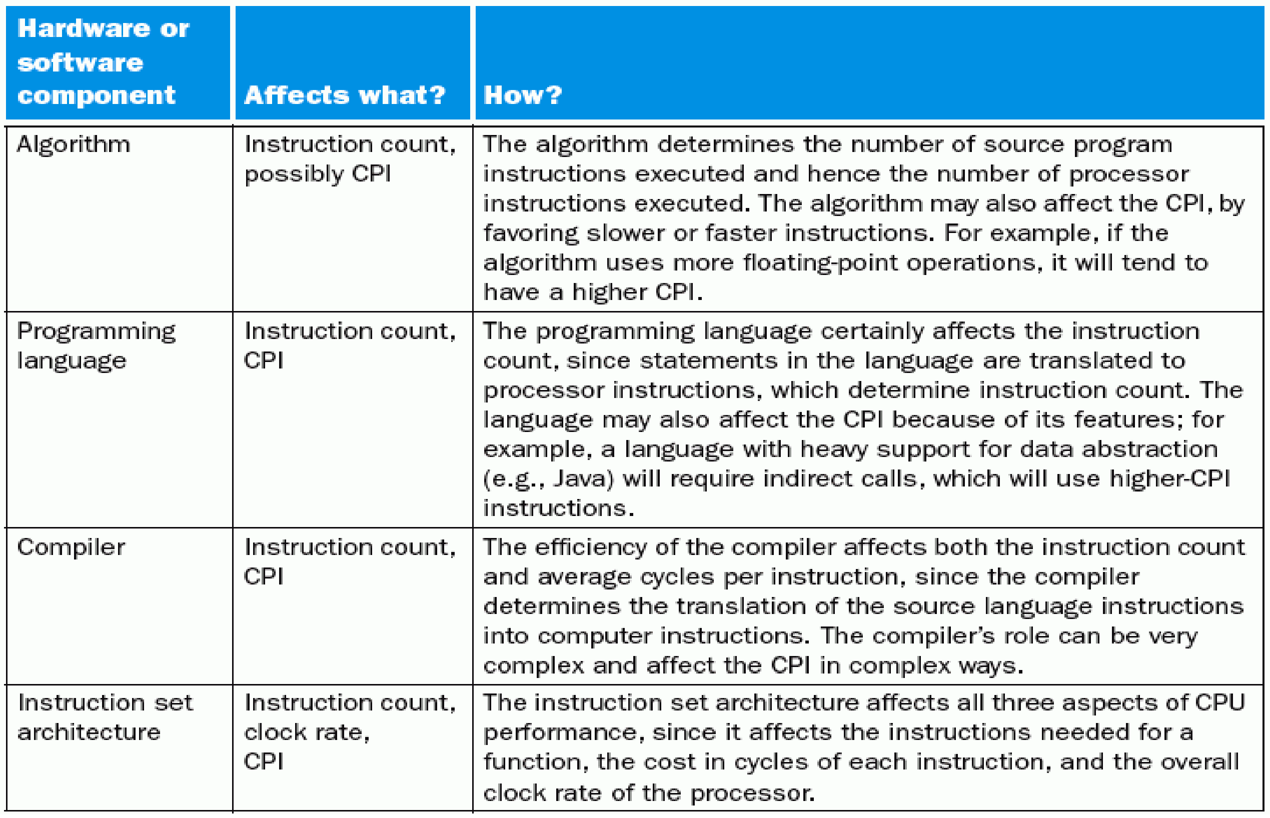


	MFP Number of instructions	MNFP Number of instructions
real mul	30×10^6	900×10^6
real add	45×10^6	900×10^6
real div	15×10^6	750×10^6
others	210×10^6	210×10^6
Totals	300×10^6	2760×10^6

$$CPU\_time_{MFP} = 300\times10^6 \times 3.6 / 1000 \times 10^6 = 1.08 \text{ sec}$$
$$CPU\_time_{MNFP} = 2760\times10^6 \times 2 / 1000 \times 10^6 = 5.52 \text{ sec}$$

Analysis of CPU Performance Equation

- CPU time = Instruction count * CPI / Clock rate
- The performance of a program depends on the algorithm, the language, the compiler, the architecture, and the actual hardware.
- Many potential performance improvement techniques primarily improve one component with small or predictable impact on the other two.
- The instruction count depends on the architecture, but not on the exact implementation
- The CPI depends on a wide variety of design details in the computer, including both the memory system and the processor structure, as well as on the mix of instruction types executed in an application.
    - Thus, CPI varies by application, as well as among implementations with the same instruction set.

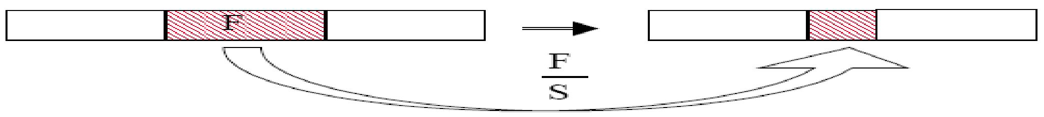


Hardware or software component	Affects what?	How?
Algorithm	Instruction count, possibly CPI	The algorithm determines the number of source program instructions executed and hence the number of processor instructions executed. The algorithm may also affect the CPI, by favoring slower or faster instructions. 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.

Amdahl's Law

Speedup due to enhancement E:

$$\text{Speedup}(E) = \frac{\text{ExTime(without E)}}{\text{ExTime(with E)}} = \frac{\text{Performance(with E)}}{\text{Performance(without E)}}$$



- Suppose that enhancement E accelerates a fraction F of the execution time by a factor S and the remainder of the time is unaffected then:

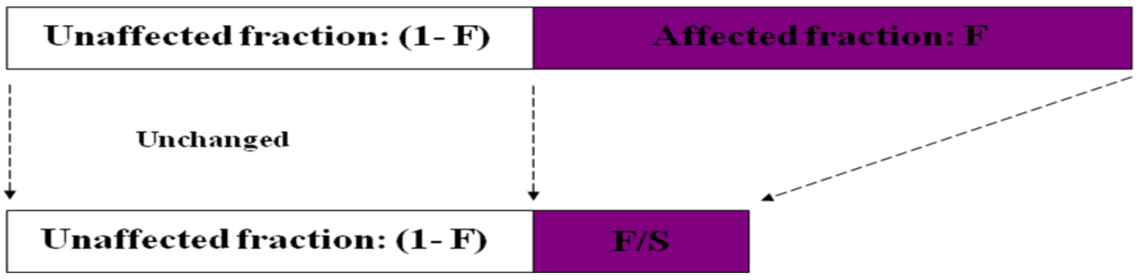
  Execution Time with E = ((1-F) + F/S) X Execution Time without E

  Hence speedup is given by:

$$\text{Speedup}(E) = \frac{\text{Execution Time without E}}{((1 - F) + F/S) \times \text{Execution Time without E}} = \frac{1}{(1 - F) + F/S}$$

- Enhancement E accelerates fraction F of original execution time by a factor of S

  Before:
  Execution Time without enhancement E: (Before enhancement is applied)
  shown normalized to 1 = (1-F) + F =1



  After:
  Execution Time with enhancement E:

$$\text{Speedup}(E) = \frac{\text{Execution Time without enhancement E}}{\text{Execution Time with enhancement E}} = \frac{1}{(1 - F) + F/S}$$

Performance Enhancement Example

- For the RISC machine with the following instruction mix given earlier:

Op	**Freq**	**Cycles**	**CPI(i)**	**% Time**
ALU	50%	1	.5	23%
Load	20%	5	1.0	45%
Store	10%	3	.3	14%
Branch	20%	2	.4	18%

- If a CPU design enhancement improves the CPI of load instructions from 5 to 2, what is the resulting performance improvement from this enhancement:

Solution1:

Fraction enhanced = F = 45% or .45
Unaffected fraction = 100% - 45% = 55% or .55
Factor of enhancement = 5/2 = 2.5
Using Amdahl's Law:

$$\text{Speedup(E)} = \frac{1}{(1 - F) + F/S} = \frac{1}{.55 + .45/2.5} = 1.37$$

Solution2:

Old CPI = 2.2
New CPI = .5 x 1 + .2 x 2 + .1 x 3 + .2 x 2 = 1.6

$$\text{Speedup(E)} = \frac{\text{Original Execution Time}}{\text{New Execution Time}} = \frac{\text{Instruction count x old CPI x clock cycle}}{\text{Instruction count x new CPI x clock cycle}}$$

$$= \frac{\text{old CPI}}{\text{new CPI}} = \frac{2.2}{1.6} = 1.37$$

Which is the same speedup obtained from Amdahl's Law in the first solution.

Example:

- Suppose an enhancement runs 10 times faster than the original machine, but is only usable 40% of the time.
- Question: what is the overall speedup?
  Answer: Fraction_enhance = 0.4
  Speedup_enhanced = 10
  Speedup_overall = 1/(0.6+0.4/10) = 1.56

Example:

- "Suppose a program runs in 100 seconds on a machine, with multiply responsible for 80 seconds of this time. How much do we have to improve the speed of multiplication if we want the program to run 4 times faster?" How about making it 5 times faster?

Solution:

$$\text{Desired speedup} = 4 = \frac{100}{\text{Execution Time with enhancement}}$$

Execution time with enhancement = 100/4 = 25 seconds

25 seconds = (100 - 80 seconds) + 80 seconds / S

25 seconds = 20 seconds + 80 seconds / S

5 = 80 seconds / S

S = 80/5 = 16

Alternatively, it can also be solved by finding enhanced fraction of execution time:

F = 80/100 = .8

and then solving Amdahl's speedup equation for desired enhancement factor S

Remember

- Performance is specific to a particular program/s
  - Total execution time is a consistent summary of performance
- For a given architecture performance increases come from:
  - increases in clock rate (without adverse CPI affects)
  - improvements in processor organization that lower CPI
  - compiler enhancements that lower CPI and/or instruction count
  - Algorithm/Language choices that affect instruction count
- Pitfall: expecting improvement in one aspect of a machine's performance to affect the total performance