Defining Performance

- *Normally interested in reducing*
  - *Response time (execution time) – the time between the start and the completion of a task*
    - *Important to individual users*
  - *Thus, to maximize performance, need to minimize execution time*

$$performance_X = 1 / execution\_time_X$$

If X is n times faster than Y, then

$$\frac{performance_X}{performance_Y} = \frac{execution\_time_Y}{execution\_time_X} = n$$

- *Throughput – the total amount of work done in a given time*
  - *Important to data center managers*
- *Decreasing response time almost always improves throughput*

Comparing and Summarizing Performance

	Computer A	Computer B
Program1(sec)	1	10
Program2(sec)	1000	100
Total time (sec)	1001	110

How to compare the performance?

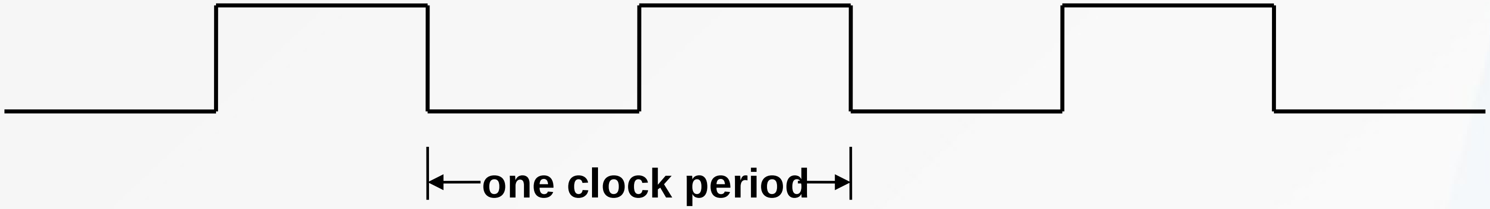
Total Execution Time : A Consistent Summary Measure

$$\frac{Performance\,B}{Performance\,A}=\frac{ExecutionTimeA}{Execution\mathrm{TimeB}}=\frac{1001}{110}=9.1$$

Machine Clock Rate

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

$$CC = 1 / CR$$



10 nsec clock cycle => 100 MHz clock rate

5 nsec clock cycle => 200 MHz clock rate

2 nsec clock cycle => 500 MHz clock rate

1 nsec clock cycle => 1 GHz clock rate

500 psec clock cycle => 2 GHz clock rate

250 psec clock cycle => 4 GHz clock rate

200 psec clock cycle => 5 GHz clock rate

CPU Performance Equation

$$\text{CPU time} = \begin{array}{c}\text{\# CPU clock cycles}\\ \text{for a program}\end{array} \times \text{ clock cycle time}$$

or

$$\text{CPU time} = \frac{\text{\# CPU clock cycles for a program}}{\text{clock rate}}$$

Clock Cycles per Instruction

Not all instructions take the same amount of time to execute
One way to think about execution time is that it equals the number of instructions executed multiplied by the average time per instruction

$$\frac{\text{\# CPU clock cycles}}{\text{for a program}} = \frac{\text{\# Instructions}}{\text{for a program}} \times \frac{\text{Average clock cycles}}{\text{per instruction}}$$

- ❑ *Clock cycles per instruction (CPI) – the average number of clock cycles each instruction takes to execute*

Effective CPI

- *Computing the overall effective CPI is done by looking at the different types of instructions and their individual cycle counts and averaging*

$$\text{Overall effective CPI} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{IC}_i)$$

  - *Where IC_i is the count (percentage) of the number of instructions of class i executed*
  - *CPI_i is the (average) number of clock cycles per instruction for that instruction class*
  - *n is the number of instruction classes*

THE Performance Equation

- Our basic performance equation is then

$$\text{CPU time} = \text{Instruction\_count} \times \text{CPI} \times \text{clock\_cycle}$$

or

$$\text{CPU time} = \frac{\text{Instruction\_count} \times \text{CPI}}{\text{clock\_rate}}$$

Calculating CPI

The table below indicates frequency of all instruction types executed in a “typical” program and, 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

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

- ***Example-1***
- *Suppose a program (or a program task) takes 1 billion instructions to execute on a processor running at 2 GHz. Suppose also that 50% of the instructions execute in 3 clock cycles, 30% execute in 4 clock cycles, and 20% execute in 5 clock cycles. What is the execution time for the program or task?*

- *solution:*
- *We have the instruction count: 10^9 instructions. The clock time can be computed quickly from the clock rate to be 0.5×10-9 seconds. So we only need to to compute clocks per instruction as an effective value:*

Value	*Frequency*	*Product*
3	*0.5*	*1.5*
4	*0.3*	*1.2*
5	*0.2*	*1.0*
CPI =		*3.7*

Then we have

Execution time $= 1.0\times10^9 \times 3.7 \times 0.5\times10^{-9}$ *sec* $= 1.85$ *sec* *example-2*

Suppose the processor in the previous example is redesigned so that all instructions that initially executed in 5 cycles now execute in 4 cycles. Due to changes in the circuitry, the clock rate has to be decreased from 2.0 GHz to 1.9 GHz. What is the overall percentage improvement?

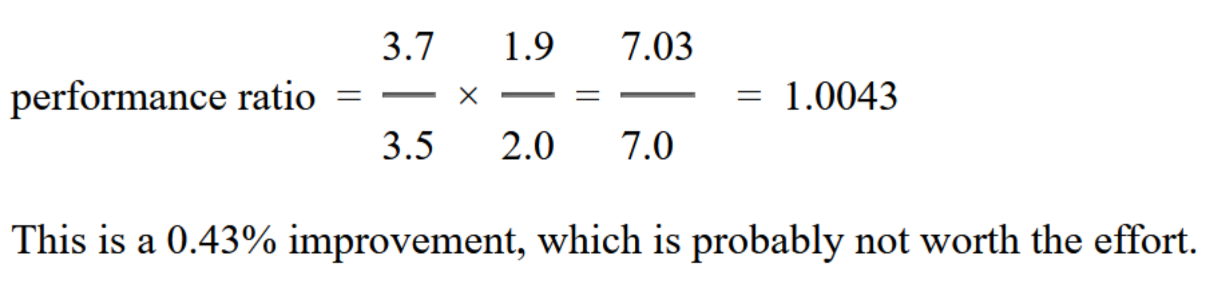
Suppose the processor in the previous example is redesigned so that all instructions that initially executed in 5 cycles now execute in 4 cycles. Due to changes in the circuitry, the clock rate has to be decreased from 2.0 GHz to 1.9 GHz. What is the overall percentage improvement?

solution:

Value	Frequency	Product
3	0.5	1.5
4	0.3	1.2
4	0.2	0.8
CPI =		3.5

Now, lower clocks per instruction means higher instruction throughput and thus better performance, so we expect this part of the performance ratio to be greater than 1.0; that is, 3.7/3.5.

Then we have



$$\text{performance ratio} = \frac{3.7}{3.5} \times \frac{1.9}{2.0} = \frac{7.03}{7.0} = 1.0043$$

This is a 0.43% improvement, which is probably not worth the effort.

A Simple Example

Op	Freq	CPI_i	Freq x CPI_i	Q1	Q2	Q3
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
Overall effective CPI			‹ = 2.2	1.6	2.0	1.95

Q-1How 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 means 37.5% faster

Q-2How does this compare with using branch prediction to save a cycle off the branch time?

CPU time new = 2.0 x IC x CC so 2.2/2.0 means 10% faster

Q-3What if two ALU instructions could be executed at once?

CPU time new = 1.95 x IC x CC so 2.2/1.95 means 12.8% faster

cpu time Example

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(store) instruction takes 4 clock cycles,
- each lw(load) 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(store) instructions
- w=20 million lw(load) instructions

Find CPU time. Assume sequentially executing CPU.

a. Approach 1:

Clock_cycles_for_a_program = $(x \times 3 + y \times 2 + z \times 4 + w \times 5)$
$= 910 \times 10^6$ clock cycles

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

b. Approach 2:

CPI = Clock_cycles_for_a_program / Instructions_count

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

Quiz on Performance equation

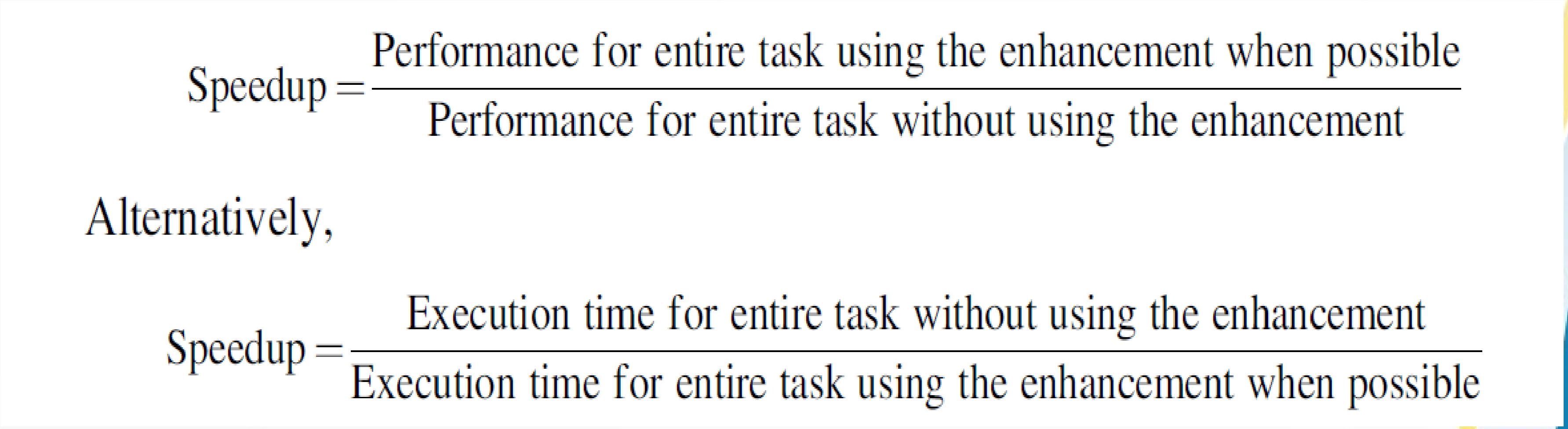
Consider two processors P1 and P2 executing the same instruction set. Assume that under identical conditions, for the same input, a program running on P2 takes 25% less time but incurs 20% more CPI (clock cycles per instruction) as compared to the program running on P1. If the clock frequency of P1 is 1GHz, then the clock frequency of P2 (in GHz) is __________.

Amdahl's LAW

- *performance gain that can be obtained by improving some portion of a computer can be calculated using Amdahl's Law.*
- ***Amdahl's Law states** that the performance improvement to be gained from using some faster mode of execution is limited by the fraction of the time the faster mode can be used.*
- *Amdahl's Law defines the **speedup** that can be gained by using a particular feature.*

What is speedup?

Speedup is the ratio

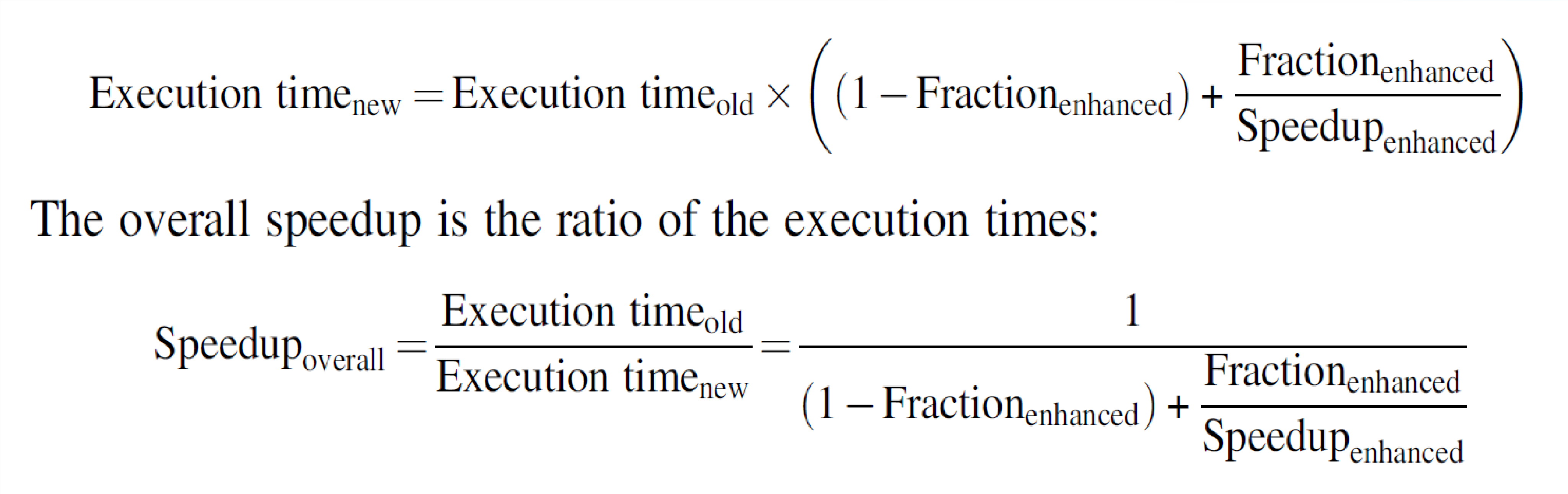


$$\text{Speedup} = \frac{\text{Performance for entire task using the enhancement when possible}}{\text{Performance for entire task without using the enhancement}}$$

Alternatively,

$$\text{Speedup} = \frac{\text{Execution time for entire task without using the enhancement}}{\text{Execution time for entire task using the enhancement when possible}}$$

□ *The execution time using the original computer with the enhanced mode will be the time spent using the unenhanced portion of the computer plus the time spent using the enhancement:*



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

The overall speedup is the ratio of the execution times:

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

Amdahl's law is used to find out overall speedup of the system when some part of the system is enhanced

Example-1 using Amdahl's Law

we have a system in which 40% operations are floating point.Suppose we enhance floating point unit such that it becomes 30 times faster.find overall speedup to the system

solution of Example-1 using Amdahl's Law

Solution:

$fraction_{enhencement}$=0.4

$speedup_{enhencement}$=30

overal speedup =1/((1-0.4)+(0.4/30))

=1/((0.6)+0.014)

=1/(0.614)

=1.62 (Ans)

Example -2 using Amdahl's law

Suppose that we want to enhance the processor used for web serving. The new processor is 10 times faster on computation in the web serving application than the old processor. Assuming that the original processor is busy with computation 40% of the time and is waiting for I/O 60% of the time, what is the overall speedup gained by incorporating the enhancement?

solution of Example-2 using Amdahl's Law

Solution:

$fraction_{enhencement}$=0.4

$speedup_{enhencement}$=10

overal speedup =1/((1-0.4)+(0.4/10))

=1/((0.6)+0.04)

=1/(0.64)

=1.5625 (Ans)

Example -3 Quiz on Amdahl's law

Consider the enhencement to the processor of a web server the enhenced server is 40time faster on search acquires than old processor.old processor is busy with search quries 75% of the time then the speedup gained by integrating enhenced cpu is --------

a)4.72

b)3.72

c)2.79

d)5.72

Example -3 using Amdahl's law

a)4.72

b)3.72(Ans)

c)2.79

d)4.72

sol:

Total speedup=1/((1-.75)+(0.75/40))
=1/(0.26875)
=3.72

Example-4 using Amdahl's Law

A common transformation required in graphics processors is square root. Implementations of floating-point (FP) square root vary significantly in performance,especially among processors designed for graphics.Suppose FP square root (FSQRT) is responsible for 20% of the execution time of a critical graphics benchmark.One proposal is to enhance the FSQRT hardware and speed up this operation by a factor of 10. The other alternative is just to try to make all FP instructions in the graphics processor run faster by a factor of 1.6; FP instructions are responsible for half of the execution time for the application. The design team believes that they can make all FP instructions run 1.6 times faster with the same effort as required for the fast square root. Compare these two design alternatives.

Answer :

We can compare these two alternatives by comparing the speedups:

Solution:

$\text{fraction}_{\text{enhencement\_FSQRT}}$ =0.2

$\text{speedup}_{\text{enhencement\_FSQRT}}$=10

overal speedup_FSQRT =1/((1-0.2)+(0.2/10))

=1/((0.8)+0.02)

=1/(0.82)

=1.2195 (Ans)

Solution:

$\text{fraction}_{\text{enhencement\_FP =0.5}}$

$\text{speedup}_{\text{enhencement\_FP=1.6}}$

overal speedup_FP =1/((1-0.5)+(0.5/1.6))

=1/(0.8125)

=1.23 (Ans)