Esercises on Amdhal Law and Performance Equation

Hennessy Patterson *Computer Architecture: A Quantitative Approach* Fifth Edition
Chapter 1 *Fundamentals of Quantitative Design and Analysis*

1.14 In this exercise, assume that we are considering enhancing a machine by adding vector hardware to it. When a computation is run in vector mode on the vector hardware, it is 10 times faster than the normal mode of execution. We call the percentage of time that could be spent using vector mode the *percentage of vectorization.*

a. Draw a graph that plots the speedup as a percentage of the computation performed in vector mode. Label the *y*-axis "Net speedup" and label the *x*-axis "Percent vectorization."

b. What percentage of vectorization is needed to achieve a speedup of 2?

c. What percentage of the computation run time is spent in vector mode if a speedup of 2 is achieved?

d. What percentage of vectorization is needed to achieve one-half the maximum speedup attainable from using vector mode?

e. Suppose you have measured the percentage of vectorization of the program to be 70%. The hardware design group estimates it can speed up the vector hardware even more with significant additional investment. You wonder whether the compiler crew could increase the percentage of vectorization, instead. What percentage of vectorization would the compiler team need to achieve in order to equal an addition 2× speedup in the vector unit (beyond the initial 10x)?

1.15 Assume that we make an enhancement to a computer that improves some mode of execution by a factor of 10. Enhanced mode is used 50% of the time, measured as a percentage of the execution time *when the enhanced mode is in use*. Recall that Amdahl's law depends on the fraction of the original, *unenhanced* execution time that could make use of enhanced mode. Thus, we cannot directly use this 50% measurement to compute speedup with Amdahl's law.

a. What is the speedup we have obtained from fast mode?

b. What percentage of the original execution time has been converted to fast mode?

1.16 When making changes to optimize part of a processor, it is often the case that speeding up one type of instruction comes at the cost of slowing down something else. For example, if we put in a complicated fast floatingpoint unit, that takes space, and something might have to be moved farther away from the middle to accommodate it, adding an extra cycle in delay to reach that unit. The basic Amdahl's law equation does not take into account this trade-off.

a. If the new fast floating-point unit speeds up floating-point operations by, on average, 2×, and floating-point operations take 20% of the original program's execution time, what is the overall speedup (ignoring the penalty to any other instructions)?

b. Now assume that speeding up the floating-point unit slowed down data cache accesses, resulting in a 1.5× slowdown (or 2/3 speedup). Data cache accesses consume 10% of the execution time. What is the overall speedup now?

c. After implementing the new floating-point operations, what percentage of execution time is spent on floating-point operations? What percentage is spent on data cache accesses?

Stalling Computer organization and Architecture
Chapter 2 Computer Evolution And Performance

A common measure of performance for a processor is the rate at which instructions are executed, expressed as millions of instructions per second (MIPS), referred to as the **MIPS rate**.
We can express the MIPS rate in terms of the clock rate and CPI as follows:

$$\text{MIPS rate} = \frac{IC}{T \times 10^6} = \frac{f}{CPI \times 10^6}$$

For example, consider the execution of a program which results in the execution of 2 million instructions on a 400-MHz processor. The program consists of four major types of instructions. The instruction mix and the CPI for each instruction type are given below based on the result of a program trace experiment:

Instruction Type	**CPI**	**Instruction Mix**
Arithmetic and logic	1	60%
Load/store with cache hit	2	18%
Branch	4	12%
Memory reference with cache miss	8	10%

The average CPI when the program is executed on a uniprocessor with the above trace results is
CPI= 0.6 + (2 x 0.18) + (4 x 0.12) + (8 x 0.1) = 2.24.
The corresponding MIPS rate is $(400 \times 10^6) / (2.24 \times 10^6) \approx 178$.

2.11 Consider two different machines, with two different instruction sets, both of which have a clock rate of 200 MHz. The following measurements are recorded on the two machines running a given set of benchmark programs:

Instruction Type	**Instruction Count (millions)**	**Cycles per Instruction**
Machine A		
Arithmetic and logic	8	1
Load and store	4	3
Branch	2	4
Others	4	3
Machine B		
Arithmetic and logic	10	1
Load and store	8	2
Branch	2	4
Others	4	3

a. Determine the effective CPI, MIPS rate, and execution time for each machine.
b. Comment on the results.

2.12. Early examples of CISC and RISC design are the VAX 11/780 and the IBM RS/6000, respectively. Using a typical benchmark program, the following machine characteristics result:

Processor	**Clock Frequency**	**Performance**	**CPU Time**
VAX 11/780	5 MHz	1 MIPS	12 *x* seconds
IBM RS/6000	25 MHz	18 MIPS	*x* seconds

The final column shows that the VAX required 12 times longer than the IBM measured in CPU time.
a. What is the relative size of the instruction count of the machine code for this benchmark program running on the two machines?
b. What are the CPI values for the two machines?

2.13. Four benchmark programs are executed on three computers with the following results:

	Computer A	**Computer B**	**Computer C**
Program 1	1	10	20
Program 2	1000	100	20
Program 3	500	1000	50
Program 4	100	800	100

The table shows the execution time in seconds, with 100,000,000 instructions executed in each of the four programs. Calculate the MIPS values for each computer for each program.