$$Speedup = \frac{1}{0.5 + \frac{0.5}{5}} = \frac{1}{0.6} = 1.67$$

The new machine will cost $\frac{2}{3} * 1 + \frac{1}{3} * 5 = 2.33$ times the original machine. Since the cost increase is larger than the performance improvement, this change does not improve cost/performance.

1.4.2. Locality of Reference

While Amdahl's Law is a theorem that applies to any computer system, the locality of reference is an important characteristic of programs. According to this principle, programs tend to reuse data and instructions they have used recently. An observation based on programs' behavior is that a program spends 90% of its execution time in only 10% of the code. An implication of locality of reference is that based on the program's recent past, it can be predicted with reasonably accuracy what instructions and data the program will use in the near future.

To examine locality of reference, several measurements were made on programs to determine what percentage of the instructions are responsible for 80% and for 90% of the instructions executed. For example, less than 4% of the *Spice* program instructions (also called the *static* instructions) represent 80% of the dynamically executed instructions, while less than 10% of the static instructions account for 90% of the executed instructions. In the *Spice* program, only 30% of the instructions are executed one or more times.

Locality of reference also applies to data accesses. There are several types of locality that have been observed. *Spatial locality* affirms that items whose addresses are near one another tend to be referenced close together in time. *Temporal locality* states that recently accessed items are likely to be accessed in the near future. These principles are especially useful for memory design.

1.5. Problems

1.5.1. A frequently used program runs in 10 seconds on computer *A*, which has a 400-MHz clock. A computer designer is asked to build a computer *B* that will run this program in 6 seconds. The designer has determined that a substantial increase of the clock rate is possible, but this increase will affect the rest of the CPU, causing computer *B* to require 1.2 times as many clock cycles as computer *A* for this program. What clock rate is needed for computer *B*?

1.5.2. Suppose we have two implementations of the same instruction set architecture. Computer *A* has a clock cycle time of 4 ns and a *CPI* of 2.0 for some

program, and computer *B* has a clock cycle time of 2 ns and a *CPI* of 1.2 for the same program. Which computer is faster for this program?

1.5.3. A computer has three instruction classes, shown in Table 1.6.

Table 1.1. Instruction classes and *CPI*s for Problem 1.5.3.

Instruction class	CPI
A	1
B	2
C	3

For a particular high-level language statement, a compiler writer is considering two code sequences that require the instruction counts shown in Table 1.7.

Table 1.2. Code sequences and instruction counts for Problem 1.5.3.

Code sequence	Instruction counts for instruction class		
	A	B	C
S_1	2	1	2
S_2	4	1	1

Which code sequence is faster? What is the *CPI* for each sequence?

1.5.4. Consider the computer with three instruction classes and *CPI* measurements from Problem 1.5.3. Suppose we examine the code for the same program from two different compilers and obtain the data presented in Table 1.8.

Table 1.3. Code sequences generated by two compilers and their instruction counts for Problem 1.5.4.

Code from	Instruction counts for instruction class		
	A	B	C
Compiler 1	5,000,000	1,000,000	1,000,000
Compiler 2	10,000,000	1,000,000	1,000,000

Assume that the clock rate is 600 MHz. Which code sequence will execute faster according to MIPS and according to execution time?

1.5.5. Consider two implementations of a computer, one with and one without special floating-point hardware. A program *P* has the instruction frequencies shown in Table 1.9.
Computer CFP (Computer with Floating-Point) has floating-point hardware and can implement the floating-point operations directly. For each instruction class, this computer requires the number of clock cycles shown in Table 1.10.
Computer CNFP (Computer with No Floating-Point) has no floating-point

hardware and so must emulate the floating-point operations using integer instructions. The number of clock cycles needed to implement each of the floating-point operations is also shown in Table 1.10.

Table 1.4. Instruction frequencies for Problem 1.5.5.

Instruction	Frequency
Floating-point add	15%
Floating-point multiply	10%
Floating-point divide	5%
Integer instructions	70%

Table 1.5. Number of clock cycles needed for the instructions of CFP and CNFP computers for Problem 1.5.5.

Instruction	Number of clock cycles	
	CFP	CNFP
Floating-point add	4	40
Floating-point multiply	6	60
Floating-point divide	20	100
Integer instructions	2	2

Both computers have a clock rate of 600 MHz. Determine the native MIPS rating for both computers.

1.5.6. If the computer CFP in Problem 1.5.5 needs 300,000,000 instructions for program *P*, how many integer instructions does the computer CNFP require for the same program?

1.5.7. Assuming that each floating-point operation counts as 1, and that CFP executes 300,000,000 instructions, find the MFLOPS rating for both computers in Problem 1.5.5.

1.5.8. Table 1.11 shows the number of floating-point operations executed in two different programs and the runtime for those programs on three different computers *A*, *B*, and *C*.

Table 1.6. Number of floating-point operations and execution times of two programs for Problem 1.5.8.

Program	Floating-point operations	Execution time (in seconds)		
		A	B	C
P_1	10,000,000	1	10	20
P_2	100,000,000	1000	100	20

Which computer is the fastest according to total execution time?

1.5.9. Determine the MFLOPS rating for each program on each computer in Problem 1.5.8, assuming that each floating-point operation counts as 1.

1.5.10. Compare the performance of the three computers in Problem 1.5.8 using the geometric mean.

1.5.11. A computer is enhanced by adding a vector mode to it. When a computation is run in vector mode, it is 20 times faster than the normal mode of execution. The percentage of time that could be spent using vector mode is called *percentage of vectorization*. What percentage of vectorization is needed to achieve a speedup of 2?

1.5.12. We are considering two alternatives for implementing conditional branch instructions. In computer *A*, a condition code is set by a compare instruction and is followed by a branch that tests the condition code. In computer *B*, a compare is included in the branch instruction. On both computers, the conditional branch instruction takes 2 clock cycles, and all other instructions take 1 clock cycle. On computer *A*, 20% of all instructions executed are branch instructions. Since every branch needs a compare instruction, another 20% of the instructions are compares. Because computer *A* does not have the compare included in the branch, its clock cycle is 25% faster than computer *B*'s. Which computer is faster?

1.5.13. Consider a computer with a clock rate of 800 MHz. The measurements in Table 1.12 have been made using a simulator.

Table 1.7. Instruction frequency and clock cycles per instructions (*CPI*) for Problem 1.5.13.

Instruction class	Frequency	CPI
A	40%	2
B	25%	3
C	25%	3
D	10%	5

What is the MIPS rating for this computer?

1.5.14. Consider a computer with special floating-point hardware. A program has the following mix of operations and number of clock cycles for each instruction class, presented in Table 1.13.

Table 1.8. Mix of operations and clock cycles per instructions (*CPI*) for Problem 1.5.14.

Instruction class	Frequency	CPI
Floating-point multiply	10%	6
Floating-point add	15%	4
Floating-point divide	5%	20
Integer instructions	70%	2

The clock rate is 800 MHz. Assuming that each floating-point operation counts as 1, and the computer needs 300,000,000 instructions for this program, find the MFLOPS rating for the program.

1.5.15. Suppose we enhance a computer making all floating-point instructions run 5 times faster. If the execution time of some benchmark program before the floating-point enhancement is 10 seconds, what will the speedup be if 50% of the execution time is spent executing floating-point instructions?