King
CS 331
13 September 2020

Homework Assignment 1

Objectives

The purpose of this assignment is to:

- Evaluate the performance of a given processor,
- Understand how changes in software and hardware affect the performance, and evaluate potential performance enhancements,
- Identify performance misconceptions and pitfalls.

Guidelines

All question numbers refer to the exercises at the end of Chapter 1 of the textbook. Solutions for the following problems are to be completed independently. Type each answer and clearly show the work/steps.

Questions

Part 1

1.5: Consider three different processors P1, P2, and P3 executing the same instruction set. P1 has a 3 GHz clock rate and a CPI of 1.5. P2 has a 2.5 GHz clock rate and a CPI of 1.0. P3 has a 4.0 GHz clock rate and has a CPI of 2.2.

a. Which processor has the highest performance expressed in instructions per second?
b. If the processors each execute a program in 10 seconds, find the number of cycles and the number of instructions.
c. We are trying to reduce the execution time by 30% but this leads to an increase of 20% in the CPI. What clock rate should we have to get this time reduction?

```
P1: CR = 3.0E+9 cycles/second, CPI = 1.5 cycles/instruction.
P2: CR = 2.5E+9 cycles/second, CPI = 1.0 cycles/instruction.
P3: CR = 4.0E+9 cycles/second, CPI = 2.2 cycles/instruction.

   a. CPI_P1 = 1.5 cycles/instruction
      CR_P1 = 3E+9 cycles/second

      Use performance = (1 / CPI) * (CR).

      performance_P1 = (1 / CPI_P1) * (CR_P1)
```

```
performance_P1 = (1 instruction / 1.5 cycles) * (3E+9 cycles / 1 second)
performance_P1 = (1 ins * 3E+9 cycs) / (1.5 cycs / 1 sec)
performance_P1 = (3E+9 ins) / (1.5 sec)
performance_P1 = 2.0E+9 ins/sec

CPI_P2 = 1.0 cycles/instruction
CR_P2 = 2.5E+9 cycles/second
performance_P2 = (1 / CPI_P2) * (CR_P2)
performance_P2 = (1 instruction / 1.0 cycles) * (2.5E+9 cycles / 1 second)
performance_P2 = (1 ins * 2.5E+9 cycs) / (1.0 cycs * 1 sec)
performance_P2 = (2.5E+9 ins) / (1.0 sec)
performance_P2 = 2.5E+9 inst/sec

CPI_P3 = 2.2 cycles/instruction
CR_P3 = 4.0E+9 cycles/second
performance_P3 = (1 / CPI_P3) * (CR_P3)
performance_P3 = (1 instruction / 2.2 cycles) * (4.0E+9 cycles / 1 second)
performance_P3 = (1 ins * 4.0E+9 cycles) / (2.2 cycs * 1 sec)
performance_P3 = (4.0E+9 ins) / (2.2 seco)
performance_P3 = 1.8E+9 ins/sec
```

Processor P2 has the highest performance at 2.0E+9 instructions per second.

b. CPU_time = 10 seconds. Given CPU_time, CR, CPI. Find CC and IC.
Use CPU_time = CC / CR, CC = IC * CPI

```
CPU_time_P1 = CC_P1 / CR_P1
10 seconds = CC_P1 / (3E+9 cycs/sec)
3.0E+10 cycles = CC_P1

CC_P1 = IC_P1 * CPI_P1
3.0E+10 cycs = IC_P1 * (1.5 cycs/ins)
2.0E+10 instructions = IC_P1

CPU_time_P2 = CC_P2 / CR_P2
10 sec = CC_P2 / (2.5E+9 cycs/sec)
2.5E+10 cycles = CC_P2

CC_P2 = IC_P2 * CPI_P2
2.5E+10 cycs = IC_P2 * (1.0 cycs/ins)
2.5E+10 instructions = IC_P2

CPU_time_P3 = CC_P3 / CR_P3
10 seconds = CC_P3 / (4.0E+9 cycs/sec)
4.0E+10 cycles = CC_P3

CC_P3 = IC_P3 * CPI_P3
4.0E+10 cycles = IC_P3 * (2.2 cycs/ins)
```

```
    1.8E+10 instructions = IC_P3

    Processor P1 has 3.0E+10 cycles and 2.0E+10 instructions, P2 has 2.5E+10
    cycles and 2.5E+10 instructions, and P3 has 4.0E+10 cycles and 1.8E+10
    instructions for a program executed in 10 seconds.

  c. CPU_time *= 0.70 and CPI *= 1.2. Find new CR.

    Use CPU_time = CC / CR, CC = IC * CPI
    CR = CC / CPU_time
    CR = (IC * CPI) / (CPU_time)

    CR_old = (IC * CPI_old) / (CPU_time_old)
    CR_new = (IC * CPI_new(1.2)) / (CPU_time_new(0.70))

    CR_old = CR_new

    ???
```

Part 2: Answer which implementation is faster and pay attention that P1 and P2 have different clock rates.

1.6: Consider two different implementations of the same instruction set architecture. The instructions can be divided into four classes according to their CPI (class A, B, C, and D). P1 with a clock rate of 2.5 GHz and CPIs of 1, 2, 3, and 3, and P2 with a clock rate of 3 GHz and CPIs of 2, 2, 2, and 2. Given a program with a dynamic instruction count of 1.0E6 instructions divided into classes as follows: 10% class A, 20% class B, 50% class C, and 20% class D, which implementation is faster?

a. What is the global CPI for each implementation?
b. Find the clock cycles required in both cases.

```
P1: CR = 2.5E+9 cycles/second, CPI = 1,2,3,3 cycles/instruction.
P2: CR = 3.0E+9 cycles/second, CPI = 2,2,2,2 cycles/instruction.

Program: IC = 1.0E+6 instructions, where
IC_A = IC(0.10) = 1.0E+6(0.10) = 1.0E+5 instructions,
IC_B = IC(0.20) = 1.0E+6(0.20) = 2.0E+5 instructions,
IC_C = IC(0.50) = 1.0E+6(0.50) = 5.0E+5 instructions,
IC_D = IC(0.20) = 1.0E+6(0.20) = 2.0E+5 instructions.

Given CR, IC, CPI. Find CPU_time.
Use CC_total = summation(IC_i * CPI_i), CPU_time = CC / CR.

CC_P1 = (1.0E+5 ins * 1.0 cycs/in) + (2.0E+5 ins * 2.0 cycs/in) + (5.0E+5 ins *
```

```
3.0 cycs/in) + (2.0E+5 ins * 3.0 cycs/in) = 2.6E+6 cycles
CPU_time_P1 = (2.6E+6 cycs) / (2.5E+9 cycs/second) = 1.04E-3 seconds

CC_P2 = (1.0E+5 ins * 2.0 cycs/in) + (2.0E+5 ins * 2.0 cycs/in) + (5.0E+5 ins *
2.0 cycs/in) + (2.0E+5 ins * 2.0 cycs/in) = 2.0E+6 cycles
CPU_time_P2 = (2.0E+6 cycs) / (3.0E+9 cycs/second) = 6.67E-4 seconds

Since 0.67 ms < 1.04 ms, the implementation for P2 is faster than that of P1.

   a. Use CPI_total = summation[CPI_i * (IC_i / IC_total)

      CPI_total_P1 = [1.0 cycs/ins * (1.0E+5 ins / 1.0E+6 ins)]
      + [2.0 cycs/ins * (2.0E+5 ins / 1.0E+6 ins)]
      + [3.0 cycs/ins * (5.0E+5 ins / 1.0E+6 ins)]
      + [3.0 cycs/ins * (2.0E+5 ins / 1.0E+6 ins)] = 2.6 cycles/instruction

      CPI_total_P2 = [1.0 cycs/ins * (2.0E+5 ins / 1.0E+6 ins)]
      + [2.0 cycs/ins * (2.0E+5 ins / 1.0E+6 ins)]
      + [2.0 cycs/ins * (5.0E+5 ins / 1.0E+6 ins)]
      + [2.0 cycs/ins * (2.0E+5 ins / 1.0E+6 ins)] = 2.0 cycles/instruction

      Implementation P1 has a global CPI of 2.6 cycles/instruction, and P2 a
      global CPI of 2.0 cycles/instruction.

   b. Before realizing CC was already found before part A,
      use CPI_total = CC / IC_total,
      CC = CPI_total * IC_total

      CC_P1 = (2.6 cycles/instruction) * (1.0E+6 instructions) = 2.6E+6 cycles

      CC_P2 = (2.0 cycles/instruction) * (1.0E+6 instructions) = 2.0E+6 cycles

      Implementation P1 requires 2.6E+6 cycles to execute the program, and P2
      requires 2.0E+6 cycles.
```

<u>Part 3</u>

1.7: Compilers can have a profound impact on the performance of an application. Assume that for a program, compiler A results in a dynamic instruction count of 1.0E9 and has an execution time of 1.1 s, while compiler B results in a dynamic instruction count of 1.2E9 and an execution time of 1.5 s.

a. Find the average CPI for each program given that the processor has a clock cycle time of 1 ns.
b. A new compiler is developed that uses only 6.0E8 instructions and has an average CPI of 1.1. What is the speedup of using this new compiler versus using compiler A or B on the original processor?

c. A new compiler is developed that uses only 6.0E8 instructions and has an average CPI of 1.1. What is the speedup of using this new compiler versus using compiler A or B on the original processor?

```
A: IC = 1.0E+9 instructions, CPU_time = 1.0 seconds
B: IC = 1.2E+9 instructions, CPU_time = 1.5 seconds

   a. Given IC, CPU_time, and CT = 1.0E-9 seconds/cycle. Find CPI.
      Use CPU_time = IC * CPI * CT
      (CPU_time) / (IC * CT) = CPI

      (1.0 seconds) / (1.0E+9 instructions * 1.0E-9 seconds/cycle) = CPI_A
      1.0 cycle/instruction = CPI_A

      (1.5 seconds) / (1.2E+9 instructions * 1.0E-9 seconds/cycle) = CPI_B
      1.25 cycles/instruction = CPI_B

      The average CPI for a program using compiler A is 1.0 cycle/instruction,
      and 1.25 cycles/instruction using compiler B.

   b. Given IC = 6.0E+8 instructions, CPI = 1.0 cycles/instruction. Find
      CPU_time multiplier.

      Use CC = IC * CPI
      CC_new = (6.0E+8 ins) * (1.0 cycs/ins) = 6.0E+8 cycles

      Given IC, CPI, CC. Find CPU_time.

      ???

   c. IC = 6.0E+8 instructions, CPI = 1.1 cycles/instruction. Find CPU_time.

      Use CPU_time = IC * CPI * CT

      CPU_time = (6.0E+8 ins) * (1.1 cycs/ins) * (CT)

      ???
```

Part 4

Section 1.10 cites as a pitfall the utilization of a subset of the performance equation as a performance metric. To illustrate this, consider the following two processors. P1 has a clock rate of 4 GHz, average CPI of 0.9, and requires the execution of 5.0E9 instructions. P2 has a clock rate of 3 GHz, an average CPI of 0.75, and requires the execution of 1.0E9 instructions.

1.12.1: One usual fallacy is to consider the computer with the largest clock rate as having the largest performance. Check if this is true for P1 and P2.

```
P1: CR = 4.0E+9 cycs/sec, CPI = 0.90 cycs/ins, IC = 5.0E+9 ins
P2: CR = 3.0E+9 cycs/sec, CPI = 0.75 cycs/ins, IC = 1.0E+9 ins

Given CR, CPI, IC. Find CPU_time.

Use CPU_time = (IC * CPI) / (CR)

CPU_time_P1 = (5.0E+9 ins * 0.9 cycs/ins) / (4.0E+9 cycs/sec) =  1.13 sec

CPU_time_P2 = (1.0E+9 ins * 0.75 cycs/ins) / (3.0E+9 cycs/sec) = 0.25 sec

CR_P1 > CR_P2
CPU_time_P1 > CPU_time_2

With the CPU time of processor P1 (with a clock rate of 4 GHz) at 1.13 seconds
and P2 (with a clock rate of 3 GHz) at 0.25 seconds, the fallacy is proven to be
false.
```

1.12.2: Another fallacy is to consider that the processor executing the largest number of instructions will need a larger CPU time. Considering that processor P1 is executing a sequence of 1.0E9 instructions and that the CPI of processors P1 and P2 do not change, determine the number of instructions that P2 can execute in the same time that P1 needs to execute 1.0E9 instructions.

```
P1: CR = 4.0E+9 cycs/sec, CPI - 0.90 cycs/ins, IC = 1.0E+9 ins
P2: CR = 3.0E+9 cycs/sec, CPI = 0.75 cycs/ins

Find CPU_time_P1 and IC_P2.

Use CC = IC * CPI, CPU_time = CC / CR

CC_P1 = (1.0E+9 ins) * (0.9 cycs/ins) = 9.0E+8 cycles
CPU_time_P1 = (9.0E+8 cycs) / (4.0E+9 cycs/second) = 0.23 seconds

Use CPU_time = (IC * CPI) / (CR)

0.23 sec = (IC_P2 * 0.75 cycs/ins) / (3.0E+9 cycs/sec)
9.2E+8 instructions = IC_P2

CPU_time_P1 == CPU_time_P2
IC_P1 > IC_P2

At the same CPU time of 0.23 seconds, processor P1 executes 1.0E+9 instructions
and P2 executes 9.2E+8 instructions. In other words, P1 executes more
```

instructions than P2 in the same amount of time.

1.12.3: A common fallacy is to use MIPS (millions of instructions per second) to compare the performance of two different processors, and consider that the processor with the largest MIPS has the largest performance. Check if this is true for P1 and P2.

```
P1: CR = 4.0E+9 cycs/sec, CPI = 0.90 cycs/ins, IC = 5.0E+9 ins
P2: CR = 3.0E+9 cycs/sec, CPI = 0.75 cycs/ins, IC = 1.0E+9 ins

Find MIPS. Use MIPS = (CR) / (CPI * 1E+6)

MIPS_P1 = (4.0E+9 cycs/sec) / (0.9E+6 cycs/ins) = 4.4E+3 ins/sec

MIPS_P2 = (3.0E+9 cycs/sec) / (0.75E+6 cycs/ins) = 4.0E+3 ins/sec

Find CPU_time. Use MIPS = (IC) / (CPU_time * 1E+6)

(4.4E+3 ins/sec) = (5.0E+9 ins) / (CPU_time_P1 * 1E+6)
(4.4E+3 ins/sec) * (CPU_time_P1 * 1E+6) = (5.0E+9 ins)
CPU_time_P1 = (5.0E+9 ins) / (4.4E+3 ins/sec * 1E+6) = 1.14 seconds

(4.0E+3 ins/sec) = (1.0E+9 ins) / (CPU_time_P2 * 1E+6)
(4.0E+3 ins/sec) * (CPU_time_P2 * 1E+6) = (1.0E+9 ins)
CPU_time_P2 = (1.0E+9 ins) / (4.0E+3 ins/sec * 1E+6) = 0.25 seconds

MIPS_P1 > MIPS_P2
CPU_time_P1 > CPU_time_P2

With the CPU time of processor P1 (with a MIPS of 4.4E+3 instructions/second) of
1.14 seconds and 0.25 seconds of P2 (with a MIPS of 4.0E+3 instructions/second),
the fallacy is proven to be false. In other words, the processor with the
largest MIPS does not necessarily mean that it has the largest performance.
```

Part 5: The program includes a mix of F/P, L/S, branch and INT instructions. The times spent executing FP, L/S, branch instructions are given so the time spent executing INT instructions is the time remaining out of the total program run time.

Another pitfall cited in Section 1.10 is expecting to improve the overall performance of a computer by improving only one aspect of the computer. Consider a computer running a program that requires 250 s, with 70 s spent executing FP instructions, 85 s executed L/S instructions, and 40 s spent executing branch instructions.

1.13.1: By how much is the total time reduced if the time for FP operations is reduced by 20%?

```
CPU_time_total = 250 seconds
CPU_time_FP = 70 seconds
CPU_time_LS = 85 seconds
CPU_time_branch = 40 seconds
CPU_time_INT = 55 seconds

CPU_time_new = CPU_time_FP(0.80) + CPU_time_LS + CPU_time_branch + CPU_time_INT
CPU_time_new = (70 sec * 0.80) + (85 sec) + (40 sec) + (55 sec) = 236 seconds

By reducing the time for FP by 20%, the total time is reduced from 250 seconds
to 236 seconds (a 5.6% decrease).
```

1.13.2: By how much is the time for INT operations reduced if the total time is reduced by 20%?

```
CPU_time_total = 250 seconds
CPU_time_FP = 70 seconds
CPU_time_LS = 85 seconds
CPU_time_branch = 40 seconds
CPU_time_INT = 55 seconds

CPU_time_total(0.80) = CPU_time_FP + CPU_time_LS + CPU_time_branch +
CPU_time_INT

250(0.80) seconds = (70 seconds) + (85 seconds) + (40 seconds) + (CPU_time_INT)
5 seconds = CPU_time_INT

% decrease = |(55 seconds - 5 seconds) / (55 seconds)| * 100 = 90.9%

By reducing the total time by 20%, the time for INT operations is reduced by 50
seconds, or 91%.
```

1.13.3: Can the total time can be reduced by 20% by reducing only the time for branch instructions?

```
CPU_time_total = 250 seconds
CPU_time_FP = 70 seconds
CPU_time_LS = 85 seconds
CPU_time_branch = 40 seconds
CPU_time_INT = 55 seconds

CPU_time_after = (CPU_time_affected / improvement_amount) + CPU_time_unaffected
(CPU_time_after - CPU_time_unaffected)(improvement_amount) = (CPU_time_affected)
```

```
improvement_amount = CPU_time_afffected / (CPU_time_after - CPU_time_unaffected)

improvement_amount = (40 seconds) / (250(0.80) seconds - (250 - 40 seconds)
improvement_amount = (40 seconds) / (-90 seconds) = -0.44

With a negative improvement multiplier of -0.44, the total time cannot be
reduced by 20% by reducing the time for branch instructions alone.
```