Jordan Winkler
computer architecture
Tue Jan 22 01:37:32 EST 2019
hw2

1) Do Exercise 1.5 on page 55 of the textbook.

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?

instructions/second = clock rate/cpi = clock cycle/second * instruction/clock cycle

P1: $3 \cdot 10^9 / 1.5 = 2 \cdot 10^9$
P2: $2.5 \cdot 10^9 / 1 = 2.5 \cdot 10^9$
P3: $4 \cdot 10^9 / 2.2 \approx 1.82 \cdot 10^9$

So P2 at $2.5 \cdot 10^9$ instructions per second.

b. If the processors each execute a program in 10 seconds, find the number of cycles and the number of instructions.

P1: $2 \cdot 10^{10}$ instructions
P2: $2.5 \cdot 10^{10}$ instructions
P3: $1.82 \cdot 10^{10}$ 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?

$time = \frac{seconds}{program} = \frac{instructions}{program} \cdot \frac{clockcycles}{instruction} \cdot \frac{seconds}{clockcycle}$
$0.7 \cdot time = 0.7 \cdot \frac{seconds}{program} = \frac{instructions}{program} \cdot 1.2(\frac{clockcycles}{instruction}) \cdot x(\frac{seconds}{clockcycle})$
$0.7 = 1.2x$
$x = 7/12$

So the clock rate should be 12/7 or about 71.4 percent faster.

2) Do Exercise 1.6 on page 55 of the textbook.

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?

P1: $1 \cdot 0.1 + 2 \cdot 0.2 + 3 \cdot 0.5 + 3 \cdot 0.2 = 2.6$ CPI
P2: $2 \cdot 0.1 + 2 \cdot 0.2 + 2 \cdot 0.5 + 3 \cdot 0.2 = 2.2$ CPI

b. Find the clock cycles required in both cases.

P1: $2.6 \cdot 10^6$ clock cycles
P2: $2.2 \cdot 10^6$ clock cycles

3) Do Exercise 1.12 (all parts: 1.12.1 - 1.12.4) on page 58 of the textbook.

1.12 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.

cpu time = instruction count * cpi / clock rate

P1: $5 \cdot 10^9 \cdot 0.9/(4 \cdot 10^9) = 1.125s$
P2: $10^9 \cdot 0.75/(3 \cdot 10^9) = 0.25s$

So P2 is prefered since it takes less cpu time.

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.

newP1: $10^9 \cdot 0.9/(4 \cdot 10^9) = 0.225s$

P2 executions $= 0.225 \cdot 3 \cdot 10^9/0.75 = 0.9 \cdot 10^9 = 9 \cdot 10^8$

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: $\frac{4 \cdot 10^9}{0.9} = 4444$ mips
P2: $\frac{3 \cdot 10^9}{0.75} = 4000$ mips

But as seen in 12.1, P2 takes less CPU time even though it has lower mips.

1.12.4 Another common performance figure is MFLOPS (millions of floating-point operations per second), defined as

MFLOPS = No. FP operations / (execution time x 1E6)

but this figure has the same problems as MIPS. Assume that 40% of the instructions executed on both P1 and P2 are floating-point instructions. Find the MFLOPS figures for the programs.

P1: $0.4(5 \cdot 10^3)/1.125 = 1777.777$ mflops
P2: $0.4(10^3)/.25 = 1600.0$ mflops

4) Do Exercise 1.13 (all parts: 1.13.1 - 1.13.3) on pages 58-59 of the textbook. 1.13 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%?

$70 \cdot 0.8 + 85 + 40 + 55 = 56 + 85 + 40 + 55 = 236s$

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

If INT is the remaining 55s then INT will be reduced to 44s.

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

$(250 - 40)/250 = 0.84 > 0.8$. So no.

5) Give the twos complement 16-bit representation of the following decimal

numbers: a) 45

$45/2 = 22$ R 1
$22/2 = 11$ R 0
$11/2 = 5$ R 1
$5/2 = 2$ R 1
$2/2 = 1$ R 0
$1/2 = 0$ R 1

0000000000101101_2
b) -23

$23/2 = 11$ R 1
$11/2 = 5$ R 1
$5/2 = 2$ R 1
$2/2 = 1$ R 0
$1/2 = 0$ R 1

$1000000000010110_{2-complement}$

c) -184

$184/2 = 92$ R 0
$92/2 = 46$ R 0
$46/2 = 23$ R 0
$23/2 = 11$ R 1
$11/2 = 5$ R 1
$5/2 = 2$ R 1
$2/2 = 1$ R 0
$1/2 = 0$ R 1

$1000000010110111_{2-complement}$

6) Do Exercise 2.1 on page 164 of the textbook.

2.1 For the following C statement, what is the corresponding MIPS assembly code? Assume that the variables f, g, h, and i are given and could be considered 32-bit integers as declared in a C program. Use a minimal number of MIPS assembly instructions.

f = g + (h - 5);

```
# c-code: f = g + (h - 5);
# let $r1 = 5, $r2 = g
```

```
addi $r1, $r1, -5
add $s1, $r2, $r1

# use $s1 for f
```