COSE222 Homwork #1 ... 2013210111 남세현

#1.2 The Eight great ideas in computer architecture are similar to ideas from other fields. Match the eight ideas from computer architecture, "Design for Moore's Law", "Use Abstraction to Simplify Design", "Make the Common Case Fast", "Performance via Parallelism", "Performance via Pipelining", "Performance via Prediction", "Hierarchy of Memories", and "Dependability via Redundancy" to the following ideas from other fields:

a. Assembly lines in automobile manufacturing

b. Suspension bridge cables

c. Aircraft and marine navigation systems that incorporate wind information

d. Express elevators in buildings

e. Library reserve desk

f. Increasing the gate area on a CMOS transistor to decrease its switching time

g. Adding electromagnetic aircraft catapults (which are electrically-powered as opposed to current steam-powered models), allowed by the increased power generation offered by the new reactor technology

h. Building self-driving cars whose control systems partically rely on existing sensor systems already installed into the base vehicle, such as lane departure systems and smart cruise control systems

Answer : a - Performance via Pipelining.

b - Performance via Pararellelism.

c - Performance via prediction.

d - Dependability via redundancy.

e - Hierarchy of Memories.

f - Make the Common Case Fast.

h - Use Abstraction to simplify Design.

#1.6 Consider two different implementations of the same instruction set architecture. The instructions can be divided into four classes accroding to their CPI (calss A, B, C, and D). P1 with a clock rate of 2.5 GHz and CPIs of 1, 2, 3, and 3, and P2 witch 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.

Answer : a - P1's global CPI : (1 * 10% + 2 * 20% + 3 * 50% + 3 * 20%) = 0.1 + 0.4 + 1.5 + 0.6 = 2.6 , P2's global CPI : ... = 2.

b - P1 : 2.6 * 1.0E6 = 2.6E6, P2 : ... = 2.0E6.

#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. Assume the compiled programs run on two different processors. If the execution times on the two processors are the same, how much faster is the clock of the processor running compiler A's code versus the clock of the processor running compiler B's code?

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?

Answer : a - Compiler A's average CPI = { 1.1 sec / 1.0E-9 (sec / clock) } / 1.0E9 instruction = 1.1E9 clock / 1.0E9 instruction = 1.1 CPI.
Compiler B's average CPI = { 1.5 sec / 1.0E-9 (sec / clock) } / 1.2E9 instruction
= 1.52E9 clock / 1.2E9 = 1.267 CPI.

b - Assume that execution time is x second. this former processor executes 1.0E9 instructions in x second, and the latter executes 1.2E9 instructions in x second.
that mean the latter faster 1.2 times than formmer.

c - Assume that the processor ahs a clock cyle time of 1 ns. Let new compiled program's execution time t sec, then
{ t sec / 1.0E-9 (sec/clock) } / 6.0E8 = 1.1 CPI
=> t * 1.0E9 = 6.6E8
=> t = 0.66 second.
even new compiler's CPI is same with compier A's, the execution time is more faster about 1.67 times. you know, B is more slower then A.

(except 1.11.6)

#1.11 The result of the SPEC CPU2006 bzip2 benchmark running on an AMD Barcelona has an instruction count of 2.389E12, an execution time of 750 s, and a reference time of 9650 s.

#1.11.1 Find the CPI if the clock cycle time is 0.333 ns.

Answer : CPI = { 7.5E2 sec / 3.33E-10 (sec / clock) } / 2.389E12 instruction

= 2.252E-12 clock / 2.389E12 instruction = 9.43E-1. (CPI = 0.943)

#1.11.2 Find the SPECratio.

Answer : 9650 / 750 = 12.87.

#1.11.3 Find the increase in CPU time if the number of instructions of the benchmark is increased by 10% without affecting the CPI.

Answer : CPU Time = Instruction Counts * CPI * clock cycle time

=> Increase of CPU Time = 10% * 2.389E12 * 9.43E-1 * 3.33E-10

= 7.502E1 sec (75.02 second)

#1.11.4 Find the increase in CPU time if the number of instructions of the benchmark is increased by 10% and the CPI is increased by 5%.

Answer : Increased CPU Time = 110% * Instruction Counts * 105% * CPI * CCT

= 8.66E2 sec. So, increase of CPU Time = 866 - 750 = 116 second.

#1.11.5 Find the change in the SPECratio for this change.

Answer : 9650 / 866 = 11.14... 12.87 => 11.14 (13% decreased)

#1.11.7 This CPI value is larger than obtained in 1.11.1 as the clock rate was increased from 3 GHz to 4 GHz. Determine whether the increase in the CPI is similar to that of the clock rate. If they are dissimilar, why?

Answer : CPU Time = Instruction Count * CPI / clock rate.
According to this equation, the increase in the CPI makes CPU Time's longer, increase of the clock rate make CPU Time shorter. They're dissimilar.

#1.11.8 By how much has the CPU time been reduced?

Answer : In 3 GHz, according to #1.11.4's answer, CPU Time is 866 sec.
with this equation, "CPU Time = I * CPI / Clock Rate", CPU Time is inversely proportional to clock rate. So, the reduced CPU Time = 866 * (3GHz/4GHz) = 649.5 sec.
216.5 sec reduced.

#1.11.9 For a second benchmark, libquantum, assume an execution time of 960 ns, CPI of 1.61, and clock rate of 3 GHz. If the execution time is reduced by an additional 10% without affecting to the CPI and with a clock rate of 4 GHz, determine the number of instructions.

Answer : 9.6E-7 = I * 1.61 / 3E9 + other IO Time ... (1)

8.64E-7 = I * 1.61 / 4E9 + other IO Time ... (2)

(1) - (2) => 9.6E-8 = I * 1.61 * (1/3E9 - 1/4E9).

I = 9.6E-8 / { 1.61 * (1/3E9 - 1/4E9) } = 7.15E2 (715)

#1.11.10 Determine the clock rate required to give a further 10% reduction in CPU time while maintaining the number of instructions and with the CPI unchanged.

Answer : According to the equation, " CPU Time = I * CPI / Clock Rate", to make CPU time 10% shorter, the clock rate should be higher than 11% (11.111111...%, (1/0.9)%).

#1.11.11 Determine the clock rate if the CPI is reduced by 15% and the CPU time by 20% while the number of instructions is unchanged.

Answer : CPU Time = I * CPI / Clock Rate
=> 80% CPU Time = I * 85% * CPI / N% Clock Rate.
N is equal to 106.25%. (N% = 85%/80%)

(except 1.12.4)

#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.

Answer : Assume better performance processor has lesser CPU Time,
then P1's CPU Time = 5.0E9 * 0.9 / 4E9 = 1.125 sec.
P2's CPU Time = 1.0E9 * 0.75 / 3E9 = 0.25 sec.
P1's clock rate (4GHz) >P2's (3GHz), P1's CPU Time (1.125 sec) > P2's(0.25 sec). 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.

Answer : First, the time of P1 need is equal to 1.0E9 * 0.9 / 4.0E9 = 0.225 sec.
According the equation, "Instructions count = CPUTime / CPI * Clock Rate",
the number of instructions that P2's could execute in 0.225 sec is equal to …
0.225 / 0.75 * 3.0E9 = 9.0E8.

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

Answer : P1's MIPS = 5.0E9 / (1.125 * 1.0E6) = 4.44E3 (4444.444….)
P2's MIPS = 1.0E9 / (0.25 * 1.0E6) = 4.0E3 (4000). But P2's CPU Time is lesser then P1's.
So, False.