Chapter Exam

Chapter 1-Computer Abstractions and Technology

2012/03/27

1. Consider two different implementations of the same instruction set architecture. There are four classes of instructions, A, B, C, and D. The clock rate and CPI of each implementation are given in the following table.

	Clock rate	CPI Class A	CPI Class B	CPI Class C	CPI Class D
P1	1.4 GHz	1	2	3	4
P2	2 GHz	2	2	2	2

(1) Given a program with 10^6 instructions divided into classes as follows: 10% class A, 20% class B, 50% class C and 20% class D, which implementation is faster? **(10 Points)**

(2) Which is the global CPI for each implementation? **(10 Points)**

2. The following table shows results for SPEC2006 benchmark programs running on an AMD Bracelona.

Name	Intr. Count x10^9	Execution time (seconds)	Reference time (seconds)
perl	2118	500	9770

(1) Find the CPI if the clock cycle time is 0.333 ns. **(10 Points)**

(2) Find the SPEC ration. **(10 Points)**

3. Consider two different processor P1 and P2 executing the same instruction set with the clock rates and CPIs given in the following table.

Processor	Clock Rate	CPI
P1	1.5 GHz	1.0
P2	3 GHz	2.5

(1) Which processor has the highest performance? **(10 Points)**

(2) If the processors each execute a program in 10 seconds, find the number of cycles and the number of instructions. **(10 Points)**

(3) We are trying to reduce the time by 10% but this leads to an increase of 20% in the CPI. What clock rate (for each processor) should we have to get this time reduction? **(10 Points)**

4. Consider a multi-cycle CPU that can run at two distinct clock-speeds: 1 GHz in “low power” mode, and 2.5 GHz in “full power” mode.

(1) We want to estimate the *ratio* of the running time of a test program when run in low power mode versus full power mode. Your friend claims this is *impossible* without knowing the details of the test program (number of instructions, types of instructions, etc.). Explain why your friend is correct, or explain why your friend is wrong. (**15 Points**)

(2) A test program runs in time *t* in full power mode. When the code is analyzed by a profiler, we find that 45% of the time is spent in function **findRoot**. By improving a key data-structure used only in this function, **findRoot** runs twice as fast in full power mode. How much faster would the rest of the code have to be made so that the program can run in time *t* in low power mode? (**15 Points**)