CSCE 513 Computer Architecture, Fall 2018

Assignment #1, ~~due 09/17/2018, 11:55PM,~~ due 09/17/2018, 11:55PM

Covered topics: 1) Quantitative analysis of power/energy and CPU performance; 2) Amdahl's Law and Effective CPI, 3) Measuring CPI using PAPI

Questions	COD 1.5	COD 1.6	COD 1.8	COD 1.14	CAQA 1.12	CAQA 1.14	CAQA A3	1.15
Points	15	10	10	20	15	15	15	15

1.15 (15 points) is required for graduate students. Undergraduates who answer receive bonus point.

Submission:

1. **Only electronic submissions on dropbox are accepted.**
2. **All your solutions should be included in a SINGLE PDF file.**
3. **Number your solutions in the same way and in the same order as the questions are numbered in this document and do NOT include the questions as part of your solutions.**
4. **Include your full name in the PDF file.**
5. **Scanned copy of handwritten answers will NOT be graded.**

1) Quantitative analysis of power/energy and CPU performance; (questions are from COD textbook)

1.5 [4] <§1.6> 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?

1.6 [20] <§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.

1.8 The Pentium 4 Prescott processor, released in 2004, had a clock rate of 3.6 GHz and voltage of 1.25 V. Assume that, on average, it consumed 10 W of static power and 90 W of dynamic power.

The Core i5 Ivy Bridge, released in 2012, had a clock rate of 3.4 GHz and voltage of 0.9 V. Assume that, on average, it consumed 30 W of static power and 40 W of dynamic power.

1.8.1 [5] <§1.7> For each processor find the average capacitive loads.

1.8.2 [5] <§1.7> Find the percentage of the total dissipated power comprised by static power and the ratio of static power to dynamic power for each technology.

1.8.3 [15] <§1.7> If the total dissipated power is to be reduced by 10%, how much should the voltage be reduced to maintain the same leakage current? Note: power is defined as the product of voltage and current.

1.14 Assume a program requires the execution of 50 × 106 FP instructions, 110 × 106 INT instructions, 80 × 106 L/S instructions, and 16 × 106 branch instructions. The CPI for each type of instruction is 1, 1, 4, and 2, respectively. Assume that the processor has a 2 GHz clock rate.

1.14.1 [10] <§1.10> By how much must we improve the CPI of FP instructions if we want the program to run two times faster?

1.14.2 [10] <§1.10> By how much must we improve the CPI of L/S instructions if we want the program to run two times faster?

1.14.3 [5] <§1.10> By how much is the execution time of the program improved if the CPI of INT and FP instructions is reduced by 40% and the CPI of L/S and Branch is reduced by 30%?

2) Amdahl's Law and Effective CPI

CAQA 1.12: a, b, c

1.12 [20/10/10/10/15] <1.9> In this exercise, assume that we are considering enhancing a quad-core machine by adding encryption hardware to it. When computing encryption operations, it is 20 times faster than the normal mode of execution. We will define percentage of encryption as the percentage of time in the original execution that is spent performing encryption operations. The specialized hardware increases power consumption by 2%.

a. [20] <1.9> Draw a graph that plots the speedup as a percentage of the computation spent performing encryption. Label the y-axis "Net speedup" and label the x-axis "Percent encryption."

b. [10] <1.9> With what percentage of encryption will adding encryption hardware result in a speedup of 2?

c. [10] <1.9> What percentage of time in the new execution will be spent on encryption operations if a speedup of 2 is achieved?

CAQA 1.14: a, b, c

1.14 [20/20/15] <1.9> When making changes to optimize part of a processor, it is often the case that speeding up one type of instruction comes at the cost of slowing down something else. For example, if we put in a complicated fast floating-point unit, that takes space, and something might have to be moved farther away from the middle to accommodate it, adding an extra cycle in delay to reach that unit. The basic Amdahl's Law equation does not take into account this trade-off.

a. [20] <1.9> If the new fast floating-point unit speeds up floating-point operations by, on average, 2x, and floating-point operations take 20% of the original program's execution time, what is the overall speedup (ignoring the penalty to any other instructions)?

b. [20] <1.9> Now assume that speeding up the floating-point unit slowed down data cache accesses, resulting in a 1.5x slowdown (or 2/3 speedup). Data cache accesses consume 10% of the execution time. What is the overall speedup now?

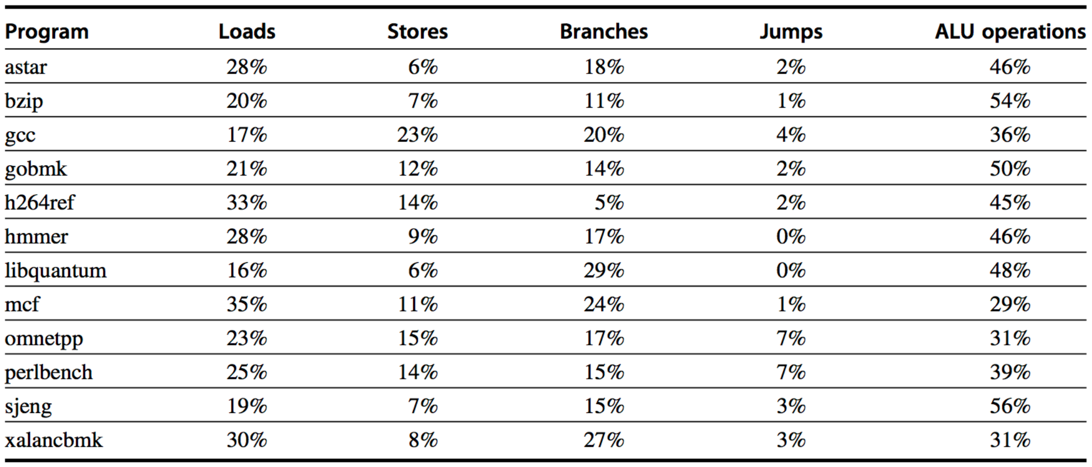
c. [15] <1.9> After implementing the new floating-point operations, what percentage of execution time is spent on floating-point operations? What percentage is spent on data cache accesses?

CAQA A.3, assume 60% of branch instructions are taken.

A.3 [10] < A.9 > Compute the effective CPI for an implementation of a RISC-V CPU using Figure A.29. Assume we have made the following measurements of average CPI for each of the instruction types:

Instruction	Clock cycles
All ALU operations	1.0
Loads	3.5
Stores	2.8
Branches	
Taken	4.0
Not taken	2.0
Jumps	2.4

Average the instruction frequencies of gobmk and mcf to obtain the instruction mix. You may assume that all other instructions (for instructions not accounted for by the types in Table A.29) require 3.0 clock cycles each.



Program	Loads	Stores	Branches	Jumps	ALU operations
astar	28%	6%	18%	2%	46%
bzip	20%	7%	11%	1%	54%
gcc	17%	23%	20%	4%	36%
gobmk	21%	12%	14%	2%	50%
h264ref	33%	14%	5%	2%	45%
hmmer	28%	9%	17%	0%	46%
libquantum	16%	6%	29%	0%	48%
mcf	35%	11%	24%	1%	29%
omnetpp	23%	15%	17%	7%	31%
perlbench	25%	14%	15%	7%	39%
sjeng	19%	7%	15%	3%	56%
xalancbmk	30%	8%	27%	3%	31%

Figure A.29 RISC-V dynamic instruction mix for the SPECint2006 programs. Omnetpp includes 7% of the instructions that are floating point loads, stores, operations, or compares; no other program includes even 1% of other instruction types. A change in gcc in SPECint2006, creates an anomaly in behavior. Typical integer programs have load frequencies that are 1/5 to 3x the store frequency. In gcc, the store frequency is actually higher than the load frequency! This arises because a large fraction of the execution time is spent in a loop that clears memory by storing x0 (not where a compiler like gcc would usually spend most of its execution time!). A store instruction that stores a register pair, which some other RISC ISAs have included, would address this issue.

3) Measuring CPI using PAPI

1.15 The sum_full.c function in the class website (https://passlab.github.io/CSCE513/exercises/sum/) currently measures the execution time of sum using timer and in this assignment, you will add PAPI calls in the sum_full.c program to collect total instruction count and total cycles spent of the call to sum function, and then calculate in your program the CPI and CPU time (s) of the call using the model we discussed in the class. Your program should also include a printf call to output the four numbers: total instruction count, total cycles, CPI, and CPU time (s). In your submission, please include a screenshot your program execution and output. Please refer to the resource section of the class website (https://passlab.github.io/CSCE513/resources/#papi) for learning how to program using the PAPI interface. https://passlab.github.io/CSCE513/resources/papi_install_run.html page gives instructions for installing and using PAPI. There is a bit tricky of PAPI reading in the papi_example.c file because it does not handle counter reset and counter overflow for the sake of simplicity. It does not always give meaningful number in every run. You need to run multiple times and pick the one that give you meaningful results.

CPU clock rate can be obtained by checking the output of the command "cat /proc/cpuinfo", see below:

```
processor       : 23
vendor_id       : GenuineIntel
cpu family      : 6
model           : 44
model name      : Intel(R) Xeon(R) CPU           X5670  @ 2.93GHz
```

Notes: make sure you store the numbers using variable of float or double data type. A list of other metrics can be derived if you read counters of those native events, checkout this: http://perfsuite.ncsa.illinois.edu/psprocess/metrics.shtml