COMPUTER ARCHITECTURE Homework 1

110062219

March 14, 2023

Contents

1 Install and use AndeSight™ for RISC-V program development 2
1.1 Effects of application program (programming style) on performance 2
1.2 Effects of application program on performance 2
1.3 Effects of compiler on performance 2
1.4 Effects of compiler on performance (cont.) 3
1.5 Effects of instruction set architecture (ISA) on performance 3
1.5.1 What is the difference between RISC and CISC? 3
1.5.2 How ISA design influences the performance qualitatively? . 3
1.5.3 The new instruction 3
1.6 Effects of great ideas on computer performance 4
1.6.1 Pipelined execution 4
1.6.2 Parallel execution 4

2 Compare the performance and battery lives of the two mobile systems 4
2.1 Peak frequency of the most performant block of cores 4
2.2 How long would it take to run a speech recognition on both devices? 5
2.3 How many clock cycles does each core take to run the program on both mobile phones? 5
2.4 Average instruction time 5
2.5 Calculate the relative performance of the two mobile phones 5

3 Amdahl's Law 6
3.1 Which class of instructions you would want to improve first and why? 6
3.2 What is the speedup? 6

4 CPI **6**
4.1 How much must we improve the CPI of FP to run two times faster? 6
4.2 How much is the execution time of the program improved? 6

1 Install and use AndeSight™ for RISC-V program development

1.1 Effects of application program (programming style) on performance

Table 1: Number of instructions & cycles took by the two functions

Func.	# Ins.	# Cyc.
`recur()`	136	245
`iter()`	46	85

It's trivial that a recursive function requires more instructions, and thus more cycles, due to the overhead of maintaining the call stack.

1.2 Effects of application program on performance

CPU times could be computed simply by dividing the number of cycles by the clock rate.

Table 2: Average CPI & CPU time by the two functions

Func.	CPI	CPU Time [ns]
`recur()`	1.80	122.5
`iter()`	1.85	42.5

1.3 Effects of compiler on performance

By taking a look at the disassembled codes, we could easily found see that both the (first) argument and the returned value are stored in the register `a0`.

1.4 Effects of compiler on performance (cont.)

Table 3

Optimizations	Func.	# Ins.	# Cyc.	CPI
`-O0`	`recur()`	242	411	1.70
`-O0`	`iter()`	150	287	1.91
`-O1`	`recur()`	145	222	1.53
`-O1`	`iter()`	36	69	1.92

1.5 Effects of instruction set architecture (ISA) on performance

1.5.1 What is the difference between RISC and CISC?

1. As their names suggest, CISC instructions set is more complex than RISC.
2. Instructions of RISC tend to have a lower CPI than the ones of CISC.
3. The length of instructions of RISC is fixed whereas the length of ones of CISC is variable. As a consequence, RISC could implement pipelining better.

1.5.2 How ISA design influences the performance qualitatively?

Table 4

	IC	CPI	Cycle Time
CISC	more	more	more
RISC	less	less	less

1.5.3 The new instruction

I assume that "it increases the execution cycles of those instructions by 25%" means that the CPI of the new instruction is 1.25 times that of the origin. Otherwise, if the total cycles increase, then the CPU time must be longer.

Suppose the total number of instructions, cycles to be I, C respectively. Then the instruction handling function calls would have a CPI of $\frac{3C}{4I}$ and the CPU time would be $\frac{C}{\text{Clock Rate}}$.

For the new instruction, the number of it would be $0.4I \times 0.65 = 0.26I$ while its CPI would be $\frac{3C}{4I} \times 1.25 = 0.9375\frac{C}{I}$. Thus we have the total number of cycles being $0.26I \times 0.9375\frac{C}{I} + 0.7C = 0.94375C < C$. So we should introduce the new instruction.

1.6 Effects of great ideas on computer performance

Based on the observations in debugging, I found that `sum *= i` (i.e., `mulw a4,a4,a5`) cost a cycle and `i++` (i.e., `c.addiw a5,1` and `c.nop`) cost 2 cycles in total, while the branch (`bge a0,a5,0x104b0`) cost 4 cycles.

1.6.1 Pipelined execution

The iteration would fork into 2 threads. One core would perform the multiplication while the other would increment the index. Then the program would join and branch.

So the number of cycles would be $\max\{1, 2\} + 4 = 6$. This would give us a speedup of $\frac{7}{6}$. It's obvious we could never achieve a 2-speedup, firmly constrained by **Amdahl's Law**.

1.6.2 Parallel execution

If we distribute the 10 iterations into 4 cores, then a core should iterate at most $\lceil\frac{10}{4}\rceil = 3$ thrice. Moreover, we need an additional cycle for B, D to get the partial results. Lastly, we need another cycle again for D to obtain the final result.

Therefore, we need $7 \times 3 + 1 + 1 = 23$ cycles. The reduction percentage is $1 - \frac{23}{7\times 10} \approx 67\%$. Similarly according to **Amdahl's Law**, we must fail to gain a speedup of 4.

2 Compare the performance and battery lives of the two mobile systems

2.1 Peak frequency of the most performant block of cores

Table 5: Snapdragon® 8 Gen 2

	Coretex-X3	Coretex-A715	Coretex-A710	Coretex-A510
Frequency [Mhz]	3200	2800	2800	2000

Table 6: Apple® A16 Bionic

	Everest	Sawtooth
Frequency [Mhz]	3460	2020

2.2 How long would it take to run a speech recognition on both devices?

Table 7

Speech Recognition	Time [s]
Samsung	14180
Apple	12290

2.3 How many clock cycles does each core take to run the program on both mobile phones?

Table 8

Speech Recognition	# Cycles [10^6]
Samsung	4.54×10^7
Apple	4.25×10^7

2.4 Average instruction time

Table 9

	CPI	Highest Freq. [MHz]	Lowest Freq. [MHz]	Highest Ins. Time [μs]	Lowest Ins. Time [μs]
Samsung	0.9	3200	2000	2.81×10^{-4}	4.50×10^{-4}
Apple	3.1	3460	2020	8.96×10^{-4}	1.53×10^{-4}

2.5 Calculate the relative performance of the two mobile phones

$$\sqrt{\frac{141.8}{122.9} \times \frac{243.5}{251.3}} \approx 1.06$$

3 Amdahl's Law

3.1 Which class of instructions you would want to improve first and why?

We should improve C fist, since its product of proportion and CPI is the largest, which means it consumes the most cycles.

3.2 What is the speedup?

The cycle proportion of C, D is $\frac{65\times3}{65\times3+10\times2+25} = 0.81$.

$$S = \frac{1}{(1-0.81)+\frac{0.81}{1.5}} \approx 1.37$$

4 CPI

4.1 How much must we improve the CPI of FP to run two times faster?

The cycle proportion of FP is $\frac{75\times4}{75\times4+120\times3+70\times5+45} \approx 0.28$. By **Amdahl's Law**, we have

$$2 = \frac{1}{(1-0.28)+\frac{0.28}{s}}$$

. There's no positive solution to s.

4.2 How much is the execution time of the program improved?

The cycle proportions of INT, FP and L/S, branch are $\frac{120\times3+75\times4}{75\times4+120\times3+70\times5+45} \approx 0.63$ and $\frac{70\times5+45}{75\times4+120\times3+70\times5+45} \approx 0.37$. So the percentage of CPU time reduction is

$$0.63\times0.29+0.37\times0.56 \approx 0.39$$

.

The original CPU time is $\frac{75\times4+120\times3+70\times5+45}{4\times10^3} \approx 0.26$ second. Hence we would reduce $0.26\times0.39 = 0.10$ second and the improved program would run for 0.16 second.