Project 1 - Question 2 - Part 1

Adil Hydari

October 16, 2024

1 Lab 1

1.1 Part 1

Benchmark	Total # of Instructions	Load %	Store %	Uncond Branch %	Cond Branch %	Integer Computation %	Floating pt Computation %
anagram.alpha	25,597,771	25.36	9.93	4.46	10.30	44.63	5.31
go.alpha	545,812,145	30.62	8.17	2.58	10.96	47.64	0.03
compress.alpha	88,337	1.66	79.05	0.19	5.78	13.30	0.00
gcc.alpha	337,341,325	24.67	11.47	4.12	13.33	46.30	0.11

Table 1: Instruction breakdown for given benchmarks

1.2 Part 2

Alpha Benchmark	Total # of Instructions	Load %	Store %	Uncond Branch %	Cond Branch %	Integer Computation %	Floating pt Computation %
test.math	49,604	17.24	10.38	3.92	11.16	55.27	1.87
test.fmath	19,693	17.88	12.40	4.65	11.50	53.00	0.42
test.llong	10,821	18.09	14.33	5.32	12.78	49.19	0.10
test-printf	983,667	18.00	10.73	4.82	11.39	54.84	0.09

Table 2: Instruction breakdown for Alpha benchmarks

Pisa Benchmark	Total # of Instructions	Load %	Store %	Uncond Branch %	Cond Branch %	Integer Computation %	Floating pt Computation %
test.math	213,745	15.96	10.66	4.22	13.85	54.42	0.88
test.fmath	53,504	16.14	14.41	4.24	15.11	49.95	0.11
test.llong	29,687	16.33	17.99	4.37	15.45	45.82	0.00
test-printf	1,813,937	19.22	9.28	5.13	17.01	49.33	0.01

Table 3: Instruction breakdown for Pisa benchmarks

1.3 Comparison with Histogram

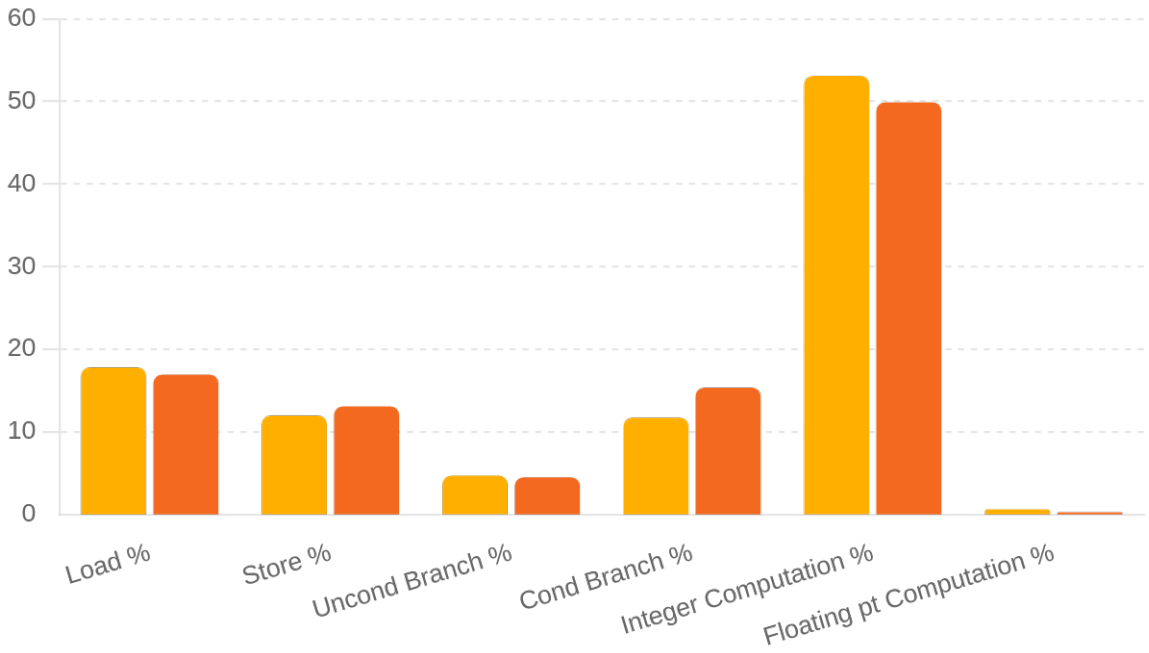

Figure 1: Comparison of Instruction Types (Averages) Between Alpha and Pisa ISAs

Seems like PISA generates a lot more instructions, compared to Alpha, from the compiler, this indicates that the perhaps PISA's MIPs implementation is more CISC like when compared to the Alpha ISA.

2 Lab 6

2.1 Tests

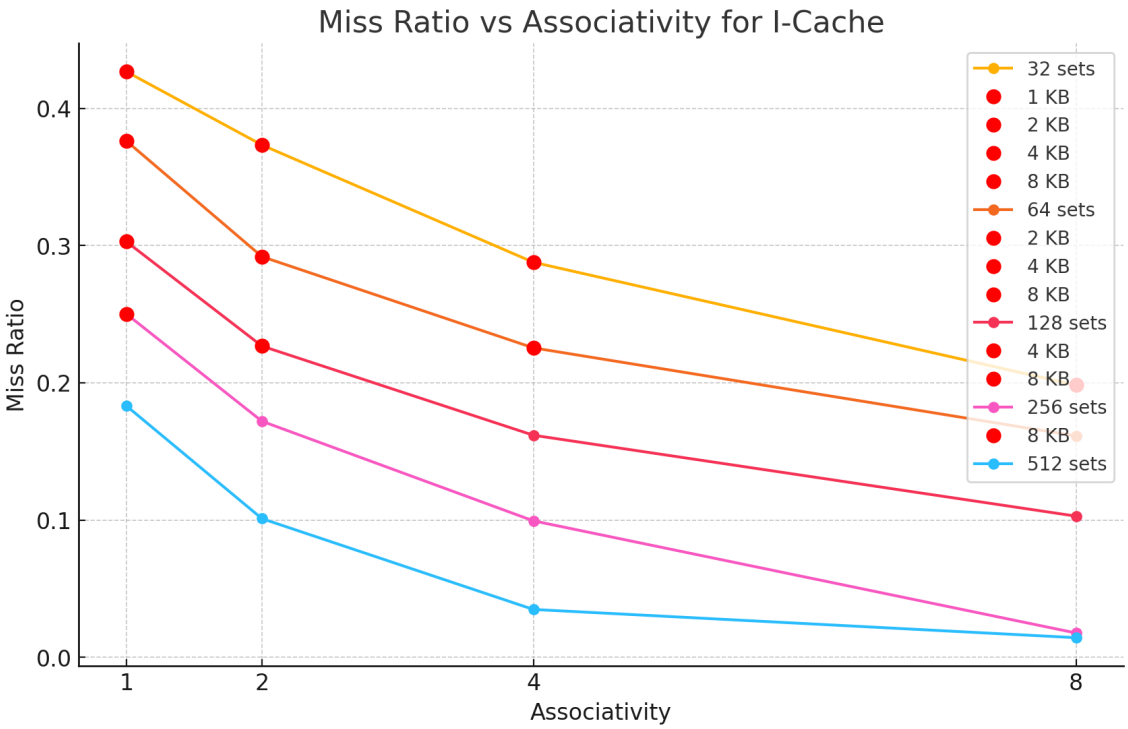

Figure 2: Miss Ratio vs Associativity for I-Cache

Miss Ratio (I-Cache)	**1-way**	**2-way**	**4-way**	**8-way**
32 sets	0.4269	0.3734	0.2879	0.1983
64 sets	0.3764	0.2920	0.2255	0.1613
128 sets	0.3030	0.2268	0.1618	0.1028
256 sets	0.2503	0.1720	0.0994	0.0176
512 sets	0.1833	0.1010	0.0348	0.0142

Table 4: Miss Ratio for I-Cache

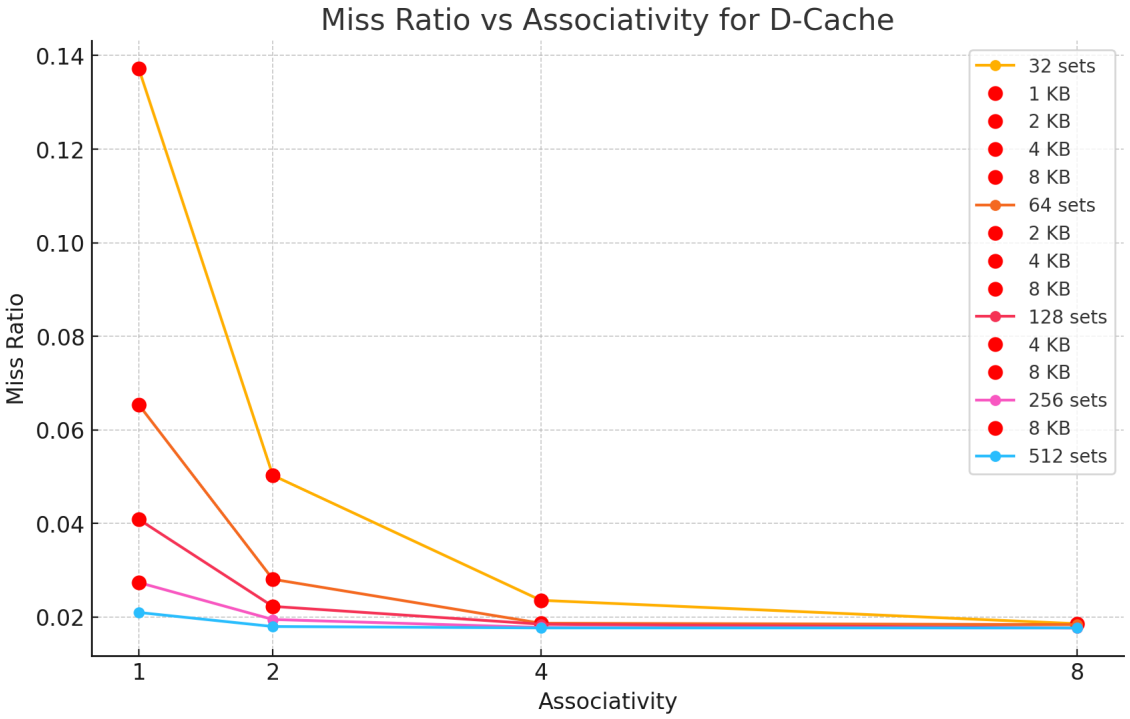

Figure 3: Miss Ratio vs Associativity for D-Cache

Miss Ratio (D-Cache)	1-way	2-way	4-way	8-way
32 sets	0.1372	0.0503	0.0236	0.0186
64 sets	0.0654	0.0281	0.0187	0.0184
128 sets	0.0409	0.0223	0.0185	0.0177
256 sets	0.0274	0.0195	0.0178	0.0177
512 sets	0.0210	0.0180	0.0177	0.0177

Table 5: Miss Ratio for D-Cache

2.2 Cache Simulation Analysis

2.2.1 Questions and Answers

Q1: For a given number of sets, what effect does increasing associativity have on the miss ratio?

For a given number of sets, increasing associativity reduces the miss ratio. This is evident from the tables where, for each fixed number of sets, the miss ratio decreases as the associativity increases. Higher associativity allows more blocks to reside in the same set, reducing conflict misses caused by multiple blocks competing for the same cache location.

Q2: For a given associativity, what is the effect of increasing the number of sets?

For a given associativity, increasing the number of sets decreases the miss ratio. This is because increasing the number of sets effectively enlarges the cache size (since total cache size = number of sets × associativity × block size), allowing more unique blocks to be stored. This reduces capacity misses, as the cache can hold more data.

Q3: For a given cache size, how does the miss ratio change when going from an associativity of one to two to four? Explain.

For a given cache size, increasing associativity from one to two to four generally leads to a reduction in the miss ratio, but the improvement diminishes with higher associativity. This is because, while higher associativity reduces conflict misses by allowing more blocks per set, the total number of unique blocks that can be stored (cache capacity) remains the same. Therefore, the initial increase from direct-mapped (1-way) to 2-way associativity shows a noticeable improvement, but going from 2-way to 4-way yields smaller gains due to diminishing returns.

Q4: If you were to design an Instruction cache, limited to a total cache size of 4 Kbytes, which cache organization would you choose, based solely on performance?

For designing a 4 Kbytes instruction cache based solely on performance, the best choice is a **4-way set-associative cache with 32 sets**. This configuration offers the lowest miss ratio among the options with the same cache size (4 Kbytes). Here's why:

- Cache size = Number of sets × Associativity × Block size
- Assuming a block size of 32 bytes, the configurations that yield 4 Kbytes cache size are:
  - 1-way, 128 sets
  - 2-way, 64 sets
  - 4-way, 32 sets
- From the I-Cache table:
  - 1-way, 128 sets: miss ratio = 0.3030
  - 2-way, 64 sets: miss ratio = 0.2920
  - **4-way, 32 sets: miss ratio = 0.2879 (lowest)**

Q5: If you were to design a data cache, limited to a total cache size of 4 Kbytes, which cache organization would you choose, based solely on performance?

For designing a 4 Kbytes data cache based solely on performance, the optimal choice is also a **4-way set-associative cache with 32 sets**. This configuration provides the lowest miss ratio among the available options with the same cache size. Supporting data:

- Cache size = Number of sets × Associativity × Block size
- Assuming a block size of 32 bytes, the configurations that yield 4 Kbytes cache size are:
  - 1-way, 128 sets
  - 2-way, 64 sets
  - 4-way, 32 sets
- From the D-Cache table:

- 1-way, 128 sets: miss ratio = 0.0409
- 2-way, 64 sets: miss ratio = 0.0281
- **4-way, 32 sets: miss ratio = 0.0236 (lowest)**