Computer Architecture
Homework 7

Tyler Reid, Alex Powell
Due: November 22, 2016

Methodology and Approach

The possible permutations of our cache configurations ranged in the thousands due to differing cache sizes, block sizes, number of sets, and replacement policies. This required a premeditated approach to find the optimal specifications. In order to understand our approach, it necessary to explain the organization of the information provided by the tables. Because cache size is a product of set count, block size, and associativity, we made tables for each associativity type and marked the geometric mean of all six benchmarks IPCs, as it appropriately reflects performance.

For the sake of convenience and to reduce human error, we wrote a bash script that eased the process. It navigated to each folder, ran each benchmark and reported the IPC, then found the geometric mean of these values. This simplified the project by magnitudes.

The left column of our tables is the cache size, while the top row is the block size. Block size for level one instruction cache (il1) and level one data cache (dl1) varied at 8, 16, 32, and 64 bytes, where the cache size fit under the constraints of 8, 16, 32, and 64 KB. This gives us 16 total simulations per associativity. The number of sets varies depending on the block size, so we assume this value implicitly in our calculations. For example, this sample table below shows how set count varies across both x and y labels. This value is assumed throughout the rest of the report and calculated accordingly.

Cache Size	8B Block	16B Block	32B Block	64B Block
8 KB	1024	512	256	128
16 KB	2048	1024	512	256
32 KB	4096	2048	1024	512
64 KB	8192	4096	2048	1024

We recognized that performance of the il1 and dl1 is mostly independent of each other, so we used the default il1 and ul2 configuration to find the optimal il1 cache. Then assuming this value, we found the best dl1, and then the best ul2 using the previously found il1's and dl1's. We also modified each cache latency as needed. Lastly, we tested

the the least recently-used replacement (LRU), first-in first-out (FIFO), and random replacement policy for each cache.

Analysis

For il1 (Table 1), the values trended towards a greater performance as we increased block size and cache size, and thus complexity and overhead. Similarly, we observed improvements when increasing the associativity to 2-way and 4-way. Implicitly, our tables show that a moderate set count is also optimal. We concluded that the best cache size for il1 was 64KB at a 256 set count, 64 block size, and 4-way associativity, or il1:256:64:4:l, at an IPC of 0.4109. However, we recognize that our final column of a 64 block size violated constraint #1 in the instructions, where the ul2 block size must be greater than the il1 + dl1 block size. Therefore, we added an additional column that takes this into account and from this, we can conclude that a block size of 32 is most feasible, or il1:512:32:4:l, with an IPC of 0.4108. Because we doubled the block size, we halved the number of sets to maintain the same ul2 cache size and latency. This behavior is explained because as the block size is increased, the miss rate will decrease. This is because larger block sizes take better advantage of spatial locality by reducing the number of compulsory misses. However, there is a tradeoff in that the miss penalty actually increases because a larger block size means fewer block are in the cache overall which increases both capacity and conflict misses. Our data supports this since a 64 block size performed a bit better than 32, but 32 is the highest number that doesn't violate the constraints.

We conducted all dl1 tests (Table 2) assuming that il1:256:64:4:l was our optimal IPC, though we recognize now it is not. However, the tables remain correct in that they reflect the best possible dl1 is the default, dl1:1028:8:1:l. The simplest possible cache configuration was most optimal. This is a typical example of how using largest cache memory does not necessarily guarantee the best performance. Our optimal associativity for the dl1 was determined to be direct mapped, which provide a good best-case time but will suffer in the worst case.

In our ul2 tests (Table 3), we found simplicity also optimal at 256KB cache size, at ul2:2048:128:1:l, giving us our overall best IPC at 0.4108. The optimal ul2 cache configuration, like dl2, favors simplicity. Again, simply increasing cache size, block size, or associativity is not necessarily a guarantee of improved performance. This adds a lot of overhead to the systems which results in added latencies that could actually cause a smaller IPC, which is what we witnessed here.

Replacement policy was tested for each cache. However, the resulting IPC difference was almost exactly the same, even to the hundredth millionth decimal (Figure 4). In these simulation tests, replacement policy appears negligible. We also used LRU

for all test cases since then, so we assume that value when declaring our best overall cache configuration (Table 5).

Conclusion

In conclusion, we determined the optimal il1 cache to have 512 sets, a block size of 32B, and 4 way associativity. The best dl1 is directly mapped with 1024 sets and an 8B block size. The unified second level cache is also directly mapped with 4096 sets and a 64B block size. These configurations gave us the highest mean IPC of 0.4152, and an individual IPC of 0.4387 for the mcf benchmark. To find the optimal cache configurations we tackled each cache level separately. We started by testing all possible combinations of the il1 cache that didn't violate the given constraints since getting the highest level 1 cache performance would mitigate any shortcomings of the level 2 cache. Only after finding the best configuration for one level did we move on to the next. By following this strategy, and because of the independence between the different cache levels, we can be confident that this is the best cache configuration for the given set of benchmarks.

Results

Il1 4-Way Associativity

Cache Size	8B Block	16B Block	32B Block	64B Block	64B(mod ul2) Block
8KB	0.3714	0.3841	0.3911	0.3972	0.3933
16KB	0.4053	0.4074	0.4079	0.4085	0.4045
32KB	0.401	0.4105	0.4106	0.4106	0.4067
64KB	0.4102	0.4106	0.4108	0.4109	0.4069

Table 1. The blue value represents best feasible IPC. The orange value is the best, violating ul2 block size constraints.

Dl2 Direct Mapped

Cache Size	8B Block	16B Block	32B Block	64B Block
8KB	0.4102	0.399	0.3972	0.3962
16KB	0.3954	0.3859	0.3844	0.3838
32KB	0.3788	0.3704	0.3691	0.3687
64KB	0.3646	0.3568	0.3556	0.3552

Table 2.

Ul2 Direct Mapped

Cache Size	64B Block	128B Block
128KB	0.3518	0.3305
256KB	0.4152	0.4108
512KB	0.4145	0.4104
1024KB	0.4132	0.4092
2048KB	0.4121	0.4084

Table 3.

Optimal Cache Configurations on each Benchmark

Benchmark	IPC
bzip2	0.377
mcf	0.4387
hmmer	0.4185
sjeng	0.437
milc	0.4176
equake	0.4059
IPC Geometric Mean	0.4152

Table 4.

Default and Optimal Cache Configurations and their IPC's

Cache Type	Default Config	Default IPC	Optimal Config	Optimal IPC (Isolated)
Il1	1024:8:1:l	0.3539	512:32:4:l	0.4108
Dl1	1024:8:1:l	0.3539	1024:8:1:l	0.3533
Ul2	1024:64:4:l	0.3539	4096:64:1:l	0.3692
			Overall Optimal IPC	0.4152

Table 5. The last column runs each optimal configuration while the other caches are using their default configuration.

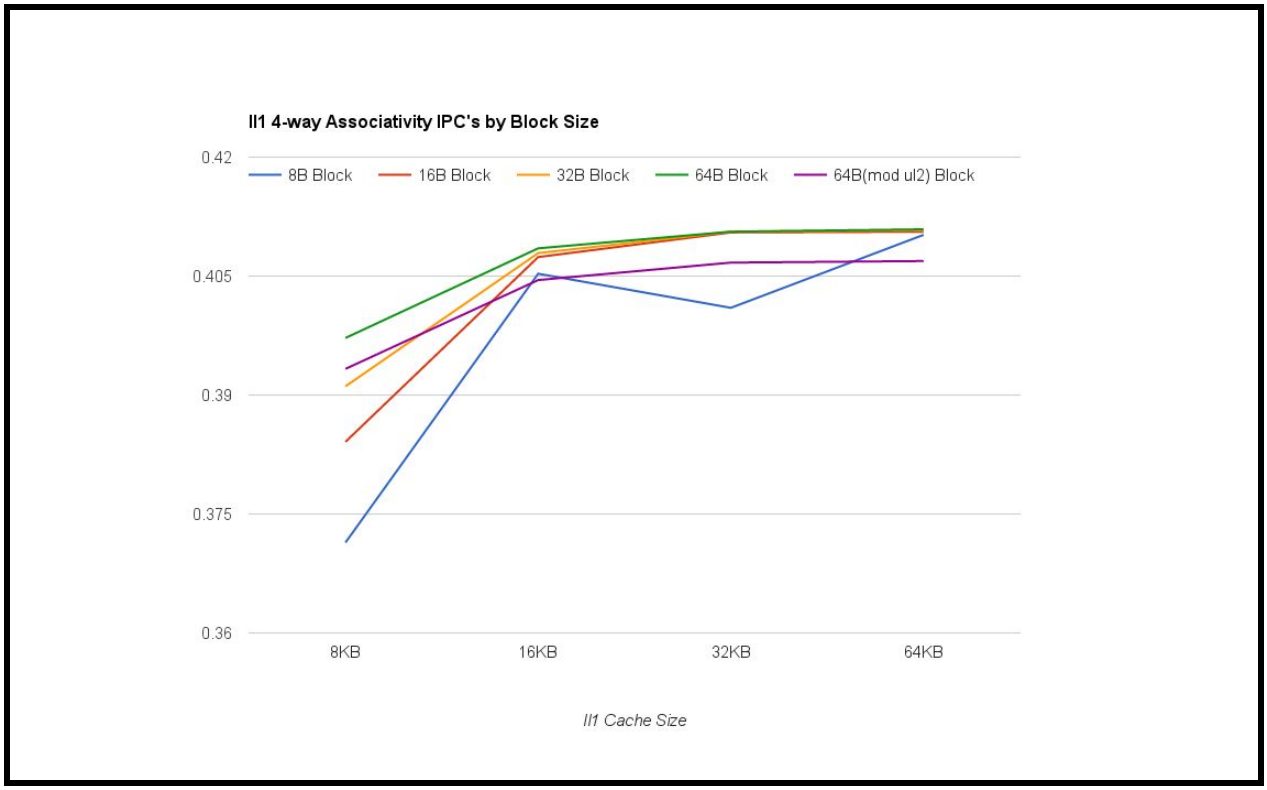

Figure 1.

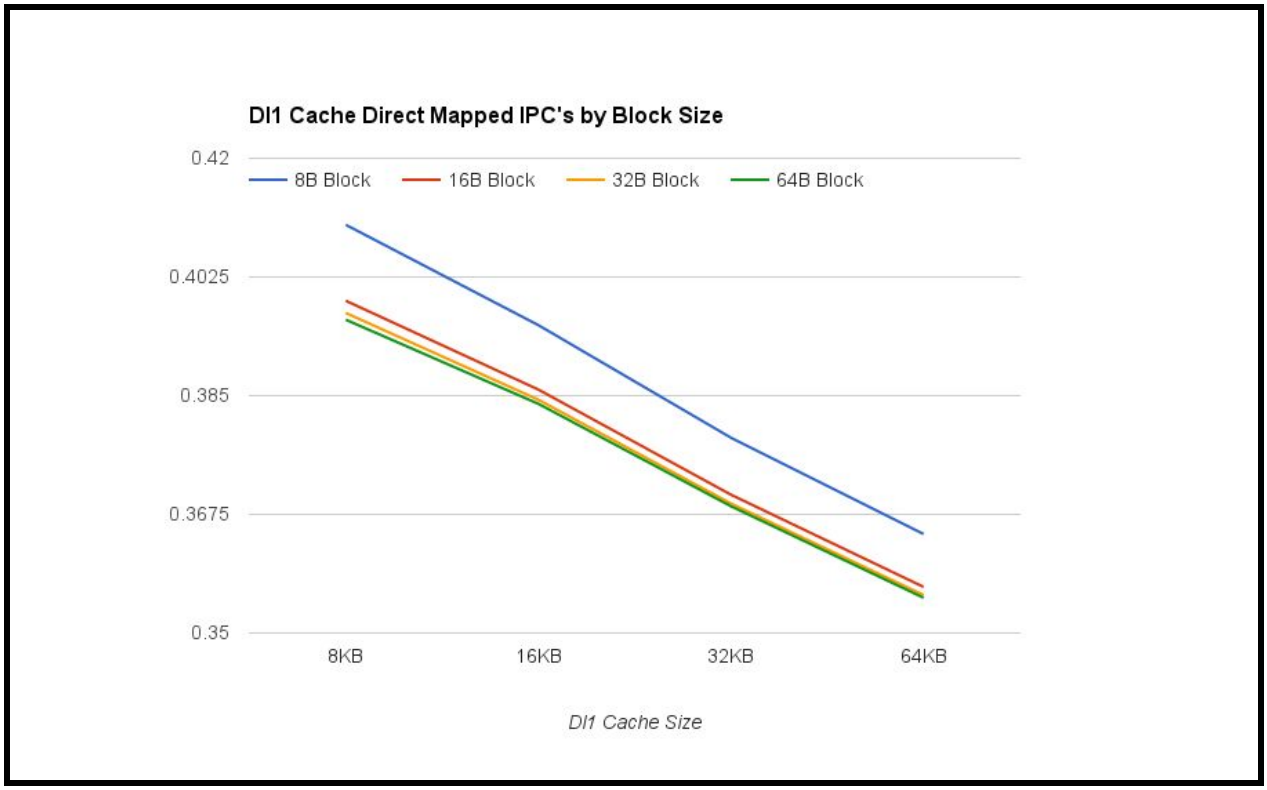

Figure 2.

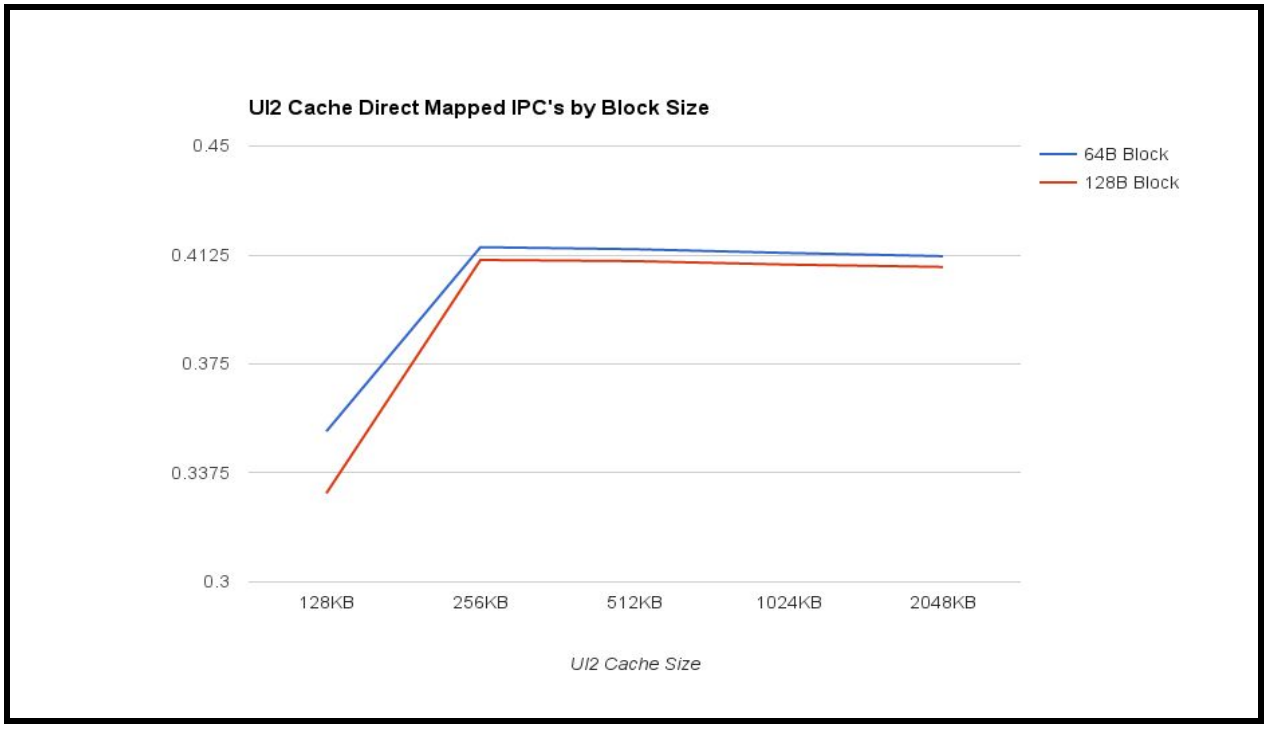

Figure 3.

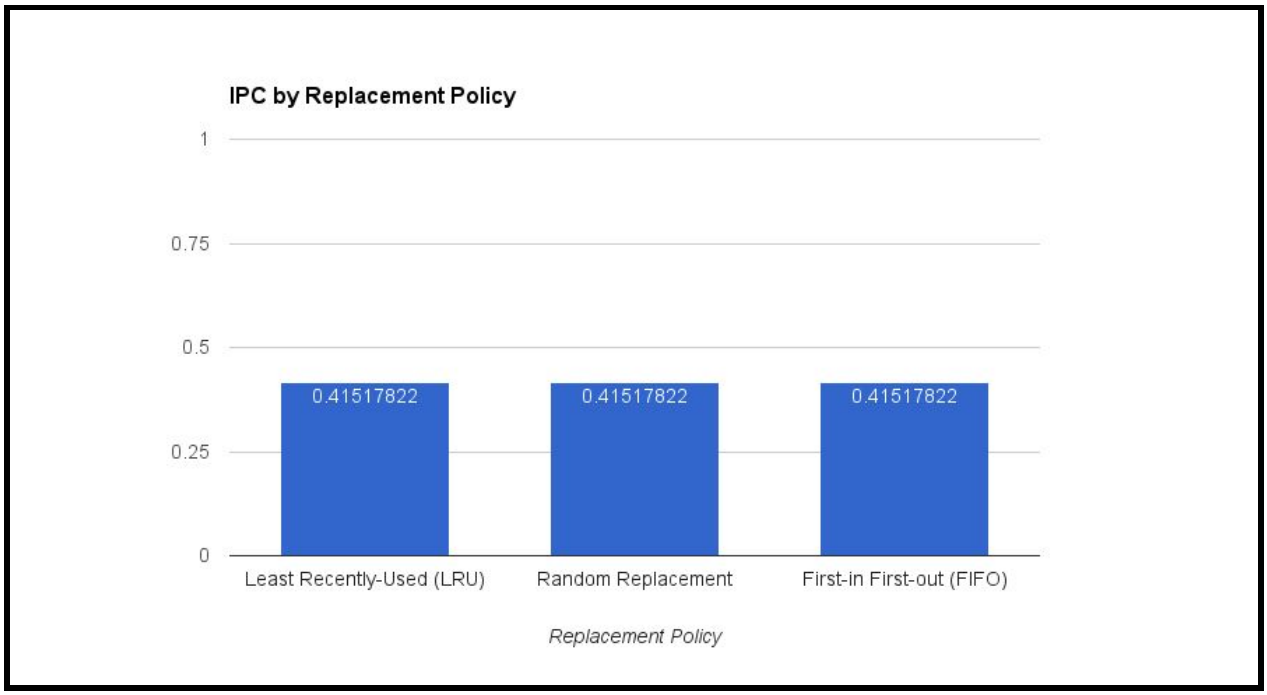

Figure 4.

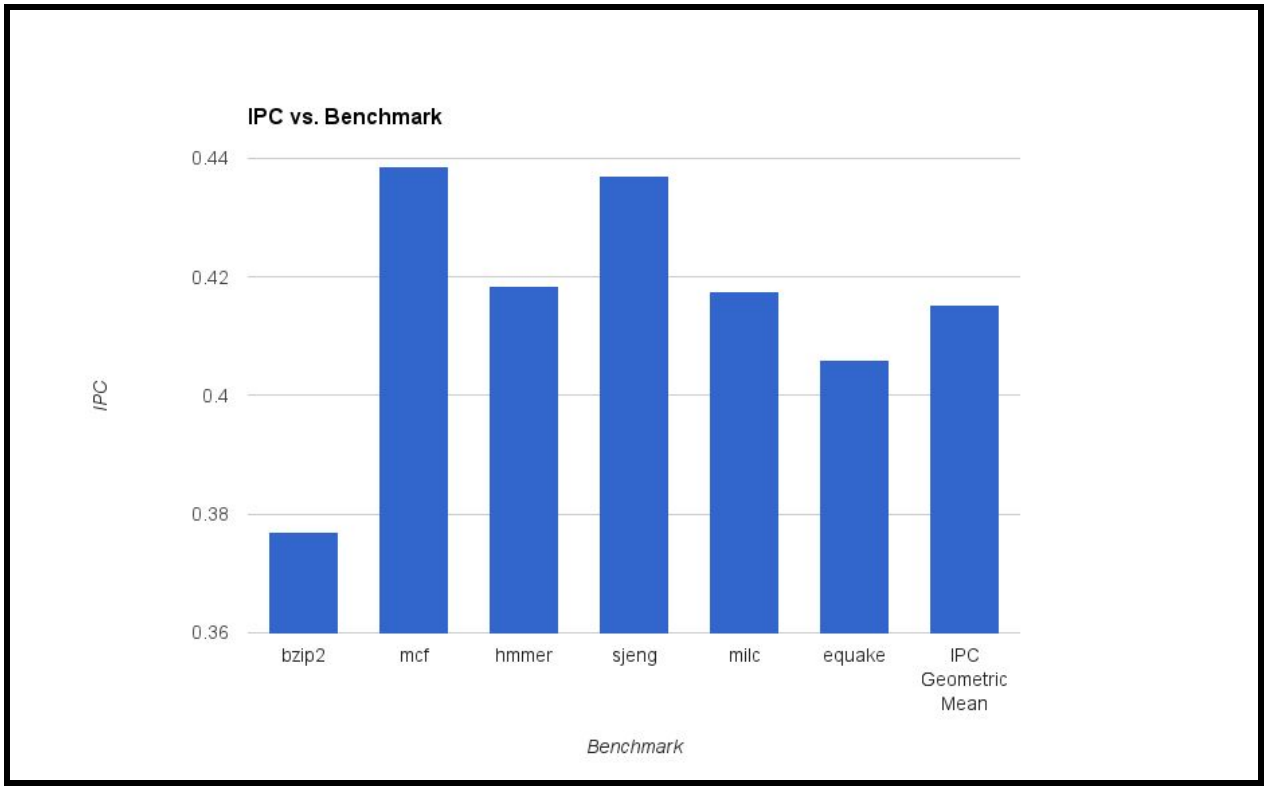

Figure 5.

Optimal config file:

-fastfwd 300000
-max:inst 2000000
-fetch:ifqsize 1
-fetch:speed 1
-decode:width 1
-issue:width 1
-ruu:size 2
-lsq:size 2
-res:ialu 1
-res:imult 1
-res:memport 2
-res:fpalu 1
-res:fpmult 1
-issue:inorder true
-issue:wrongpath false
-cache:il1 il1:512:32:4:l
-cache:il2 dl2
-cache:dl1 dl1:1024:8:1:l
-cache:dl2 ul2:4096:64:1:l
-cache:dl1lat 1
-cache:il1lat 6
-cache:dl2lat 6
-cache:il2lat 6
-mem:lat 51 7
-mem:width 8
-tlb:lat 30
-bpred nottaken
-redir:sim sim.out