Cesar Marin

CSE 420 Assignment 1

Programming Assignment 1

TASK 1.

Know your CPU first! Please state the CPU vendor and the model name you are using for this assignment, with following information.

AMD FX 6300, 64-way (16ways/sub-cache)

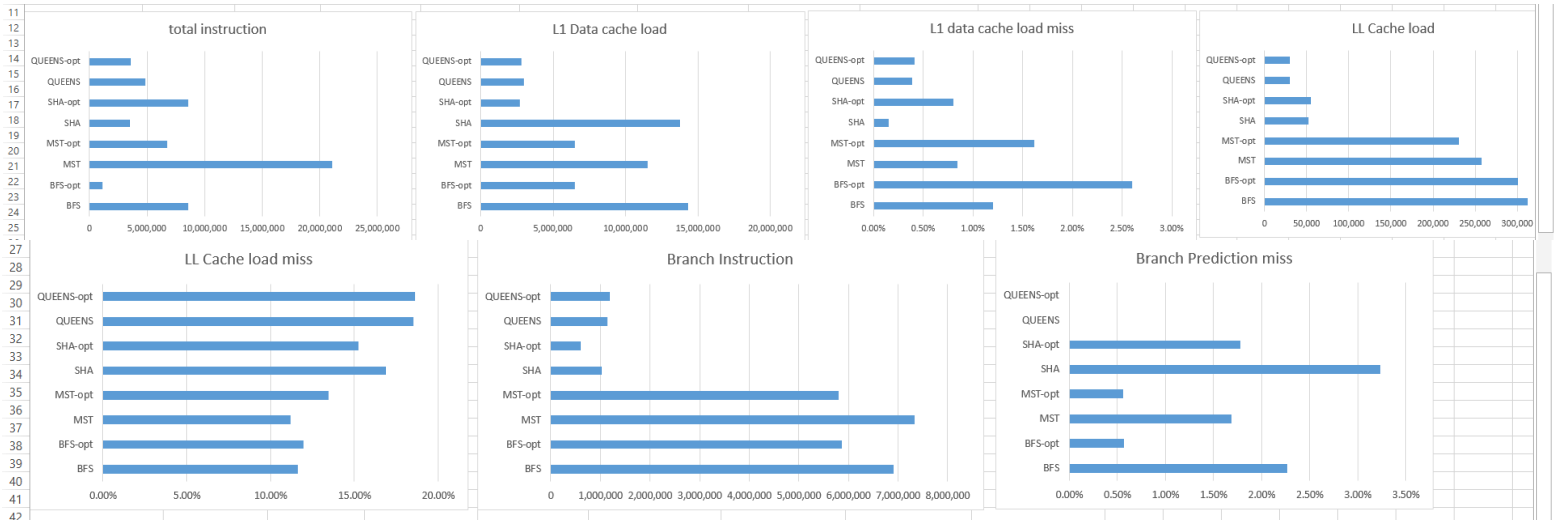
a. CPU frequency: 3.5 GHz

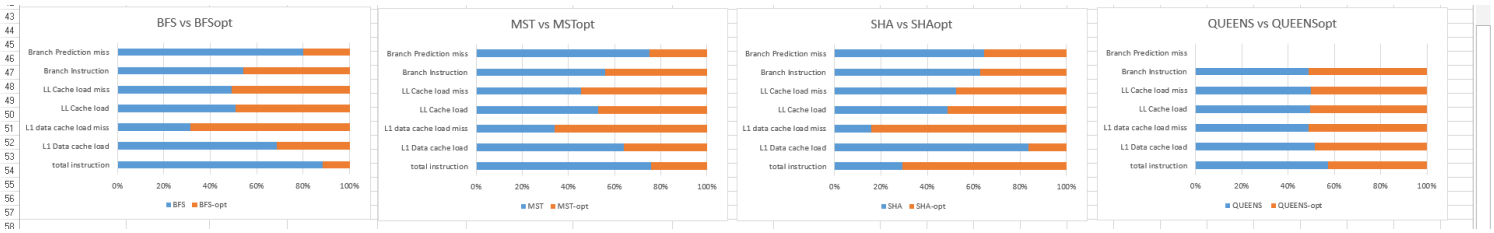
b. Cache hierarchy and size (How many cache level and the size of each level)
L1 cache size – 288KB

L2 cache size – 3x2MB (6MB)

Shared L3 cache size – 8MB

	BFS	BFS-opt	MST	MST-opt	SHA	SHA-opt	QUEENS	QUEENS-opt
total instruction	8,584,501	1,133,566	21,104,065	6,737,753	3,536,433	8,601,501	4,822,659	3,595,154
L1 Data cache load	14,336,530	6,505,382	11,534,097	6,522,327	13,790,623	2,697,533	3,004,210	2,822,736
L1 data cache load miss	1.20%	2.60%	0.84%	1.62%	0.15%	0.80%	0.39%	0.41%
LL Cache load	312,176	300,355	257,754	230,275	52,294	54,850	29,637	30,191
LL Cache load miss	11.64%	11.98%	11.20%	13.46%	16.89%	15.27%	18.53%	18.60%
Branch Instruction	6,918,369	5,862,559	7,338,056	5,797,544	1,019,731	604,442	1,145,562	1,191,104
Branch Prediction miss	2.27%	0.57%	1.69%	0.56%	3.23%	1.78%	0	0
							<not counted>	<not counted>





1. **Describe each workload's characteristic. Which are memory intensive applications? Which one is more cache friendly? Which one does show the challenge to branch prediction?**

   Total instruction: MST seems to be the one with the most instructions
   L1 Data cache load: both SHA and BFS seem to be using L1 the most
   L1 miss rate: Optimized BFS seems to be missing the most cache loads on L1
   LL Cache load: BFS and MST seem to be the ones that use last level cache the most
   LL miss rate: Queens seems to be the one that has the greatest miss rate on LLC, and SHA is close too
   Branch instructions: BFS and MST seems to have the ones with most branches.
   Branch prediction misses: However, SHA seems to cause the most misspredictions (makes sense because of hashing).
   All of these seem to make sense when you think about the nature of the algorithms of each benchmark.

2. **Summarize the change between optimized and non-optimized binaries in the data you collect for all 4 workloads. Do all 4 workloads shows the same behavior change between different compiler optimization? Does compiler optimization change the program attribute like whether they are cache or branch friendly or not?**
   BFS: optimization of bfs seems to help branch misses, as well as total instructions, and a bit of l1 loads, but seems to make l1 miss rate worse.
   MST: optimization of mst seems to help in branch misses and total instructions too, and also makes l1 miss rates worse.
   SHA: SHA_opt really only helps visibly with l1 cache loads. However, the miss rate on l1 increases a lot. Everything else stays around the same as before.
   QUEENS: queens optimization seems to be the one that changes the least, there is very little differences.

TASK 2.

Used 512kB of L2 size (<1MB as said, and power of 2)

Note: on gem5 SHA could not open the input file. Error opening inputs/example-sha-input.txt for reading. Stats were still collected before it tries to open, and these stats are included, but not really useful. I tried removing the .txt file (because it looked like an executable). I created it again

on the same folder, and copied the contents over, and now it looked like a regular file. That still have me the same error.

512kB of LLC	BFS-LRU	BFS-rand	MST-LRU	MST-rand	SHA-LRU	SHA-rand	QUEENS-LRU	QUEENS-rand		BFS-rand
total instruction	1,380,272	1,380,272	1,383,715	1,383,715	148,811	148,811	4,293,800	4,293,800		1,380,272
L1 Data cache load	684,873	688,905	686,192	690,189	106,818	106,818	3,900,960	3,900,960		685,319
L1 data cache load miss	5.34%	5.31%	5.34%	5.33%	12.17%	12.17%	0.17%	0.17%		5.31%
LL Cache load	10,203	10,210	10,290	10,298	2,291	2,291	1,278	1,278		10,210
LL Cache load miss	64.24%	65.03%	63.85%	64.63%	84.29%	84.29%	87.56%	87.56%		63.36%
Branch Instruction	453,071	456,457	455,610	458,224	89,707	89,707	2,375,197	2,375,197		454,313
Branch Prediction miss	5.47%	5.52%	5.42%	5.50%	8.14%	8.14%	12.02%	12.02%		5.45%
					Error on input file			(same?)		Example of stats using full 8MB of l2cache

1. **Which cache replacement policy give you the performance most close to what you measure from your CPU?**
   MST seems to be the one closer in numbers when compared to using perf instead of gem5. Not only in that the numbers are similar, but also in that the replacement policy doesn't change anything in queens, similar to how optimization changed almost nothing.
   Now, overall out of the 2 replacement policies, LRU seems to barely be more similar to my actual CPU. But there is really almost no difference between these two on gem5.