UNIVERSITY OF
GEORGIA
College of Engineering

Cache Project - Unix Team
Fall 2024

Table of Contents

Table of Contents 2
Best Performance: 3
Flow Diagram: 3
Graph of Miss Rates: 4
Discussion Questions: 5
LRU Design: 8

Team Members:
Vinessa Almanza Castillo
Samuel Brewster
Drew Gilliland
Tray Glover
Jake VanEssendelft
Giovanni Zayas

Code can be found at: Github Repository
Document for recording data: Google Spreadsheet

Best Performance:

First we analyzed the Average Memory Access Time (AMAT) to pick our *best* cache designs. We used a process that filtered through each variable one at a time and continued with the best 3 of each category. Results are shown below.

art trace

Cache Config 1:
Associativity: **2**
Block Size: **128**
Cache Size: **128**

Cache Config 2:
Associativity: **4**
Block Size: **128**
Cache Size: **128**

Cache Config 3:
Associativity: **8**
Block Size: **128**
Cache Size: **128**

mcf trace

Cache Config 1:
Associativity: **2**
Block Size: **128**
Cache Size: **128**

Cache Config 2:
Associativity: **4**
Block Size: **128**
Cache Size: **128**

Cache Config 3:
Associativity: **8**
Block Size: **128**
Cache Size: **128**

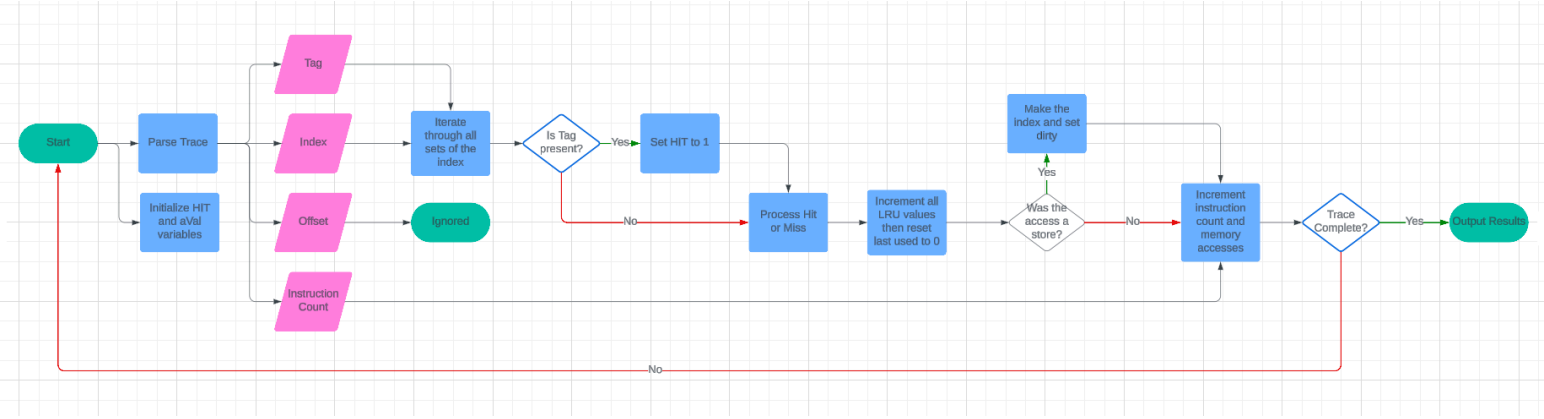
swim trace

Cache Config 1:
Associativity: **1**
Block Size: **128**
Cache Size: **128**

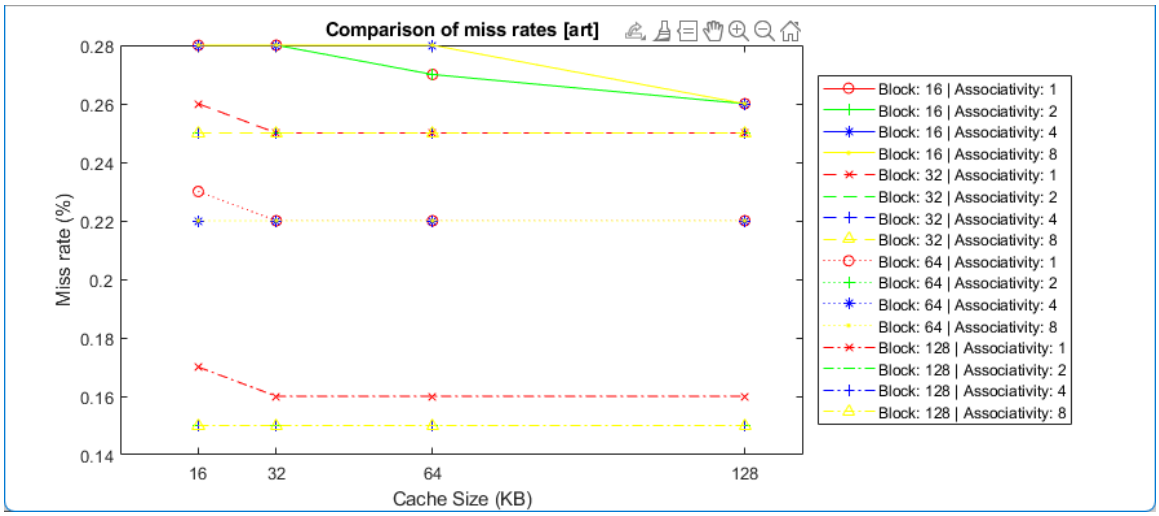
Cache Config 2:
Associativity: **2**
Block Size: **128**
Cache Size: **128**

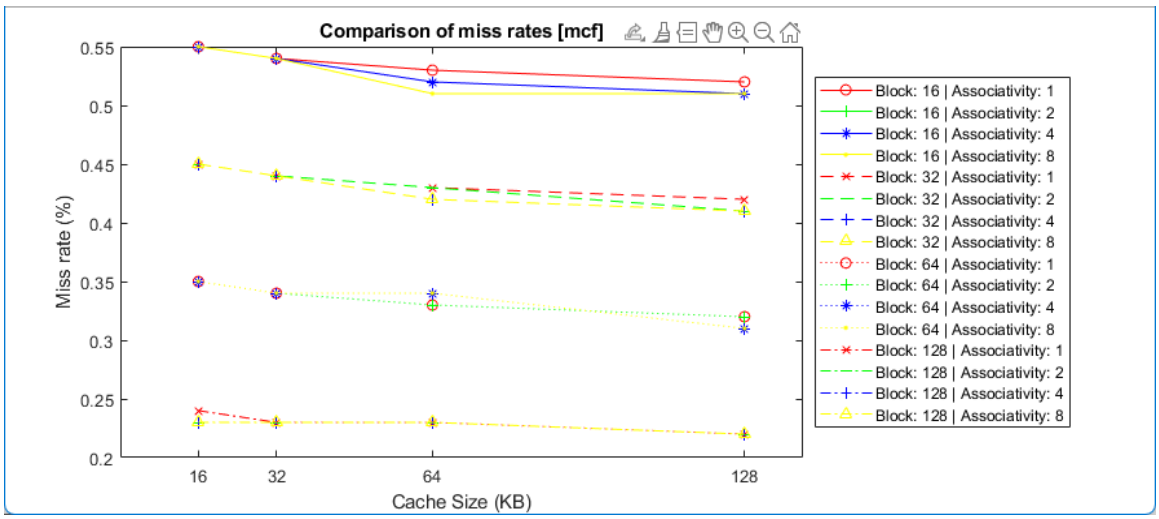
Cache Config 3:
Associativity: **8**
Block Size: **128**
Cache Size: **128**

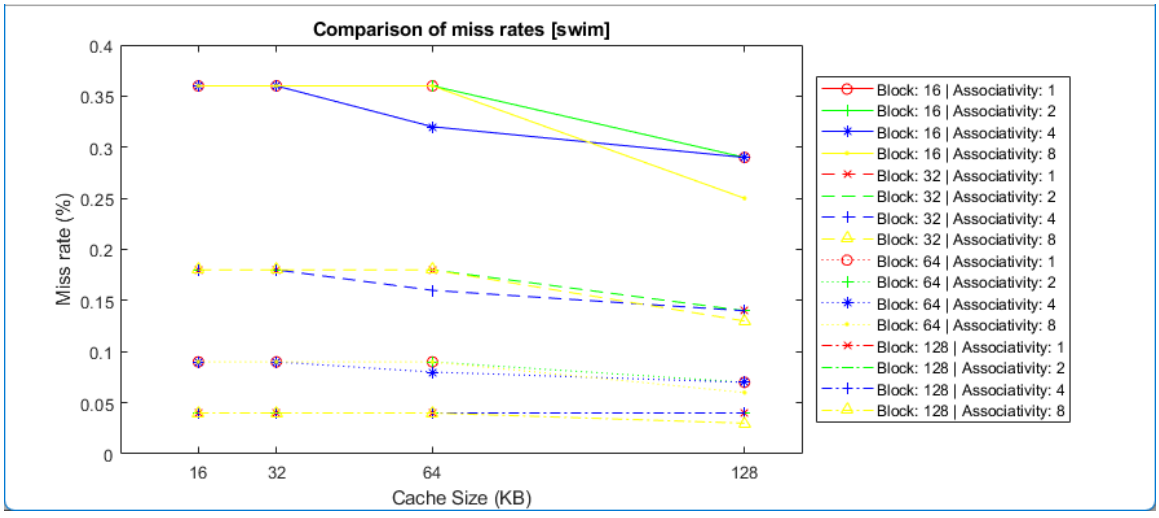
Flow Diagram:



Graph of Miss Rates:







Discussion Questions:

2) Do your results for Total CPI match miss rate results? In other words, do you see better overall Total CPI with better overall miss rate? Discuss.

Based on the art trace, when block size is kept constant (i.e. 16 KB) and cache size is incrementally doubled from 16 bytes up to 128 bytes, the total CPI decreases slowly along with overall miss rate. Data to support this is shown in Figure 1. Note that Figure 1 keeps associativity set to 1 (i.e. direct mapped, since in previous results when determining "best performance" associativity resulted in little influence on actual overall trends).

Figure 1

Cache Size	16	32	64	128
Associativity	1	1	1	1
Block Size	16	16	16	16
Overall Miss Rate	0.28	0.28	0.27	0.26
Total CPI	2.64	2.63	2.57	2.53

However, when cache size and block size are increased together, it appears that while miss rate decreases, total CPI overall shows a general increase (i.e. the relationship between CPI and miss rate is inversely proportional in this instance). Data to support this conclusion is shown in Figure 2.

Figure 2

Cache Size	16	32	64	128
Associativity	1	1	1	1
Block Size	16	32	64	128
Overall Miss Rate	0.28	0.25	0.22	0.16
Total CPI	2.64	2.68	2.79	2.74

3) Does lowest miss rate correlate with overall best execution time? Discuss.

While a lower miss rate can achieve a slight improvement of execution time, it is not necessarily a steady trend and is highly dependent on the configuration of the cache. Typically, the miss rate improves (decreases) with a larger block size, but it is also noted in the project description that a larger block size is not obtainable without a tradeoff. As block size increases, so does the miss penalty. So while the number of misses may decrease, the time it takes to have those misses increases enough to offset the benefit. It is also important to note that a higher cache size and associativity can also very slightly improve the miss rate, but those come at the cost of adding a percent

4) Does lowest CPI correlate with overall best execution time? Discuss.

Similar to the previous question, a lower CPI can contribute to achieving a faster execution time, but they are not necessarily correlated because of the penalties and costs associated with the different block sizes, associativities, and cache sizes that are needed to change the CPI. Typically, a small cache size and high associativity resulted in a high CPI while a large cache size and low associativity resulted in a low CPI, both of which contribute to a percent increase in the execution time.

5) Compare results for the different traces. Do you see any difference based on trace type or were the results uniform? Speculate on results and discuss your findings.

The different traces offer similar trends when comparing Block Size, Associativity, and Cache Size; but all are unique in their statistics even when using the same cache configuration. This is most likely due to the pattern of memory accesses used in each of the traces, for example, one trace may be constantly accessing unique indexes (causing many misses) while another may be accessing the same indexes with the same tag often.

6) Choose what you believe to be the best case design for the cache assuming no limits. Justify choice. What is the difference compared to two default cases (CPI, execution time, and AMAT). What is the final clock rate? Why did you choose this design?

We decided that the best cache design would have 128 KB of Cache Size, 128 Bytes of Block Size, 1 way Associative (direct mapped), and a 3.5 GHz Clock. We decided these values because we see it consistent in our data that a high block size and cache size are the biggest contributing factors to better AMAT and miss rate. We chose a low associativity because the sets do not provide much benefit, hardly even changing the AMAT by 0.01, but accumulate the CPU time penalty (as discussed in the project description).
Using this design provides:
CPI: 3.6 times improvement over no cache, and 1.17 times over the default cache
Execution time: 3.6 times improvement over no cache, and 1.17 times over the default cache

CPU time: 3.09 times improvement over no cache, and 1 times over the default cache
AMAT: 4.59 times improvement over no cache, and 1.24 times over the default cache

7) Now choose best design for fastest overall clock rate. The marketing folks want to advertise the highest clock speed. Now, what is your choice? What is the overall CPI, AMAT, and execution time and final clock rate. Justify choice. Compare to #6. What is your fastest overall clock rate?

We chose the 3.5 GHz clock rate by testing different clock speeds and seeing that each 0.5 GHz added would decrease the CPU time by 0.001 seconds, until the 3.5 GHz mark, where it would take ~1.5 GHz increase to improve the time by 0.001 s, but consistently increasing the miss penalty and total clock cycles ran. At 3.5 GHz, we found a CPI of 4.95, AMAT of 10.36, execution time of 25418888 cycles, and CPU time of 0.008 seconds. We also thought about power consumption on a physical chip increases with clock frequency, and decided that there should be a hard limit at 6 GHz, so we only gathered data from 0.5 - 6 GHz. The clock rate is only 1.5 GHz away from the default 2 GHz (used in Q6).

LRU Design:

An LRU design in hardware can be implemented by using bits at each set that keeps track of the order in which the sets were used. This can, inefficiently, be done by using 1 bit per set in the associativity. The order can be set that the set with a 1 in the MSB was the most recently used, and the set with a 1 in the LSB was the least recently used.

Any time a set is interacted with, its LRU bits would be used with a comparator, and all sets with bits that are greater than the current set will be fed through a shift register that will compute a shift right. The set interacted with will then have a 1 stored in the MSB. Any set with a LRU bit less than the compared set will be unchanged.

When a set needs to be evicted, the LSB can act as the store input to the register because then the least recently used (with a 1 in the LSB) will be stored to. The hardware will then need to shift all of the set's bits right. A shift right with wrap around will set the LRU bits for the evicted set as 1 in the MSB, which is the desired result since we just wrote new data to the set.
This design will work with all associativities, and just needs registers of the same size (2 way = 2 bits, etc.)